Revisiting the Static Model in Robust Reinforcement Learning

Adil Zouitine
IRT Saint-Exupéry
ISAE-SUPAERO
Toulouse, France
adil.zouitine@irt-saintexupery.com

Matthieu Geist
Google Deepmind
Paris, France
mfgeist@google.com

Emmanuel Rachelson
ISAE-SUPAERO
Université de Toulouse
ANITI
Toulouse, France
emmanuel.rachelson@isae-supaero.fr

Abstract

Designing control policies whose performance level is guaranteed to remain above a given threshold in a span of environments is a critical feature for the adoption of reinforcement learning (RL) in real-world applications. The search for such robust policies is a notoriously difficult problem, often cast as a two-player game, whose formalization dates back to the 1970's. This two-player game is strongly related to the so-called dynamic model of transition function uncertainty, where the environment dynamics are allowed to change at each time step. But in practical applications, one is rather interested in robustness to a span of static transition models throughout interaction episodes. The static model is known to be harder to solve than the dynamic one, and seminal algorithms, such as robust value iteration, as well as most recent works on deep robust RL, build upon the dynamic model. In this work, we propose to revisit the static model. We suggest an analysis of why solving the static model under some mild hypotheses is a reasonable endeavor, and formalize the general intuition that robust MDPs can be solved by tackling a series of static problems. We introduce a generic meta-algorithm called IWOCS, which incrementally identifies worst-case transition models so as to guide the search for a robust policy. Discussion on IWOCS sheds light on new ways to decouple policy optimization and adversarial transition functions and opens new perspectives for analysis. We derive a deep RL version of IWOCS and demonstrate it is competitive with state-of-the-art algorithms on classical benchmarks.

1 Introduction

One major obstacle in the way of real-life deployment of reinforcement learning (RL) algorithms is their inability to produce policies that retain, without further training, a guaranteed level of efficiency when controlling a system that somehow differs from the one they were trained upon. This property is referred to as *robustness*, by opposition to *resilience*, which is the ability to recover, through continued learning, from environmental changes. For example, when learning control policies for aircraft stabilization using a simulator, it is crucial that the learned controller be able to control a span of aircraft configurations with different geometries, or masses, or in various atmospheric conditions.

16th European Workshop on Reinforcement Learning (EWRL 2023).

Depending on the criticality of the considered application, one will prefer to optimize the expected performance over a set of environments —thus weighting in the probability of occurrence of a given configuration— or, at the extreme, optimize for the worst case configuration. In the present work, we consider such worst case guarantees and revisit the framework of robust Markov Decision Processes (MDPs) [Iyengar, 2005].

Departing from the common perspective which views robust MDPs as two-player games, we investigate whether it is possible to solve them through a series of non-robust problems. The two-player game formulation is called the dynamic model of transition function uncertainty, as an adversarial environment is allowed to change the transition dynamics at each time step. The solution to this game can be shown to be equivalent, under mild hypotheses, to that of the static model, where the environment retains the same transition function throughout the time steps. Our first contribution is a series of arguments which cast the search for a robust policy as a resolution of the static model (Section 2). We put this formulation in perspective of recent related works in robust RL (Section 3). Then, we introduce a generic meta-algorithm which we call IWOCS for Incremental Worst-Case Search (Section 4). IWOCS builds upon the idea of incrementally identifying worst case transition functions and expanding a discrete uncertainty set, for which a robust policy can be approximated through a finite set of non-robust value functions. We instantiate two IWOCS algorithms, one on a toy illustrative problem with a discrete state space, then another on popular, continuous states and actions, robust RL benchmarks where it is shown to be competitive with state-of-the art robust deep RL algorithms (Section 5).

2 Problem statement

Reinforcement Learning (RL) [Sutton and Barto, 2018] considers the problem of learning a decision making policy for an agent interacting over multiple time steps with a dynamic environment. At each time step, the agent and environment are described through a state $s \in \mathcal{S}$, and an action $a \in \mathcal{A}$ is performed; then the system transitions to a new state s' according to probability $T(s'|s,a)$, while receiving reward $r(s,a,s')$. The tuple $M_T = (\mathcal{S}, \mathcal{A}, T, r)$ forms a Markov Decision Process (MDP) [Puterman, 2014], which is often complemented with the knowledge of an initial state distribution $p_0(s)$. Without loss of generality and for the sake of readability, we will consider a unique starting state s_0 in this paper, but our results extend straightforwardly to a distribution $p_0(s)$. A stationary decision making policy is a function $\pi(a|s)$ mapping states to distributions over actions (writing $\pi(s)$ the action for the special case of deterministic policies). Training a reinforcement learning agent in MDP M_T consists in finding a policy that maximizes the expected discounted return: $J^\pi_T = \mathbb{E}[\sum_{t=0}^\infty \gamma^t r(s_t, a_t, s_{t+1})|s_0, a_t \sim \pi, s_{t+1} \sim T] = V^\pi_T(s_0)$, where V^π_T is the value function of π in MDP M_T. An optimal policy in M_T will be noted π^*_T and its value function V^*_T. A convenient notation is the state-action value function $Q^\pi_T(s,a) = \mathbb{E}_{s' \sim T}[r(s,a,s') + \gamma V^\pi_T(s')]$ of policy π in MDP M_T, and the corresponding optimal Q^*_T.

Robust MDPs, as introduced by Iyengar [2005] or Nilim and El Ghaoui [2005], introduce an additional challenge. The transition functions T are picked from an uncertainty set $\mathcal{T}$ and are allowed to change at each time step. A common assumption, called *sa-rectangularity*, states that $\mathcal{T}$ is a Cartesian product of independent marginal sets of distributions on $\mathcal{S}$, for each state-action pair. The value of a stationary policy π in the sequence of MDPs induced by $\mathbf{T} = \{T_t\}_{t \in \mathbb{N}}$ is noted $V^\pi_{\mathbf{T}}$. The pessimistic value function for π is $V^\pi(s) = \min_{\mathbf{T}} V^\pi_{\mathbf{T}}(s)$, where the agent plays a sequence of actions $a_t \in \mathcal{A}$ drawn from π, against the environment, which in turn picks transition models $T_t \in \mathcal{T}$ so as to minimize the overall return. The robust value function is the largest such pessimistic value function and hence the solution to $V^*(s) = \max_\pi \min_{\mathbf{T}} V^\pi_{\mathbf{T}}(s)$. The robust MDP problem can be cast as the zero-sum two-player game, where $\hat{\pi}$ denote the decision making policy of the adversarial environment, deciding $T_t \in \mathcal{T}$ based on previous observations. Then, the problem becomes $\max_\pi \min_{\hat{\pi}} V^\pi_{\hat{\pi}}(s)$, where $V^\pi_{\hat{\pi}}$ is the expected value of a trajectory where policies π and $\hat{\pi}$ play against each other. Hence, the optimal policy becomes the minimax policy, which makes it robust to all possible future evolutions of the environment's properties.

Robust Value Iteration. Following Iyengar [2005, Theorem 3.2], the optimal robust value function $V^*(s) = \max_\pi \min_{\mathbf{T}} V^\pi_{\mathbf{T}}(s)$ is the unique solution to the robust Bellman equation $V(s) = \max_a \min_T \mathbb{E}_{s' \sim T}[r(s,a,s') + \gamma V(s')] = \mathcal{L}V(s)$. This directly translates into a robust value iteration algorithm which constructs the $V_{n+1} = \mathcal{L}V_n$ sequence of value functions [Satia and Lave Jr, 1973, Iyengar, 2005]. Such robust policies are, by design, very conservative, in particular when the uncertainty set is large and under the rectangularity assumption. Several attempts at mitigating this over-conservativeness have been made from various perspectives. For instance, Lim et al. [2013] propose to learn and tighten the uncertainty set, echoing other works that incorporate knowledge about this set into the minimax resolution [Xu and Mannor, 2010, Mannor et al., 2012]. Other approaches [Wiesemann et al., 2013, Lecarpentier and Rachelson, 2019, Goyal and Grand-Clement, 2022] propose to lift the rectangularity assumption and capture correlations in uncertainties across states or time steps, yielding significantly less conservative policies. Ho et al. [2018] and Grand-Clément and Kroer [2021] retain the rectangularity assumption and propose algorithmic schemes to tackle large but discrete state and action spaces.

The static model. In many applications, one does not wish to consider non-stationary transition functions, but rather to be robust to any transition function from $\mathcal{T}$ which remains stationary throughout a trajectory. This is called the *static* model of transition function uncertainty, by opposition to the *dynamic* model where transition functions can change at each time step. Hence, the static model's minimax game boils down to $\max_\pi \min_T V_T^\pi(s)$. If the agent is restricted to stationary policies $\pi(a|s)$, then $\max_\pi \min_\mathbf{T} V_\mathbf{T}^\pi(s) = \max_\pi \min_T V_T^\pi(s)$ [Iyengar, 2005, Lemma 3.3], that is the static and dynamic problems are equivalent, and the solution to the dynamic problem is found for a static adversary.[1] In this paper, we will only consider stationary policies.

No-duality gap. Wiesemann et al. [2013, Equation 4 and Proposition 9] introduce an important saddle point condition stating that $\max_\pi \min_T V_T^\pi(s) = \min_T \max_\pi V_T^\pi(s)$.

Incrementally solving the static model. Combining the static and dynamic models equivalence and the no-duality gap condition, we obtain that, for rectangular uncertainty sets and stationary policies, the optimal robust value function $V^*(s) = \max_\pi \min_\mathbf{T} V_\mathbf{T}^\pi(s) = \max_\pi \min_T V_T^\pi(s) = \min_T \max_\pi V_T^\pi(s) = \min_T V_T^*(s)$. The key idea we develop in this paper stems from this formulation. Suppose we are presented with M_{T_0} and solve it to optimality, finding $V_0^*(s) = V_{T_0}^*(s)$. Then, suppose we identify M_{T_1} as a possible better estimate of a worst case MDP in $\mathcal{T}$ than T_1. We can solve for $V_{T_1}^*$ and $V_1^*(s) = \min\{V_{T_0}^*(s), V_{T_1}^*(s)\}$ is the robust value function for the discrete uncertainty set $\mathcal{T}_1 = \{T_0, T_1\}$. The intuition we attempt to capture is that by incrementally identifying candidate worst case MDPs, one should be able to define a sequence of discrete uncertainty sets $\mathcal{T}_i = \{T_j\}_{j\in[0,i]}$ whose robust value function V_i^* may converge to V^*. In other words, it should be possible to incrementally robustify a policy by identifying the appropriate sequence of transition models and solving individually for them, trading the complexity of the dynamic model's resolution for a sequence of classical MDP problems. The algorithm we propose in Section 4 follows this idea and searches for robust stationary policies in the static model, by incrementally growing a finite uncertainty set.

3 Related work

Robust RL as two-player games. A common approach to solving robust RL problems is to cast the dynamic formulation as a zero-sum two player game, as formalized by Morimoto and Doya [2005]. In this framework, an adversary, denoted by $\hat{\pi} : \mathcal{S} \rightarrow \mathcal{T}$, is introduced, and the game is formulated as $\max_\pi \min_{\hat{\pi}} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0, a_t \sim \pi(\cdot|s_t), T_t = \hat{\pi}(s_t), s_{t+1} \sim T_t(\cdot|s_t, a_t)]$. Most methods differ in how they constrain $\hat{\pi}$'s action space within the uncertainty set. A first family of methods define $\hat{\pi}(s_t) = T_{ref} + \Delta(s_t)$, where T_{ref} denotes the reference transition function. Among this family, Robust Adversarial Reinforcement Learning (RARL) [Pinto et al., 2017] applies external forces at each time step t to disturb the reference dynamics. For instance, the agent controls a planar monopod robot, while the adversary applies a 2D force on the foot. In noisy action robust MDPs (NR-MDP) [Tessler et al., 2019] the adversary shares the same action space as the agent and disturbs the agent's action $\pi(s)$. Such gradient-based approaches incur the risk of finding stationary points for π and $\hat{\pi}$ which do not correspond to saddle points of the robust MDP problem. To prevent this, Mixed-NE [Kamalaruban et al., 2020] defines mixed strategies and uses stochastic gradient Langevin dynamics. Similarly, Robustness via Adversary Populations (RAP) [Vinitsky et al., 2020] introduces a population of adversaries, compelling the agent to exhibit robustness against a diverse range of potential perturbations rather than a single one, which also helps prevent finding stationary points that are not saddle points. Aside from this first family, State Adversarial MDPs [Zhang

[1] This does not imply the solution to the static model is the same as that of the dynamic model in the general case: the optimal static π may be non-stationary and solving the static model is known to be NP-hard.

et al., 2020, 2021, Stanton et al., 2021] involve adversarial attacks on state observations, which implicitly define a partially observable MDP. The goal in this case is not to address robustness to the worst-case transition function but rather against noisy, adversarial observations. A third family of methods considers the general case of $\bar{\pi}(s_t) = T_t$ where $T_t \in \mathcal{T}$. Minimax Multi-Agent Deep Deterministic Policy Gradient (M3DDPG) [Li et al., 2019] is designed to enhance robustness in multi-agent reinforcement learning settings, but boils down to standard robust RL in the two-agents case. Max-min TD3 (M2TD3) [Tanabe et al., 2022] considers a policy π, defines a value function $Q(s, a, T)$ which approximates $Q^{\pi}_{T}(s, a) = \mathbb{E}_{s' \sim T}[r(s, a, s') + \gamma V^{\pi}_{T}(s')]$, updates an adversary $\bar{\pi}$ so as to minimize $Q(s, \pi(s), \bar{\pi}(s))$ by taking a gradient step with respect to $\bar{\pi}$'s parameters, and updates the policy π using a TD3 gradient update in the direction maximizing $Q(s, \pi(s), \bar{\pi}(s))$. As such, M2TD3 remains a robust value iteration method which solves the dynamic problem by alternating updates on π and $\bar{\pi}$, but since it approximates Q^{π}_{T}, it is also closely related to the method we introduce in the next section.

Regularization. Derman et al. [2021], Eysenbach and Levine [2022] also highlighted the strong link between robust MDPs and regularized MDPs, showing that a regularized policy learned during interaction with a given MDP was actually robust to an uncertainty set around this MDP. Kumar et al. [2023] propose a promising approach in which they derive the adversarial transition function in a closed form and demonstrate that it is a rank-one perturbation of the reference transition function. This simplification results in more streamlined computation for the robust policy gradient.

Domain randomization (DR) [Tobin et al., 2017] learns a value function $V(s) = \max_{\pi} \mathbb{E}_{T \sim \mathcal{U}(\mathcal{T})} V^{\pi}_{T}(s)$ which maximizes the expected return *on average* across a fixed (generally uniform) distribution on $\mathcal{T}$. As such, DR approaches do not optimize the worst-case performance. Nonetheless, DR has been used convincingly in applications [Mehta et al., 2020, OpenAI et al., 2019]. Similar approaches also aim to refine a base DR policy for application to a sequence of real-world cases [Lin et al., 2020, Dennis et al., 2020, Yu et al., 2018].

For a more complete survey of recent works in robust RL, we refer the reader to the work of Moos et al. [2022]. To the best of our knowledge, the approach sketched in the previous section and developed in the next one is the only one that directly addresses the static model. For that purpose, it exploits the equivalence with the dynamic model for stationary policies and solves the dual of the minimax problem, owing to the no-duality gap property.

4 Incremental Worst-case Search

In order to search for robust policies, we consider the no-duality gap property: the best performance one can expect in the face of transition function uncertainty $\max_{\pi} \min_{T} V^{\pi}_{T}(s_0)$, is also the worst performance the environment can induce for each transition function's optimal policy $\min_{T} V^{*}_{T}(s_0)$. If the value $V^{\pi}_{T}(s_0)$ was strictly concave / convex with respect to π / T respectively, we could hope to solve for the robust policy through a (sub)gradient ascent / descent method. Unfortunately, it seems $V^{\pi}_{T}(s_0)$ easily admits more convoluted optimization landscapes, involving stationary points, and local minima and maxima. The $\max_{\pi}$ problem often benefits from regularization [Geist et al., 2019]. Although one could study regularization for the $\min_{T}$ problem [Grand-Clément and Petrik, 2022] or the equivalence with a regularized objective [Derman et al., 2021], we turn towards a simpler process conceptually. We consider a (small) discrete set of MDPs $\mathcal{T}_i = \{T_j\}_{j \in [0,i]}$, for which we derive the corresponding optimal value functions $Q^{*}_{T_j}$. Notably, $\mathcal{T}_i$ is a subset of a (supposed) sa-rectangular uncertainty set, but is not sa-rectangular itself, so there is no guarantee that the static-dynamic equivalence holds in $\mathcal{T}_i$. However, one can consider the sa-rectangular set $\tilde{\mathcal{T}}_i = \bigtimes_{s,a} \{T_j(\cdot|s, a)\}_{j \in [0,i]}$ composed of the cartesian product of all local $\{T_j(\cdot|s, a)\}_{j \in [0,i]}$ sets for each s, a pair. Then we implicitly define Q_i as the function that maps any pair s, a to the smallest expected optimal outcome $Q_i(s, a) = \min_{j \in [0,i]}\{Q^{*}_{T_j}(s, a)\}$. Consequently, Q_i is an upper-bound on the robust Q-function with respect to the $\tilde{\mathcal{T}}_i$ uncertainty set. We abusively call Q_i the $\mathcal{T}_i$-robust value function. In s_0, Q_i coincides with the robust value function for the static model of uncertainty with respect to the $\mathcal{T}_i$ uncertainty set. The corresponding greedy policy is $\pi_i(a) = \arg\max_a Q_i(s, a)$ and is a candidate for the robust policy. Let us write $T_{i+1} = \arg\min_{T \in \mathcal{T}} V^{\pi_i}_{T}(s_0)$. Then, if $V^{\pi_i}_{T_{i+1}}(s_0) = Q_i(s_0, \pi_i(s_0))$, we have found a robust policy for all transition models in $\mathcal{T}$. Otherwise, we can solve for $Q^{*}_{T_{i+1}}$, append T_{i+1} to $\mathcal{T}_i$ to form $\mathcal{T}_{i+1}$ and repeat. Of course, one bottleneck difficulty lies in solving the $\min_{T}$ problem accurately enough, but at least this approach decouples this difficulty from the policy optimization process, which is only concerned with static MDPs. Consequently, the idea we develop is to incrementally expand $\mathcal{T}_i$ by solving $\min_{T \in \mathcal{T}} V_T^{\pi_i}(s_0)$ using optimization methods that can cope with ill-conditioned optimization landscapes. We call Incremental Worst Case Search (IWOCS) this general meta-algorithm, which we summarize in Algorithm 1. We illustrate IWOCS in the current section on a toy example, while Section 4 extends it to the case of high-dimensional state and action spaces, which involve a need for function approximation, and permit only collecting samples in a sub-part of the state space.

Algorithm 1: Incremental Worst-Case Search meta-algorithm

```
Input: $\mathcal{T}$, $T_0$, max number of iterations $M$, tolerance on robust value $\epsilon$
for $i = 0$ to $M$ do
    Find $Q^*_{T_i} = \max_\pi Q^\pi_{T_i}$                                  /* Non-robust policy optimization */
    Define $\mathcal{T}_i = \{T_j\}_{j \in [0,i]}$
    Define $Q_i : s, a \mapsto \min_{j \in [0,i]} \{Q^*_{T_j}(s,a)\}$          /* $\mathcal{T}_i$-robust value function */
    Define $\pi_i(s) = \arg\max_a (Q_i(s,a))$                                   /* Candidate policy */
    Find $T_{i+1} = \arg\min_{T \in \mathcal{T}} V_T^{\pi_i}(s_0)$                 /* Identify worst $T$ */
    if $|V^{\pi_i}_{T_{i+1}}(s_0) - Q_i(s_0, \pi_i(s_0))| \leqslant \epsilon$ then
        return $\pi_i$, $T_{i+1}$, $V^{\pi_i}_{T_{i+1}}(s_0)$                  /* Early termination condition */
return $\pi_M$, $T_{M+1}$, $V^{\pi_M}_{T_{M+1}}(s_0)$
```

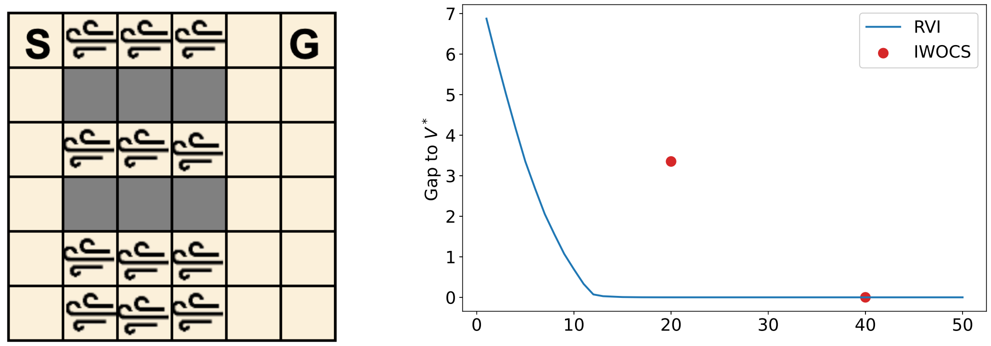
We implement an IWOCS algorithm for robust MDPs, using value iteration (VI) as the policy optimization algorithm and a brute force search across transition functions to identify worst-case MDPs ($V_T^{\pi_i}(s_0)$ is evaluated through Monte-Carlo rollouts). Detailed pseudo-code is provided in Appendix B. Note that any policy optimization method, or optimization method for T, is a suitable fit here. The goal here is to illustrate the behavior of IWOCS, compare it to the seminal robust value iteration (RVI) algorithm, and validate empirically that IWOCS is able to find worst-case static MDPs and robust policies.

This vanilla IWOCS is evaluated on the "windy walk" grid-world MDP illustrated on Figure 1a, where an agent wishes to navigate from a starting position S to a goal G. Actions belong to the discrete $\{N, S, E, W\}$ set and transitions are deterministic, except in the "corridors", where wind can knock back the agent to the left. In the topmost corridor, the probability of being knocked left is α, in the middle corridor it is α^3 and it is α^6 in the bottom corridor (details in Appendix B). Hence, the uncertainty set is fully parameterized by α, which takes 25 discrete values, uniformly distributed in $[0, 0.5]$. Rewards are -1 at each time step and the goal is an absorbing state yielding zero reward.

Figure 1b illustrates how IWOCS converges to the robust value function V^*. RVI builds the sequence $V_{n+1} = \mathcal{L}V_n$ and we plot $|V_n(s_0) - V^*(s_0)|$ versus the number of robust Bellman iterates n. On the other hand, IWOCS requires its first policy optimization to terminate before it can report its first $Q_i(s_0, \pi_i(s_0))$. Thus, we plot $|Q_i(s_0, \pi_i(s_0)) - V^*(s_0)|$ after a fixed number of 100 standard Bellman backups for VI. It is important to note that one iterate of the standard Bellman operator requires solving a $\max_a$ in each state, while an iterate of the robust Bellman operator requires solving a more costly $\max_a \min_T$ problem in each state. Therefore, the x-axis does not account for computational time. IWOCS finds the worst-case static model after two iterations and converges to the same value as RVI.

5 Deep IWOCS

We now turn towards challenging robust control problems and introduce an instance of IWOCS meant to accommodate large and continuous state and action spaces, using function approximators such as neural networks. This instance of IWOCS uses Soft Actor Critic (SAC) Haarnoja et al. [2018] as the policy optimization method. One motivation for using SAC is that it has been proven to yield a locally robust policy around the MDP it is trained upon [Eysenbach and Levine, 2022].



(a) Windy walk grid-world.

(b) Convergence to V^* versus Bellman iterates.

Figure 1: IWOCS and RVI.

5.1 Method

Accounting for regularization terms. Since SAC learns a regularized Q-function which accounts for the policy's entropy, and lets the importance of this term vary along the optimization, orders of magnitude may change between Q_{T_i} and Q_{T_j}. To avoid the influence of this regularization term when defining the $\mathcal{T}_i$-robust Q-function, we train an additional unregularized Q-network which only takes rewards into account. We call π_T the policy network which approximates the optimal policy of the regularized MDP based on T. This policy's (regularized) value function is approximated by the Q'_T network (our implementation uses double Q-networks as per the common practice — all details in Appendix C), while an additional Q_T network (double Q-network also) tracks the unregularized value function of π_T. The $\mathcal{T}_i$-robust Q-function is defined with respect to this unregularized value function as $Q_i(s,a) = \min_{j\in[0,i]}\{Q_{T_j}(s,a)\}$.

Partial state space coverage. In large state space MDPs, it is likely that interactions will not explore the full state space. Consequently, the different Q_{T_j} functions are trained on replay buffers whose empirical distribution's support my vary greatly. Evaluating neural networks outside of their training distribution is prone to generalization errors. This begs for indicator functions specifying on which (s,a) pair each Q_T is relevant. We chose to implement such an indicator function using predictive coding [Rao and Ballard, 1999] on the dynamical model T_j. Note that other choices can be equally good (or better), such as variance networks [Neklyudov et al., 2019], ensembles of neural networks [Lakshminarayanan et al., 2016] or 1-class classification [Béthune et al., 2023]. Our predictive coding model for T_j predicts $\hat{T}_j(s,a) = s'$ for deterministic dynamics, by minimizing the expected value of the loss $\ell(\hat{T}_j(s,a);s') = \|\hat{T}_j(s,a) - s'\|_1$. At inference time, along a trajectory, we consider Q_{T_j} has been trained on sufficient data in s_t, a_t, if $\ell(\hat{T}_j(s_{t-1},a_{t-1});s_t) \leqslant \rho_j$, ie. if the prediction error for s_t is below the threshold ρ_j (details about tuning ρ_j in Appendix D). We redefine $Q^*_{T_j}$ to be $+\infty$ in all states where $\ell(\hat{T}_j(s_{t-1},a_{t-1});s_t) > \rho_j$, so that it does not participate in the definition of Q_i.

Worst case identification. When $V_T^{\pi_i}(s_0)$ is non differentiable with respect to T (or T's parameters), one needs to fall back on black-box optimization to find $T_{i+1} = \arg\min_{T\in\mathcal{T}} V_T^{\pi_i}(s_0)$. We turn to evolutionary strategies, and in particular CMA-ES [Hansen and Ostermeier, 2001] for that purpose, for its ability to escape local minima and efficiently explore the uncertainty set $\mathcal{T}$ (hyperparameters in Appendix C). Note that making $V_T^{\pi}(s_0)$ differentiable with respect to T is feasible by making the critic network explicitly depend on T's parameters, as in the work of Tanabe et al. [2022]. We do not resort to such a model, as it induces the risk for generalization errors, but it constitutes a promising alternative for research. To evaluate $V_T^{\pi_i}(s_0)$ for a given T, we run a roll-out from s_0 by applying $\pi_i(s)$ in each encountered state s. Since we consider continuous action spaces and keep track of the critics Q_{T_j}, Q'_{T_j} and the actor π_{T_j} for all $T_j \in \mathcal{T}_i$, we can make direct use of π_{T_j} which is designed to mimic an optimal policy in M_{T_j}. Specifically, in s, we evaluate $j* = \arg\min_{j\leqslant i} Q^*_{T_j}(s, \pi_{T_j}(s))$, and apply $\pi_i(s) = \pi_{j*}(s)$. If no $Q^*_{T_j}$ is valid in s, we fall back to a default policy trained with domain randomization.

5.2 Empirical evaluation

Experimental framework. This section assesses the proposed algorithm's worst-case performance and generalization capabilities. Experimental validation was performed on optimal control problems using the MuJoCo simulation environments [Todorov et al., 2012]. The algorithm was benchmarked against state-of-the-art robust reinforcement learning methods, including M2TD3 [Tanabe et al., 2022], M3DDPG [Li et al., 2019], and RARL [Pinto et al., 2017]. We also compare with Domain Randomization (DR) [Tobin et al., 2017] for completeness. For each environment, two versions of the uncertainty set are considered, following the benchmarks reported by Tanabe et al. [2022]. In the first one, T is parameterized by a global friction coefficient and the agent's mass, making $\mathcal{T}$ isomorphic to $\mathbb{R}^2$. In the second one, a third, environment-dependent parameter is included (details in Appendix G). To ensure a fair comparison, performance metrics were collected after 4 million steps for environments with a 2D uncertainty set and after 5 million steps for those with a 3D uncertainty set. All reference methods optimize a single policy along these 4 or 5 million steps, but IWOCS optimizes a sequence of non-robust policies, for which we divided this sample budget: we constrained IWOCS to train its default policy and each subsequent SAC agent for a fixed number of interaction steps, so that the sum is 4 or 5 million steps. All results reported below were averaged over 10 distinct random seeds.

Worst-case performance. Table 1 reports the normalized worst-case scores comparing IWOCS, M2TD3, SoftM2TD3, M3DDPG, RARL, and DR using TD3. The worst-case score for the policy returned by IWOCS is a direct by-product of the algorithm's last steps, since they search for worst-case transition functions that minimize the value of Monte-Carlo rollouts. The worst-case scores for all other methods are drawn from the M2TD3 paper [Tanabe et al., 2022] where they are evaluated by defining a uniform grid over the transition function's parameter space and performing roll-outs for each transition model[2]. To obtain comparable metrics across environments, we normalized each method's score v using the vanilla TD3 (trained on the environment with default transition function parameters) reference score v_{TD3} as a minimal baseline and the M2TD3 score v_{M2TD3} as target score: $(v - v_{TD3})/|v_{M2TD3} - v_{TD3}|$. Hence this metric reports how much a method improves upon TD3, compared to how much M2TD3 improved upon TD3. The non-normalized scores are reported in Appendix H. This instance of IWOCS demonstrates competitive performance, outperforming all other methods in 7 out of the 11 environments (note that we did not report results on environments from the literature that feature a simpler 1D uncertainty set). IWOCS permits a 2.36-fold improvement on average across environments, over the state-of-the-art M2TD3. It seems that Ant is an environment where IWOCS struggles to reach convincing worst case scores (and Humanoid to a lesser extent). We conjecture this is due to the wide range of possible mass and friction parameters, which make the optimization process very noisy (almost zero mass and friction is a worst-case T making the ant's movement rather chaotic and hence induces a possibly misleading replay buffer) and may prevent the policy optimization algorithm to yield good non-robust policies and value functions given its sample budget. However, IWOCS provides a major (up to 10-fold) improvement on Hopper.

Table 1: Avg. of normalized worst-case performance over 10 seeds for each method.

Environment	IWOCS	M2TD3	SoftM2TD3	M3DDPG	RARL	DR (TD3)
Ant 2	-0.19 ± 0.87	$\mathbf{1.00 \pm 0.04}$	0.92 ± 0.06	-0.72 ± 0.05	-1.32 ± 0.04	0.02 ± 0.05
Ant 3	0.45 ± 0.59	$\mathbf{1.00 \pm 0.09}$	0.97 ± 0.18	-0.36 ± 0.20	-1.28 ± 0.06	0.61 ± 0.03
HalfCheetah 2	$\mathbf{1.35 \pm 0.28}$	1.00 ± 0.05	1.07 ± 0.05	-0.02 ± 0.02	-0.05 ± 0.02	0.84 ± 0.04
HalfCheetah 3	$\mathbf{1.84 \pm 0.56}$	1.00 ± 0.14	1.39 ± 0.15	-0.03 ± 0.05	-0.13 ± 0.05	1.10 ± 0.04
Hopper 2	$\mathbf{5.56 \pm 0.74}$	1.00 ± 0.05	1.09 ± 0.06	0.46 ± 0.06	0.61 ± 0.17	0.87 ± 0.03
Hopper 3	$\mathbf{9.32 \pm 0.99}$	1.00 ± 0.09	0.68 ± 0.08	0.22 ± 0.04	0.56 ± 0.17	0.73 ± 0.13
HumanoidStandup 2	0.82 ± 0.18	1.00 ± 0.12	$\mathbf{1.25 \pm 0.16}$	0.98 ± 0.12	0.88 ± 0.13	1.14 ± 0.14
HumanoidStandup 3	0.90 ± 0.63	$\mathbf{1.00 \pm 0.11}$	0.96 ± 0.07	0.97 ± 0.07	0.88 ± 0.13	0.86 ± 0.06
InvertedPendulum 2	$\mathbf{2.82 \pm 0.00}$	1.00 ± 0.37	0.38 ± 0.08	-0.00 ± 0.00	-0.00 ± 0.00	0.15 ± 0.01
Walker 2	$\mathbf{1.26 \pm 0.34}$	1.00 ± 0.14	0.83 ± 0.15	0.04 ± 0.04	-0.08 ± 0.01	0.71 ± 0.17
Walker 3	$\mathbf{1.87 \pm 0.50}$	1.00 ± 0.23	1.03 ± 0.20	0.06 ± 0.05	-0.10 ± 0.01	0.65 ± 0.19
Aggregated	$\mathbf{2.36 \pm 0.51}$	1.00 ± 0.13	0.96 ± 0.11	0.14 ± 0.06	0.0 ± 0.07	0.70 ± 0.08

[2] The same grid was used as a sanity check for IWOCS to verify that CMA-ES indeed found the worst-case transition function.

Table 2: Avg. of normalized average performance over 10 seeds for each method.

Environment	IWOCS	M2TD3	SoftM2TD3	M3DDPG	RARL	DR (TD3)
Ant 2	0.45 ± 0.51	1.00 ± 0.02	1.04 ± 0.00	-0.13 ± 0.12	-1.04 ± 0.02	$\mathbf{1.28} \pm \mathbf{0.03}$
Ant 3	-3.56 ± 2.96	-1.00 ± 0.44	-0.36 ± 0.46	-6.98 ± 0.44	-8.94 ± 0.18	$\mathbf{0.92} \pm \mathbf{0.22}$
HalfCheetah 2	1.74 ± 0.26	1.00 ± 0.03	1.10 ± 0.04	-1.08 ± 0.07	-1.94 ± 0.03	$\mathbf{1.84} \pm \mathbf{0.09}$
HalfCheetah 3	1.90 ± 0.32	1.00 ± 0.07	1.17 ± 0.03	-1.43 ± 0.14	-2.48 ± 0.05	$\mathbf{2.33} \pm \mathbf{0.12}$
Hopper 2	$\mathbf{1.90} \pm \mathbf{0.80}$	1.00 ± 0.07	0.74 ± 0.12	-0.40 ± 0.11	1.86 ± 0.92	0.36 ± 0.08
Hopper 3	$\mathbf{6.17} \pm \mathbf{3.30}$	-1.00 ± 0.23	-1.20 ± 0.13	-2.20 ± 0.37	1.63 ± 1.53	1.17 ± 0.23
HumanoidStandup 2	$\mathbf{3.00} \pm \mathbf{3.00}$	-1.00 ± 0.67	0.67 ± 0.83	-1.83 ± 0.67	-3.00 ± 1.33	0.50 ± 0.67
HumanoidStandup 3	$\mathbf{1.87} \pm \mathbf{4.12}$	1.00 ± 0.75	0.38 ± 0.37	0.00 ± 0.50	-1.75 ± 0.87	0.38 ± 0.87
InvertedPendulum 2	$\mathbf{2.86} \pm \mathbf{0.00}$	1.00 ± 0.68	1.06 ± 0.46	-1.10 ± 0.25	-0.47 ± 0.32	2.47 ± 0.03
Walker 2	$\mathbf{1.03} \pm 0.45$	1.00 ± 0.06	0.83 ± 0.16	-0.53 ± 0.11	-1.21 ± 0.02	0.91 ± 0.15
Walker 3	$\mathbf{1.19} \pm \mathbf{0.50}$	1.00 ± 0.13	0.96 ± 0.18	-0.57 ± 0.09	-1.43 ± 0.04	1.13 ± 0.10
Aggregated	$\mathbf{1.68} \pm \mathbf{1.47}$	0.45 ± 0.28	0.58 ± 0.25	-1.47 ± 0.25	-1.70 ± 0.48	1.20 ± 0.23

Average performance. While our primary aim is to maximize the worst-case performance, we also appreciate the significance of average performance in real-world scenarios. Table 2 reports the normalized average score (non-normalized scores in Appendix H) obtained by the resulting policy over a uniform grid of 100 transition functions in 2D uncertainty sets (1000 in 3D ones). Interestingly, M3DDPG and RARL feature negative normalized scores and perform worse on average than vanilla TD3 on most environments (as M2TD3 on 3 environments). DR and IWOCS display the highest average performance. Although this outcome was anticipated for DR, it may initially seem surprising for IWOCS, which was not explicitly designed to optimize mean performance. We posit this might be attributed to two factors. First, in MDPs which have not been identified as worst-cases, encountered states are likely to have no valid Q_{T_j} value function. In these MDPs, if we were to apply any of the π_{T_j}, its score could be as low as the worst cast value (but not lower, otherwise the MDP should have been identified as a worst case earlier). But since IWOCS' indicator functions identify these states as unvisited, the applied policy falls back to the DR policy, possibly providing a slightly better score above the worst case value for these MDPs. Second, the usage of indicator functions permits defining the IWOCS policy as an aggregate of locally optimized policies, possibly avoiding averaging issues. As for the worst-case scores, IWOCS does not perform well on Ant environments. However, it provides better average scores than both DR and M2TD3 on the Humanoid benchmarks. Overall, it performs better than all other methods on 7 out of the 11 environments and is only slightly outperformed by DR on 2 more.

Tracking the convergence of IWOCS. IWOCS aims at solving iteratively the robust optimization problem by covering the worst possible case at each iteration. Consequently, we could expect the value of the candidate robust policy π_i to increase throughout iterations. Figure 2 reports the score of π_i across iterations, averaged over 10 optimization runs, along with the observed standard deviation, on all 3D uncertainty sets. In the three environments where IWOCS dominates over other reference methods, we observe a nice steady increase in value, with limited standard deviations across runs. Notably, in the Hopper environment, the search for a worst-case environment seems very beneficial to IWOCS, enabling a large leap in score over the first iteration (causing the 10-fold increase reported in Table 1). Results on the Ant benchmark seem to confirm the conjecture made to explain the poor behavior of IWOCS: the large variance in π_i scores tend to indicate that optimizing for non-robust policies is somewhat noisy and might prevent finding robust policies overall.

Worst case paths in the uncertainty set. IWOCS seems to reliably find worst case MDPs and policies in a number of cases. Table 3 permits tracking the worst case MDPs and policies along iterations for the HalfCheetah 2D environment (all other results in Appendix I). Specifically, it reports the score of the first optimized policy $J_{T_0}^{\pi_{T_0}}$, then the parameters ψ_1^1 and ψ_2^1 (friction and mass) of the worst case transition function T_1, and the value $J_{T_1}^{\pi_0}$ of the candidate robust policy π_0 in T_1. The next columns repeats these values for later iterations. Each line corresponds to a different random seed. At iteration 0, the first trained policy π_{T_0} displays a large variance in score across runs, as is often the case with deep RL methods (SAC included). In turn, it is no surprise that the identified worst case MDPs differ from one line to the other and that their scores features some variability either. However, all instances of T_1 tend to lie on some corner of the uncertainty set. As IWOCS move on

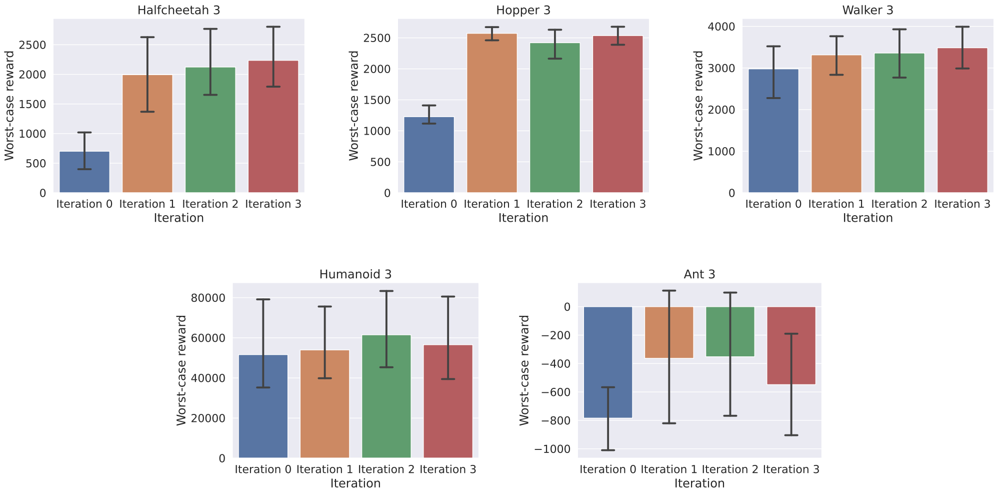


Figure 2: Worst-case performance for each iteration of IWOCS on 3D uncertainty sets

Table 3: Halfcheetah 2, worst parameters search for each iteration over 10 seeds.

	$J^{\pi_{T_0}}_{T_0}$	ψ_1^1	ψ_2^1	$J^{\pi_0}_{T_1}$	$J^{\pi_{T_1}}_{T_1}$	ψ_1^2	ψ_2^2	$J^{\pi_1}_{T_2}$	$J^{\pi_{T_2}}_{T_2}$	ψ_1^3	ψ_2^3	$J^{\pi_2}_{T_3}$
0	5753.2	4.0	7.0	2128.3	20559.5	4.0	0.1	3626.3	9724.7	4.0	7.0	3745.6
1	109.755	4.0	7.0	2027.0	3732.0	4.0	7.0	3827.6	6183.4	4.0	7.0	3827.6
2	5897.4	0.1	7.0	2354.0	9930.0	4.0	7.0	2497.4	3670.5	4.0	7.0	3874.4
3	6396.9	4.0	7.0	2473.0	3815.1	4.0	0.6	3053.9	6022.4	4.0	7.0	3825.5
4	2143.26	0.1	7.0	1633.3	3807.9	4.0	7.0	1978.4	6282.5	4.0	7.0	1978.4
5	4476.0	4.0	7.0	1983.2	13040.2	4.0	0.1	2597.4	3694.9	4.0	0.1	2822.3
6	6275.2	4.0	7.0	-91.1	3639.7	4.0	0.1	3448.4	5347.6	4.0	7.0	3986.0
7	6213.8	0.1	7.0	1587.1	2046.31	4.0	7.0	2411.1	3757.7	4.0	7.0	3769.5
8	6409.8	4.0	7.0	2053.8	3709.9	4.0	0.1	3182.3	3712.3	4.0	7.0	3872.9
9	6801.4	0.1	0.1	2341.4	3620.1	0.5	4.4	3619.3	5737.3	0.6	6.8	5431.3
avg	5047.6			1849.0	6790.1			3024.2	5413.3			3713.4
std	2210.6			740.5	5940.3			623.4	1886.8			876.3

to iteration 1, then 2, worst-case MDPs tend to concentrate on the same corner (the true worst-case MDP) and robust scores tend to increase (as Figure 2 showed in the 3D case).

6 Conclusion

The search for robust policies in uncertain MDPs is a long-standing challenge. In this work, we proposed to revisit the static model of transition function uncertainty, which is equivalent to the dynamic model in the case of sa-rectangular uncertainty sets and stationary policies. We proposed to exploit this equivalence and the no-duality-gap property to design an algorithm that trades the resolution of a two-player game, for a sequence of one-player MDP resolutions. This led to the IWOCS (Incremental Worst-Case Search) meta-algorithm, which incrementally builds a discrete, non-sa-rectangular uncertainty set and a sequence of candidate robust policies. An instance of IWOCS, using SAC for policy optimization, and CMA-ES for worst-case search, appeared as a relevant method on popular robust RL benchmarks, and outperformed the state-of-the-art algorithms on a number of environments. IWOCS proposes a new perspective on the resolution of robust MDPs and robust RL problems, which appears as a competitive formulation with respect to traditional methods. It also poses new questions, like the tradeoffs between policy optimization precision and overall robustness, gradient-based methods for worst-case search, bounds due to approximate value functions, or validity of using Q_i as a surrogate of the robust value function for the $\mathcal{T}_i$ uncertainty set. All these open interesting avenues for future research.

References

Garud N Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2): 257–280, 2005.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Jay K Satia and Roy E Lave Jr. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):728–740, 1973.

Shiau Hong Lim, Huan Xu, and Shie Mannor. Reinforcement learning in robust markov decision processes. *Advances in Neural Information Processing Systems*, 26, 2013.

Huan Xu and Shie Mannor. Distributionally robust markov decision processes. *Advances in Neural Information Processing Systems*, 23, 2010.

Shie Mannor, Ofir Mebel, and Huan Xu. Lightning does not strike twice: robust mdps with coupled uncertainty. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 451–458, 2012.

Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Erwan Lecarpentier and Emmanuel Rachelson. Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Vineet Goyal and Julien Grand-Clement. Robust markov decision processes: Beyond rectangularity. *Mathematics of Operations Research*, 2022.

Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. Fast bellman updates for robust mdps. In *International Conference on Machine Learning*, pages 1979–1988. PMLR, 2018.

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.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

Parameswaran Kamalaruban, Yu ting Huang, Ya-Ping Hsieh, Paul Rolland, C. Shi, and V. Cevher. Robust reinforcement learning via adversarial training with langevin dynamics. *Neural Information Processing Systems*, 2020.

Eugene Vinitsky, Yuqing Du, Kanaad Parvate, Kathy Jang, Pieter Abbeel, and Alexandre Bayen. Robust reinforcement learning using adversarial populations. *arXiv preprint arXiv:2008.01825*, 2020.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21024–21037, 2020.

Huan Zhang, Hongge Chen, Duane S Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*, 2021.

Samuel Stanton, Rasool Fakoor, Jonas Mueller, Andrew Gordon Wilson, and Alex Smola. Robust reinforcement learning for shifting dynamics during deployment. In *Workshop on Safe and Robust Control of Uncertain Systems at NeurIPS*, 2021.

Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4213–4220, 2019.

Takumi Tanabe, Rei Sato, Kazuto Fukuchi, Jun Sakuma, and Youhei Akimoto. Max-min off-policy actor-critic method focusing on worst-case robustness to model misspecification. In *Advances in Neural Information Processing Systems*, 2022.

Esther Derman, Matthieu Geist, and Shie Mannor. Twice regularized mdps and the equivalence between robustness and regularization. *Advances in Neural Information Processing Systems*, 34, 2021.

Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. In *International Conference on Learning Representations*, 2022.

Navdeep Kumar, Esther Derman, Matthieu Geist, Kfir Levy, and Shie Mannor. Policy gradient for s-rectangular robust markov decision processes. *arXiv preprint arXiv:2301.13589*, 2023.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J. Pal, and Liam Paull. Active domain randomization. In *Proceedings of the Conference on Robot Learning*, volume 100, pages 1162–1176, 2020.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *arXiv preprint arXiv: Arxiv-1910.07113*, 2019.

Zichuan Lin, Garrett Thomas, Guangwen Yang, and Tengyu Ma. Model-based adversarial meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 10161–10173, 2020.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, A. Bayen, Stuart J. Russell, Andrew Critch, and S. Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Neural Information Processing Systems*, 2020.

Wenhao Yu, C. K. Liu, and Greg Turk. Policy transfer with strategy optimization. *International Conference On Learning Representations*, 2018.

Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315, 2022.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR, 2019.

Julien Grand-Clément and Marek Petrik. On the convex formulations of robust markov decision processes. *arXiv preprint arXiv:2209.10187*, 2022.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv: Arxiv-1812.05905*, 2018.

Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variance networks: When expectation does not meet your expectations. In *International Conference on Learning Representations*, 2019.

B Lakshminarayanan, A Pritzel, and C Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. arxiv. *arXiv preprint arXiv:1612.01474*, 2016.

Louis Béthune, Paul Novello, Thibaut Boissin, Guillaume Coiffier, Mathieu Serrurier, Quentin Vincenot, and Andres Troya-Galvis. Robust one-class classification with signed distance function using 1-lipschitz neural networks. *arXiv preprint arXiv:2303.01978*, 2023.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Appendix

Table of Contents

A Computing resources

All experiments were run on a desktop machine (Intel i9, 10th generation processor, 64GB RAM) with a single NVIDIA RTX 3090 GPU. Averages, medians, and standard deviations were computed from 10 independent repetitions of each experiment.

B Windy-walk gridworld

The windy-walk environment used in Section 4 is a discrete grid-world environment illustrated in Figure 3. It features 36 discrete states corresponding to positions on the grid, and 4 discrete actions corresponding to cardinal moves. Six states are unreachable and correspond to walls, defining three corridors. The transition model is deterministic by default, except in the corridors where the wind blows. This transition model is parameterized by a scalar parameters α. In the Northern corridor:

- the W action moves West with probability 1,
- the N and S actions leave the position unchanged with probability $1 - \alpha$ and the agent is pushed West with probability α,
- the E action moves East with probability $1 - \alpha$ and West with probability α.

The middle corridor works the same way, but with probability α^3 instead of α. In the Southern corridor:

- the W action moves West with probability 1,

- the N (resp. S) action move the agent respectively North (resp. South) with probability $1 - \alpha^6$ (unless it runs into a wall in which case the position is unchanged), and West with probability α^6,
- the E action moves East with probability $1 - \alpha^6$ and West with probability α^6.

Rewards are -1 for all transitions and the G state is an absorbing goal states yielding zero reward. The agent always starts in state S. Consequently, windy-walk is a stochastic shortest path. For small values of α, the optimal policy is to go straight from S to G, but as α increases, the wind blows harder, and it becomes more interesting to make a detour through the middle then Southern corridors. The corresponding robust MDP problem features an uncertainty set spanned by 25 discrete values of α, uniformly distributed in $[0, 0.5]$.

The instance of IWOCS evaluated in Section 4 uses value iteration as a policy optimization algorithm and a brute-force grid search as a search method for worst-case transition functions, as summarized in Algorithm 2. In Algorithm 2 we abusively write T_α the transition model parameterized by α.

In the experiments of Section 4, value iteration is run until a tolerance of 10^{-3} is met. γ is set to 0.95. Monte-Carlo estimates of $V_T^{\pi_i}(s_0)$ use 300 independent rollouts (of length at most 10^4) from the starting state.

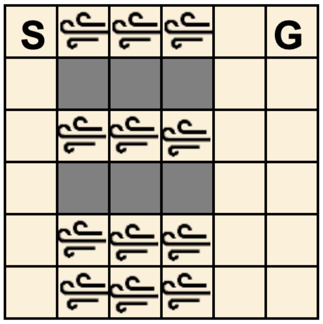


Figure 3: Windy walk grid-world.

Algorithm 2: IWOCS with value iteration and brute force worst-case search

```
Input: $\mathcal{T}$, $T_0$, max number of iterations $M$, tolerance on robust value $\epsilon$
for $i = 0$ to $M$ do
    Find $Q^*_{T_i}$ = value_iteration($T_i$)        /* Non-robust policy optimization */
    Define $\mathcal{T}_i = \{T_j\}_{j \in [0,i]}$
    Define $Q_i : s, a \mapsto \min_{j \in [0,i]} \{Q^*_{T_j}(s,a)\}$        /* $\mathcal{T}_i$-robust value function */
    Define $\pi_i(s) = \arg\max_a (Q_i(s,a))$        /* Candidate policy */
    $V^{\pi_i}_{T_{i+1}}(s_0) = +\infty$
    for $\alpha \in \mathcal{T}$ do        /* Identify worst $T$ */
        $\tilde{V}$ = Monte-Carlo_rollouts($\pi_i$)
        if $\tilde{V} < V^{\pi_i}_{T_{i+1}}(s_0)$ then
            $V^{\pi_i}_{T_{i+1}}(s_0) = \tilde{V}$
            $T_{i+1} = T_\alpha$
    if $|V^{\pi_i}_{T_{i+1}}(s_0) - Q_i(s_0, \pi_i(s_0))| \leqslant \epsilon$ then
        return $\pi_i$, $T_{i+1}$, $V^{\pi_i}_{T_{i+1}}(s_0)$        /* Early termination condition */
return $\pi_M$, $T_{M+1}$, $V^{\pi_M}_{T_{M+1}}(s_0)$
```

C Soft Actor-Critic and CMA-ES hyperparameters

Deep IWOCS uses SAC [Haarnoja et al., 2018] for policy optimization and trains jointly a predictive coding model to predict the outcome of a state-action pair. Specifically, a single network called "enhanced critic" is trained to predict the regularized value function $Q'(s,a)$, the unregularized value function $Q(s,a)$ and a prediction of the transition outcome $\hat{T}(s,a)$. The network's architecture is summarized in Figure 4a. All activation functions are ReLU except for the output layers (identity functions). Note that one more layer was necessary to appropriately estimate Q compared to Q'. Our implementation also uses double critics as per the common practice, to avoid overestimating Q and Q' (totally independent networks, no shared layers). Given a replay buffer $\mathcal{D}$, learning Q minimizes the loss

$$L_Q = \mathbb{E}_{s,a,s'\sim\mathcal{D},a'\sim\pi}\left[Q(s,a) - [r + \gamma Q^{-}(s',a')]\right]^2,$$

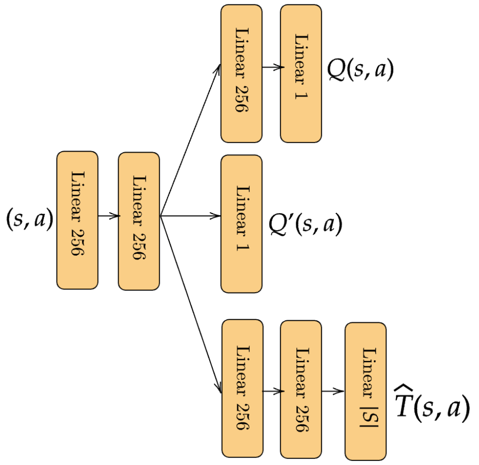
where Q^{-} is a target network, updated through Polyak averaging. Similarly, Q' minimizes

$$L'_Q = \mathbb{E}_{s,a,s'\sim\mathcal{D},a'\sim\pi}\left[Q'(s,a) - [r + \gamma(Q'^{-}(s',a') - \alpha \log \pi(a'|s'))]\right]^2.$$

Finally, $\hat{T}$ minimizes

$$L_{\hat{T}} = \mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\|\hat{T}(s,a) - s'\|_1\right].$$

These three objective functions are minimized in turn with three distinct Adam optimizers to account for possible different orders of magnitude.



(a) Enhanced critic

(b) Actor

Figure 4: Network architectures

The actor network is a standard SAC actor trained with respect to the regularized Q-function Q'. The network's architecture is depicted in Figure 4b. All activation functions are ReLU, except for the output values (identify for μ and $\tanh$ for $\log\sigma$ as per the common practice). The output action drawn from the network's output is run through an additional $\tanh$ function following the usual SAC implementations.

The search for worst case transition functions is performed by using the CMA-ES black-box optimization method [Hansen and Ostermeier, 2001]. All hyperparameter values for SAC and CMA-ES are summarized in Table 4. These values are the same across all experiments.

D Adaptive thresholding for predictive coding

As introduced in Section 5, the SAC-based implementation of IWOCS used in the experiments exploits a predictive coding mechanism in order to characterize each policy's training distribution support and to avoid using a given π_{T_i} on samples outside its training distribution. Policy π_{T_i} and value function Q_{T_i} are deemed usable in state s_t along the current trajectory if s_t was accurately

Table 4: Hyperparameters of SAC

Hyperparameter	**Value**
Learning rate actor	3e-4
Learning rate critic	1e-3
Adam epsilon	1e-5
Adam (β_1, β_2)	(0.9, 0.999)
Batch size	256
Memory size	1e6
Gamma	0.99
Polyak update	0.995
Number of steps before training	1e4
CMA-ES generations	10
CMAS-ES population size	100
CMA ES mean	0.5
CMA ES std	0.5

predicted by the dynamics model $\hat{T}_i(s_{t-1}, a_{t-1})$. Specifically, we consider a threshold ρ_i on the prediction error and consider Q_{T_i} and π_{T_i} to be viable in s_t if $\ell(\hat{T}_i(s_{t-1}, a_{t-1}), s_t) \leqslant \rho_i$, with $\ell(\hat{T}_i(s_{t-1}, a_{t-1}), s_t) = \|\hat{T}_i(s_{t-1}, a_{t-1}) - s_t\|_1$. In states where Q_{T_i} is non-viable, we arbitrarily set its value to $+\infty$ so that it does not participate in $Q_i(s,a) = \min_{j \leqslant i}\{Q_{T_j}(s,a)\}$. We noted in the main text that alternative characterizations of the support distribution were possible, and we do not claim the present choice outperforms the alternatives. Notably, all choices induce a number of parameters to tune (here ρ_i). This leads to a number of design choices that make the implementation somehow more convoluted than the simple principle of IWOCS. While the main text kept things focused on the principles of IWOCS, we provide here a full pseudo-code (which is more representative of the provided code) for the sake of completeness. Appendix C already covered the network structure, the training losses and the training hyperparameters for SAC and CMA-ES. Hence, the present section focuses on how to adjust each ρ_i.

Training of the enhanced critic network does not provide a usable value for ρ_i and experimental results demonstrated that accurate characterization of the training distribution's support required per-MDP tuning. Since ρ_i needs to be tuned for π_{T_i} during iteration i (and is kept fixed thereafter), we couple its search with that of the worst case transition model to permit better overall efficiency. Specifically, at iteration i, we consider a discrete set R of possible values for threshold ρ_i. For each value in R, we identify the worst case transition model. Then, we keep the value of ρ_i that enabled the best pessimistic value. In a sense, this makes ρ_i a parameter of the candidate robust policy. This parameter is single-dimensional and hence its optimization is computationally undemanding. We emphasize that this tuning mechanism is both very naive and arbitrary. It is naive since it performs a grid search over discrete values of ρ_i, where it could have exploited optimization methods. It is arbitrary in the sense that it picks ρ_i by keeping as a selection heuristic the overall goal of identifying robust policies.

Algorithm 3 summarizes the complete IWOCS process with adaptive thresholding for predictive coding. In the experiments of Section 5, R is a discrete set of 10 values evenly spaced between 0.1 and 1.

E Sample budgets

In order to enable a fair comparison with the results of Tanabe et al. [2022] which we report in Table 1, we evaluate IWOCS with the same overall sample budget, ie. 4 million samples in 2D uncertainty set environments and 5 million samples in 3D uncertainty set environments.

In 2D environments, the default DR policy is trained for $1.6 \cdot 10^6$ steps, then 3 IWOCS iterations of $8 \cdot 10^5$ each are run, for a total of $4 \cdot 10^6$ collected samples.

In 3D environments, the default DR policy is trained for $1.8 \cdot 10^6$ steps, then 4 IWOCS iterations of $8 \cdot 10^5$ each are run, for a total of $5 \cdot 10^6$ collected samples.

Algorithm 3: Deep IWOCS with adaptive threshold

```
Input: $\mathcal{T}$, $T_0$, maximum number of iterations $M$, discrete thresholds set $R$
for $i = 0$ to $M$ do
    Find $Q_{T_i}, \pi_{T_i}, \hat{T}_i = \texttt{SAC}(T_i)$                    // Non-robust SAC and pred coding
    Define $\mathcal{T}_i = \{T_j\}_{j \in [0,i]}$
    Define $\hat{V} = -\infty$                                                  // candidate worst value
    for $\rho \in R$ do                                                         // loop over thresholds
        Define $\tilde{Q}_{T_i}(s,a) = \begin{cases} +\infty \text{ if } \ell(\hat{T}_i(s_{t-1}, a_{t-1}); s_t) > \rho \text{ for } s = s_t \\ Q_{T_i}(s,a) \text{ otherwise.} \end{cases}$
        Define $\tilde{Q}_i(s,a) = \min_{j \leqslant i} \{\tilde{Q}_{T_j}(s,a)\}_{j \in [0,i]}, \forall s, a$   // $\mathcal{T}_i$-robust value function
        Define $J^*(s) = \arg\min_{j \leqslant i} \tilde{Q}_{T_j}(s, \pi_{T_j}(s)), \forall s$
        Define $\tilde{\pi}_i(s) = \begin{cases} \pi_{default}(s) \text{ if } \tilde{Q}_i(s,a) = +\infty, \\ \pi_{T_{j*}}(s) \text{ with } j^* \in J^*(s) \text{ otherwise.} \end{cases}$   // Candidate policy
        Find $\tilde{T}_{i+1}, V^{\tilde{\pi}_i}_{\tilde{T}_{i+1}}(s_0) = \texttt{CMA-ES}(V^{\tilde{\pi}_i}_{T}(s_0))$   // Identify worst $T$
        if $V^{\pi_i}_{T'_{i+1}}(s_0) \geqslant \hat{V}$ then                  // keep best $\rho$
            $\rho_j = \rho$
            $T_{i+1} = \tilde{T}_{i+1}$, $\pi_i = \tilde{\pi}_i$, $V^{\pi_i}_{T_{i+1}}(s_0) = V^{\tilde{\pi}_i}_{\tilde{T}_{i+1}}(s_0)$, $Q_i = \tilde{Q}_i$
    if $|V^{\pi_i}_{T_{i+1}}(s_0) - Q_i(s_0, \pi_i(s_0))| \leqslant \epsilon$ then
        return $\pi_i$, $T_{i+1}$, $V^{\pi_i}_{T_{i+1}}(s_0)$                  // Early termination condition
return $\pi_M$, $T_{M+1}$, $V^{\pi_M}_{T_{M+1}}(s_0)$
```

Environment	SAC	IWOCS
Ant 2	18h	44h
Ant 3	22.5h	76h
Halfcheetah 2	20h	43h
Halfcheetah 3	25h	66h
Walker 2	19h	47h
Walker 3	24h	64h
Hopper 2	20h	44h
Hopper 3	25h	65h
HumanoidStandup 2	18h	49h
HumanoidStandup 3	22.5h	68h

Table 5: Average wall-clock time for plain SAC and for IWOCS for the same number of samples.

No fine-tuning of these training durations was performed.

F Computational overhead due to IWOCS

In Table 5 we report the average wall-clock time needed for our implementation of SAC to cover the 4 (resp. 5) million samples allocated for 2D (resp. 3D) environments without IWOCS. Then, we report the time required by IWOCS to cover the same sample budget. This permits a fair evaluation of the overhead computational cost of IWOCS, without the bias due to implementation optimizations.

G Uncertainty sets in MuJoCo environments

The experiments of Section 5 follow the evaluation protocol proposed by Tanabe et al. [2022] and based on MuJoCo environments [Todorov et al., 2012]. These environments are designed with 2D or 3D uncertainty sets. Table 6 lists all environments evaluated, along with their uncertainty sets. The uncertainty sets column defines the ranges of variation for the parameters within each environment.

The reference parameters column indicates the nominal or default values. The uncertainty parameters column describes the physical meaning of each parameter.

Table 6: List of environment and parameters for the experiements

Environment	Uncertainty set $\mathcal{T}$	Reference values	Uncertainty parameters
Ant 2	$[0.1, 3.0]\times[0.01, 3.0]$	$(0.33, 0.04)$	torso mass; front left leg mass
Ant 3	$[0.1, 3.0] \times [0.01, 3.0] \times [0.01, 3.0]$	$(0.33, 0.04, 0.06)$	torso mass; front left leg mass; front right leg mass
HalfCheetah 2	$[0.1, 4.0] \times [0.1, 7.0]$	$(0.4, 6.36)$	world friction; torso mass
HalfCheetah 3	$[0.1, 4.0] \times [0.1, 7.0] \times [0.1, 3.0]$	$(0.4, 6.36, 1.53)$	world friction; torso mass; back thigh mass
Hopper 2	$[0.1, 3.0] \times [0.1, 3.0]$	$(1.00, 3.53)$	world friction; torso mass
Hopper 3	$[0.1, 3.0] \times [0.1, 3.0] \times [0.1, 4.0]$	$(1.00, 3.53, 3.93)$	world friction; torso mass; thigh mass
HumanoidStandup 2	$[0.1, 16.0]\times[0.1, 8.0]$	$(8.32, 1.77)$	torso mass; right foot mass
HumanoidStandup 3	$[0.1, 16.0] \times [0.1, 5.0] \times [0.1, 8.0]$	$(8.32, 1.77, 4.53)$	torso mass; right foot mass; left thigh mass
InvertedPendulum 2	$[1.0, 31.0] \times [1.0, 11.0]$	$(4.90, 9.42)$	pole mass; cart mass
Walker 2	$[0.1, 4.0] \times [0.1, 5.0]$	$(0.7, 3.53)$	world friction; torso mass
Walker 3	$[0.1, 4.0] \times [0.1, 5.0] \times [0.1, 6.0]$	$(0.7, 3.53, 3.93)$	world friction; torso mass; thigh mass

H Non-normalized results

Table 7 reports the non-normalized worst case scores, averaged across 10 independent runs for each benchmark. Table 8 reports the average score obtained by each agent across a grid of environments, also averaged across 10 independent runs for each benchmark.

I Worst-case paths

Table 3 illustrated the path followed by the successive identified worst-case transition functions T_i in the 2D uncertainty set of the HalfCheetah 2 environment, across 10 independent optimization runs. For the sake of completeness, we provide here the same results for all environments, which permit drawing similar conclusions. Tables 9 and 10 start by recalling the physical meaning of each transition function's parameters. Then, Tables 11 to 21 follow the same logic as Table 3 and report the evolution of worst-case parameters and values on all other environments than HalfCheetah 2.

J Median scores

Table 1 reported the worst case score obtained by all algorithms, averaged across 10 independent runs. Table 22 reports the median rather than the average across these runs.

Similarly, Table 2 reported the average score across a span of environments, averaged across 10 independent runs. Table 23 reports also the median across these runs rather than the average.

Unfortunately it was not possible to relate these figures to competitor methods as only averages are reported in the work of Tanabe et al. [2022].

Table 7: Avg. ± std. error of worst-case performance over 10 seeds for each method

Environment	IWOCS	M2TD3	SoftM2TD3	M2-DDPG	M3DDPG	RARL (DDPG)	DR (TD3)	TD3 (ref)
Ant 2 $(\times 10^3)$	1.1 ± 2.2	**4.13**± 0.11	3.92± 0.14	0.95± 0.20	−0.25± 0.13	−1.77± 0.09	1.64± 0.13	1.59± 0.08
Ant 3 $(\times 10^3)$	−0.5± 0.64	**0.10**± 0.10	0.07± 0.20	−1.13± 0.28	−1.38± 0.22	−2.38± 0.07	−0.32± 0.03	−0.99± 1.13
HalfCheetah 2 $(\times 10^3)$	**3.71**± 0.87	2.61± 0.16	2.82± 0.16	2.54± 0.23	−0.58± 0.06	−0.70± 0.05	2.12± 0.13	−0.53± 0.06
HalfCheetah 3 $(\times 10^3)$	**2.23**± 0.87	0.93± 0.21	1.53± 0.23	1.20± 0.22	−0.66± 0.08	−0.81± 0.07	1.09± 0.06	−0.61± 0.08
Hopper 2 $(\times 10^2)$	**28.7**± 3.78	5.33± 0.28	5.79± 0.29	4.30± 0.57	2.58± 0.29	3.34± 0.89	4.68± 0.15	0.21± 0.04
Hopper 3 $(\times 10^2)$	**25.3**± 2.67	2.84± 0.25	1.98± 0.22	2.25± 0.29	0.73± 0.11	1.64± 0.46	2.10± 0.35	0.14± 0.03
HumanoidStandup 2 $(\times 10^4)$	5.46± 1.06	6.50± 0.70	**7.94** ± 0.90	6.24± 0.54	6.37± 0.72	5.78± 0.73	7.31± 0.78	0.73± 0.07
HumanoidStandup 3 $(\times 10^4)$	5.63± 3.55	**6.20**± 0.64	5.99± 0.37	5.96± 0.58	6.01± 0.38	5.54± 0.76	5.41± 0.34	0.57± 0.04
InvertedPendulum 2 $(\times 10^2)$	**10**±0	3.56± 1.32	1.36± 0.30	1.10± 0.62	0.02± 0.00	0.02± 0.00	0.57± 0.02	0.03± 0.00
Walker 2 $(\times 10^3)$	**3.91**± 0.97	3.14± 0.39	2.64± 0.43	0.85± 0.12	0.39± 0.11	0.06± 0.04	2.31± 0.50	0.28± 0.07
Walker 3 $(\times 10^3)$	**3.48**± 0.88	1.94± 0.40	2.00± 0.35	0.82± 0.13	0.28± 0.09	0.00± 0.02	1.32± 0.34	0.17+ 0.06

Table 8: Avg. ± std. deviation of average performance over 10 seeds for each method

Environment	IWOCS	M2TD3	SoftM2TD3	M3DDPG	RARL	DR (TD3)	TD3 (ref)
Ant 2 $(\times 10^3)$	3.69 ± 1.60	5.44 ± 0.05	5.56 ± 0.01	1.86 ± 0.38	−1.00± 0.06	**6.32** ± **0.09**	2.28 ± 0.09
Ant 3 $(\times 10^3)$	1.38 ± 1.48	2.66 ± 0.22	2.98 ± 0.23	−0.33± 0.22	−1.31± 0.09	**3.62** ± **0.11**	3.16 ± 1.00
HalfCheetah 2 $(\times 10^3)$	5.63 ± 0.44	4.35 ± 0.05	4.52 ± 0.07	0.77 ± 0.12	−0.70± 0.05	**5.79** ± 0.15	2.63 ± 0.20
HalfCheetah 3 $(\times 10^3)$	4.98 ± 0.42	3.79 ± 0.09	4.02 ± 0.04	0.58 ± 0.18	−0.81± 0.07	**5.54** ± 0.16	2.47 ± 0.18
Hopper 2 $(\times 10^3)$	**3.38** ± 0.78	2.51 ± 0.07	2.26 ± 0.12	1.15 ± 0.11	3.34 ± 0.89	1.89 ± 0.08	1.54 ± 0.17
Hopper 3 $(\times 10^3)$	**3.0** ± 0.99	0.85 ± 0.07	0.79 ± 0.04	0.49 ± 0.11	1.64 ± 0.46	1.50 ± 0.07	1.15 ± 0.14
HumanoidStandup 2 $(\times 10^5)$	**1.21** ± 0.18	0.97 ± 0.04	1.07 ± 0.05	0.92 ± 0.04	0.85 ± 0.08	1.06 ± 0.04	1.03 ± 0.03
HumanoidStandup 3 $(\times 10^5)$	**1.16** ± 0.33	1.09 ± 0.06	1.04 ± 0.03	1.01 ± 0.04	0.87 ± 0.07	1.04 ± 0.07	1.01 ± 0.03
InvertedPendulum 2 $(\times 10^2)$	**10** ± 0	6.13 ± 1.42	6.26 ± 0.95	1.76 ± 0.51	3.07 ± 0.66	9.18 ± 0.07	4.05 ± 0.52
Walker 2 $(\times 10^3)$	**4.78** ± 0.90	4.72 ± 0.12	4.37 ± 0.32	1.63 ± 0.22	0.26 ± 0.05	4.54 ± 0.31	2.70 ± 0.20
Walker 3 $(\times 10^3)$	**4.58** ± 0.83	4.27 ± 0.21	4.21 ± 0.30	1.65 ± 0.15	0.21 ± 0.07	4.48 ± 0.16	2.60 ± 0.18

Table 9: Physical meaning of transition function parameters in 2D environments

Environment	ψ_1	ψ_2
Ant 2	torso mass	front left leg mass
Halfcheetah 2	world friction	torso mass
Hopper 2	world friction	torso mass
HumanoidStandup 2	torso mass	right thigh mass
Walker 2	world friction	torso mass
InvertedPendulum 2	pole mass	cart mass

Table 10: Physical meaning of transition function parameters in 3D environments

Environment	ψ_1	ψ_2	ψ_3
Ant 3	torso mass	front left leg mass	front right leg mass
Halfcheetah 3	world friction	torso mass	back thigh mass
Hopper 3	world friction	torso mass	thigh mass
HumanoidStandup 3	torso mass	left thigh mass	right foot mass
Walker 3	world friction	torso mass	thigh mass

K How many valid policies in s ?

The Deep-IWOCS method proposed in Section 5 introduced indicator functions constraining the use of a given policy to a subset of states. Depending on the environment and uncertainty parameters, we expect some policies to remain within the same set of explored states, while others will cover a very different state distribution. To quantify this aspect, we ran an experiment where the IWOCS final policy is run on each benchmark, across a grid of transition functions. For each encountered state, we count how many policies are valid. Figure 5 reports the corresponding histograms (note the log-scale on the y-axis).

Obviously, the number of valid policies is capped by the number of iterations performed by IWOCS (3 in 2D environments, 4 in 3D ones). Depending on the environment, we observe a wide variety of state coverages. For instance, optimizing policies for different transition functions of HalfCheetah induce very different state distributions and very few states feature more than one valid policy. Conversely, in Ant or Hopper environments, there seems to be a larger number of states that belong to the training distribution of several policies.

L Iwocs using grid search

In the Deep IWOCS approach, CMA-ES is employed to conduct the worst-case search, described by $T_{i+1} = \arg\min_{T\in\mathcal{T}} V_T^{pi_i}(s_0)$. However, a grid search can also serve as an alternative method. We've designated this variant as IWOCS*. For our worst-case search in IWOCS*, we adopted the configuration suggested by [Tanabe et al., 2022]. Each dimension of the uncertainty space is divided into 10 segments. This results in a total of 1000 grid points for 3 dimensions and 100 grid points for 2 dimensions.Due to the combinatorial nature of the search space, we refrained from conducting 3D experiments, as they proved computationally prohibitive in terms of finding the worst-case scenario. For SAC, we adhered to the hyperparameters listed in Table 4.

M Carbon Impact of Our Experiments

In recent years, the environmental impact of large-scale experiments has become a topic of increasing importance. Here, we detail the carbon footprint resulting from the computational experiments conducted for this paper.

To put things into perspective, the total computational time dedicated to our experiments amounted to 24,585 hours. Each hour of operation for our machine consumed 1 kW. Given that we are located in

Table 11: Walker 3 worst parameters for each iteration over 10 seeds

	$J^{\pi_0}_{T_0}$	ψ^1_1	ψ^1_2	ψ^1_3	$J^{\pi_0}_{T_1}$	$J^{\pi_1}_{T_1}$	ψ^2_1	ψ^2_2	ψ^2_3	$J^{\pi_1}_{T_2}$	$J^{\pi_2}_{T_2}$	ψ^3_1	ψ^3_2	ψ^3_3	$J^{\pi_2}_{T_3}$	$J^{\pi_3}_{T_3}$	ψ^4_1	ψ^4_2	ψ^4_3	$J^{\pi_3}_{T_4}$
0	5773.2	4.0	5.0	6.0	3159.6	5698.0	3.0	0.3	0.3	3339.8	4142.3	4.0	0.3	0.2	3426.7	174.3	4.0	2.9	6.0	3186.7
1	885.1	4.0	0.1	0.1	3469.4	4977.1	4.0	5.0	6.0	1833.3	782.7	0.3	5.0	0.1	2327.9	3851.3	4.0	5.0	5.8	2847.7
2	5194.5	4.0	5.0	6.0	3671.0	3233.1	4.0	5.0	6.0	3488.4	3875.6	4.0	5.0	0.1	4206.2	4891.6	4.0	1.6	6.0	3515.8
3	5743.1	4.0	5.0	6.0	2149.0	3688.3	4.0	5.0	6.0	2763.6	2997.6	4.0	5.0	4.3	2453.2	4261.8	4.0	5.0	6.0	2525.5
4	3967.8	4.0	0.1	0.1	2644.8	4760.4	4.0	0.1	0.1	2997.1	998.9	4.0	5.0	6.0	2674.0	3767.1	4.0	5.0	6.0	2884.0
5	4934.1	4.0	5.0	6.0	2843.1	6709.4	4.0	0.1	0.1	3108.7	4610.2	4.0	5.0	6.0	2048.6	3287.5	4.0	2.2	6.0	2333.4
6	5101.6	1.5	5.0	6.0	470.5	651.4	4.0	5.0	6.0	2910.4	2932.1	0.1	0.1	0.1	3183.0	4042.3	2.4	5.0	6.0	4286.7
7	4218.4	4.0	0.6	6.0	3862.5	3392.6	4.0	0.1	6.0	3885.9	3926.7	4.0	0.1	6.0	3885.9	5802.3	4.0	0.1	6.0	3885.9
8	5418.9	1.9	5.0	6.0	3765.5	4164.1	4.0	5.0	6.0	4825.9	5430.6	4.0	5.0	6.0	4954.7	4424.0	4.0	5.0	6.0	4954.8
9	5121.6	1.4	0.1	0.1	3752.2	4333.7	0.3	1.7	2.4	4000.5	69.7	0.1	0.1	0.1	4435.4	3814.6	0.1	0.1	0.1	4435.4
mean	4635.8				2978.8	4160.8				3315.4	2976.6				3359.5	3831.7				3485.6
std	1440.5				1046.7	1629.7				810.8	1793.1				989.1	1463.0				880.1

Table 12: Walker 2 worst parameters for each iteration over 10 seeds

	$J^{\pi_0}_{T_0}$	ψ^1_1	ψ^1_2	$J^{\pi_0}_{T_1}$	$J^{\pi_1}_{T_1}$	ψ^2_1	ψ^2_2	$J^{\pi_1}_{T_2}$	$J^{\pi_2}_{T_2}$	ψ^3_1	ψ^3_2	$J^{\pi_2}_{T_3}$
0	4265.4	4.0	5.0	4108.0	4256.9	4.0	5.0	4134.6	5258.1	4.0	5.0	4094.9
1	3997.2	4.0	5.0	4353.9	5036.0	4.0	1.9	4515.3	2955.4	4.0	5.0	4422.6
2	5801.6	4.0	0.1	3428.5	201.6	4.0	5.0	3505.1	652.8	4.0	5.0	4320.2
3	5314.0	4.0	5.0	5181.1	3173.4	4.0	3.5	5250.7	3145.4	4.0	3.8	5161.9
4	5912.1	4.0	0.1	3524.2	5382.1	4.0	0.1	3360.3	3120.6	4.0	0.1	3360.3
5	567.9	4.0	5.0	1663.0	37.9	4.0	5.0	4543.1	4932.0	4.0	5.0	4306.8
6	5677.3	4.0	4.5	3481.8	4431.5	4.0	5.0	4243.9	4629.9	4.0	5.0	3710.1
7	4739.0	4.0	0.1	4693.6	2332.3	4.0	0.1	4595.2	2862.6	4.0	0.1	4870.1
8	5370.4	4.0	5.0	3316.4	4710.1	4.0	0.1	3297.3	4265.2	4.0	5.0	3164.1
9	157.6	4.0	5.0	1879.1	5101.4	4.0	0.1	3155.3	3163.5	4.0	5.0	1776.1
mean	4180.2			3562.9	3466.3			4060.1	3498.5			3918.7
std	2112.3			1123.1	1991.5			697.8	1341.5			975.4

France, the carbon emission for electricity is notably low due to the heavy reliance on nuclear power. Specifically, for every kW consumed per hour, 60 grams of CO is emitted.

To compute the total carbon emission for our work, we use the following formula:

$$\text{Total Carbon Emission (kg)} = \frac{\text{Total Hours} \times \text{Power per Hour (kW)} \times \text{Carbon per kW-hour (g)}}{1000}$$

Plugging in our values:

$$\text{Total Carbon Emission (kg)} = \frac{24585 \times 1 \times 60}{1000} = 1475.1 \text{ kg}$$

Hence, the carbon footprint for the entirety of our experiments is 1,475.1 kg of CO. It underscores the need for efficient algorithms and sustainable energy sources in reinforcement learning research.

Table 13: InvertedPendulum 2 worst parameters for each iteration over 10 seeds

	$J_{T_0}^{\pi_0}$	ψ_1^1	ψ_2^1	$J_{T_1}^{\pi_0}$	$J_{T_1}^{\pi_1}$	ψ_1^2	ψ_2^2	$J_{T_2}^{\pi_1}$	$J_{T_2}^{\pi_2}$	ψ_1^3	ψ_2^3	$J_{T_3}^{\pi_2}$
0	1000	16.8	8.5	1000.0	1000	16.8	8.5	1000.0	1000	16.8	8.5	1000.0
1	1000	11.8	4.2	1000.0	1000	11.8	4.2	1000.0	1000	11.8	4.2	1000.0
2	1000	25.3	2.4	1000.0	1000	25.3	2.4	1000.0	1000	25.3	2.4	1000.0
3	1000	16.0	4.5	1000.0	1000	16.0	4.5	1000.0	1000	16.0	4.5	1000.0
4	1000	27.9	10.9	1000.0	1000	27.9	10.9	1000.0	1000	27.9	10.9	1000.0
5	402	16.5	8.0	1000.0	1000	16.5	8.0	1000.0	1000	16.5	8.0	1000.0
6	1000	11.3	9.7	1000.0	1000	11.3	9.7	1000.0	1000	11.3	9.7	1000.0
7	1000	22.6	4.3	1000.0	1000	22.6	4.3	1000.0	1000	22.6	4.3	1000.0
8	1000	9.8	5.7	1000.0	1000	9.8	5.7	1000.0	274	9.8	5.7	1000.0
9	1000	20.8	4.8	1000.0	1000	20.8	4.8	1000.0	1000	20.8	4.8	1000.0
avg	940.2			1000	1000.000			1000.000	927.400			1000.000
std	189.1			0	0			0	229.5			0

Table 14: Halfcheetah 3 worst parameters for each iteration over 10 seeds

	$J_{T_0}^{\pi_0}$	ψ_1^1	ψ_2^1	ψ_3^1	$J_{T_1}^{\pi_0}$	$J_{T_1}^{\pi_1}$	ψ_1^2	ψ_2^2	ψ_3^2	$J_{T_2}^{\pi_1}$	$J_{T_2}^{\pi_2}$	ψ_1^3	ψ_2^3	ψ_3^3	$J_{T_3}^{\pi_2}$	$J_{T_3}^{\pi_3}$	ψ_1^4	ψ_2^4	ψ_3^4	$J_{T_4}^{\pi_3}$
0	2861.4	4.0	0.1	0.1	572.7	4226.3	4.0	0.1	0.1	1329.2	3577.5	4.0	7.0	3.0	1221.3	4490.9	4.0	7.0	0.1	1781.1
1	6543.0	4.0	7.0	0.1	650.8	3645.6	4.0	7.0	3.0	1932.2	4250.6	0.1	0.1	3.0	1835.7	5277.7	4.0	7.0	3.0	1863.6
2	6227.7	0.1	0.1	3.0	338.1	4273.2	4.0	7.0	3.0	1116.4	3593.7	4.0	7.0	0.2	1133.7	3549.4	0.1	0.1	0.1	1659.6
3	4943.3	4.0	7.0	3.0	824.7	10990.4	4.0	0.1	3.0	2016.6	11954.7	4.0	1.2	3.0	2380.5	15077.8	4.0	0.1	3.0	2402.4
4	5377.0	4.0	0.1	0.4	1019.4	15130.2	0.1	7.0	3.0	85.4	12553.3	4.0	7.0	3.0	1414.6	10673.7	4.0	0.1	0.1	1016.5
5	4849.7	0.1	0.1	3.0	812.0	6698.7	0.3	0.2	0.3	4363.3	7513.8	0.3	0.2	0.3	4363.3	3612.6	0.3	0.2	0.3	4363.3
6	6547.4	0.1	7.0	3.0	-109.3	3572.7	0.3	0.3	0.5	2414.9	5265.6	0.3	0.3	0.5	2414.9	4027.0	0.3	0.3	0.5	2414.9
7	3896.5	4.0	7.0	0.1	158.4	18261.8	4.0	7.0	3.0	2315.3	3598.3	4.0	7.0	3.0	1996.3	4339.0	4.0	2.3	3.0	2307.8
8	6481.7	4.0	7.0	3.0	1077.4	4013.2	4.0	7.0	0.1	2415.3	3697.0	4.0	0.1	3.0	2428.4	17035.0	4.0	0.1	3.0	2503.1
9	5853.4	4.0	7.0	0.2	1676.7	6602.8	4.0	7.0	3.0	1955.0	6720.0	4.0	0.9	3.0	2047.2	3677.4	4.0	0.9	3.0	2047.2
mean	5358.1				702.1	7741.5				1994.4	6272.4				2123.6	7176.0				2235.9
std	1242.0				508.2	5277.3				1102.1	3444.8				922.7	5150.4				873.2

Table 15: Halfcheetah 2 worst parameters for each iteration over 10 seeds

	$J_{T_0}^{\pi_0}$	ψ_1^1	ψ_2^1	$J_{T_1}^{\pi_0}$	$J_{T_1}^{\pi_1}$	ψ_1^2	ψ_2^2	$J_{T_2}^{\pi_1}$	$J_{T_2}^{\pi_2}$	ψ_1^3	ψ_2^3	$J_{T_3}^{\pi_2}$
0	5753.2	4.0	7.0	2128.3	20559.6	4.0	0.1	3626.3	9724.8	4.0	7.0	3745.6
1	109.8	4.0	7.0	2027.0	3732.0	4.0	7.0	3827.6	6183.5	4.0	7.0	3827.6
2	5897.5	0.1	7.0	2354.0	9930.0	4.0	7.0	2497.4	3670.6	4.0	7.0	3874.4
3	6396.9	4.0	7.0	2473.0	3815.1	4.0	0.6	3053.9	6022.4	4.0	7.0	3825.5
4	2143.3	0.1	7.0	1633.3	3807.9	4.0	7.0	1978.4	6282.5	4.0	7.0	1978.4
5	4476.0	4.0	7.0	1983.2	13040.2	4.0	0.1	2597.4	3694.9	4.0	0.1	2822.3
6	6275.2	4.0	7.0	-91.1	3639.7	4.0	0.1	3448.4	5347.6	4.0	7.0	3986.0
7	6213.8	0.1	7.0	1587.1	2046.3	4.0	7.0	2411.1	3757.7	4.0	7.0	3769.5
8	6409.9	4.0	7.0	2053.8	3709.9	4.0	0.1	3182.3	3712.3	4.0	7.0	3872.9
9	6801.4	0.1	0.1	2341.4	3620.1	0.5	4.4	3619.3	5737.4	0.6	6.8	5431.3
mean	5047.7			1849.0	6790.1			3024.2	5413.4			3713.3
std	2210.7			740.6	5940.3			623.4	1886.9			876.3

Table 16: Hopper 3 worst parameters for each iteration over 10 seeds

	$J_{T_0}^{\pi_0}$	ψ_1^1	ψ_2^1	ψ_3^1	$J_{T_1}^{\pi_0}$	$J_{T_1}^{\pi_1}$	ψ_1^2	ψ_2^2	ψ_3^2	$J_{T_2}^{\pi_1}$	$J_{T_2}^{\pi_2}$	ψ_1^3	ψ_2^3	ψ_3^3	$J_{T_3}^{\pi_2}$	$J_{T_3}^{\pi_3}$	ψ_1^4	ψ_2^4	ψ_3^4	$J_{T_4}^{\pi_3}$
0	2959.1	3.0	0.1	0.1	1167.7	1016.4	3.0	3.0	4.0	2618.9	1479.5	3.0	3.0	4.0	2660.0	3605.8	2.8	1.8	2.4	2731.0
1	864.7	3.0	0.1	0.1	1224.4	1333.9	2.8	0.5	0.7	2763.3	1720.0	2.1	0.6	0.8	2801.2	3247.3	0.8	0.6	0.9	2747.6
2	716.7	3.0	0.1	0.1	1068.8	3381.1	3.0	3.0	4.0	2572.9	3702.9	3.0	3.0	4.0	2572.9	1197.7	3.0	3.0	4.0	2572.9
3	1222.3	3.0	0.1	0.1	1158.5	417.2	3.0	3.0	4.0	2272.9	4193.7	0.4	0.6	0.6	2657.4	3189.8	0.4	0.1	0.4	2142.3
4	2533.3	3.0	0.1	0.1	1164.8	3033.3	2.1	0.6	0.5	2696.7	2586.3	1.7	2.0	0.9	2682.9	2808.7	3.0	0.1	0.1	2223.6
5	1022.2	0.7	0.4	0.4	1999.0	1902.1	1.4	1.8	3.0	2746.7	1559.3	1.6	0.2	0.1	1891.8	3220.8	3.0	0.1	0.1	2731.3
6	1704.2	3.0	0.1	0.1	1152.3	994.7	2.9	0.6	0.8	2361.9	1845.2	0.1	0.1	0.1	1503.0	3462.2	3.0	1.9	1.2	2158.1
7	2458.2	3.0	0.1	0.1	1173.5	3504.1	2.3	3.0	4.0	2744.1	780.8	2.1	3.0	4.0	2706.2	3662.7	2.0	3.0	0.1	2882.2
8	1019.8	3.0	0.1	0.1	1045.1	516.7	3.0	3.0	4.0	2590.0	3786.4	1.4	0.5	4.0	2457.6	1649.9	3.0	3.0	4.0	2623.2
9	768.0	3.0	0.1	0.1	1125.6	3429.2	2.0	2.0	3.4	2353.4	3491.4	3.0	3.0	4.0	2273.7	1392.0	1.1	0.2	0.2	2547.5
mean	1526.8				1228.0	1952.9				2572.1	2514.6				2420.7	2743.7				2536.0
std	832.5				275.9	1264.7				181.4	1196.2				418.5	954.6				267.8

Table 17: Hopper 2 worst parameters for each iteration over 10 seeds

	$J^{\pi_0}_{T_0}$	ψ^1_1	ψ^1_2	$J^{\pi_0}_{T_1}$	$J^{\pi_1}_{T_1}$	ψ^2_1	ψ^2_2	$J^{\pi_1}_{T_2}$	$J^{\pi_2}_{T_2}$	ψ^3_1	ψ^3_2	$J^{\pi_2}_{T_3}$
0	821.3	3.0	0.1	3091.5	3431.0	3.0	2.9	3101.6	3709.7	2.6	1.5	3061.0
1	636.4	3.0	0.1	2854.2	1318.7	3.0	3.0	3169.6	2055.3	2.9	0.2	1847.3
2	829.4	3.0	3.0	2982.7	2967.9	3.0	0.6	2875.3	1951.7	1.7	0.5	3012.2
3	976.6	3.0	0.1	2906.8	1231.2	1.4	2.6	3294.3	3736.7	3.0	0.6	2902.8
4	872.5	3.0	0.1	3166.5	3601.7	3.0	3.0	3102.7	1338.4	2.9	2.3	3028.8
5	2467.8	3.0	0.1	2720.5	1308.5	1.6	3.0	2995.4	2090.7	2.6	0.2	3112.0
6	1363.3	1.8	0.2	2410.1	3066.3	0.9	3.0	2893.4	1852.3	2.9	2.9	3032.2
7	943.7	3.0	0.1	2952.7	1431.0	1.5	3.0	3061.7	834.7	0.1	3.0	3027.3
8	1177.2	3.0	0.9	2899.6	3491.6	1.4	0.2	3024.6	1491.0	2.3	0.1	3003.1
9	3411.0	3.0	0.1	3027.0	3553.9	1.8	2.9	3283.7	1437.3	3.0	0.1	2689.2
mean	1349.9			2901.2	2540.2			3080.2	2049.8			2871.6
std	889.4			212.9	1068.0			142.9	961.6			378.4

Table 18: Ant 3 worst parameters for each iteration over 10 seeds

	$J^{\pi_0}_{T_0}$	ψ^1_1	ψ^1_2	ψ^1_3	$J^{\pi_0}_{T_1}$	$J^{\pi_1}_{T_1}$	ψ^2_1	ψ^2_2	ψ^2_3	$J^{\pi_1}_{T_2}$	$J^{\pi_2}_{T_2}$	ψ^3_1	ψ^3_2	ψ^3_3	$J^{\pi_2}_{T_3}$	$J^{\pi_3}_{T_3}$	ψ^4_1	ψ^4_2	ψ^4_3	$J^{\pi_3}_{T_4}$
0	2118.9	0.1	0.0	1.2	-613.8	5659.7	3.0	0.0	1.8	-916.5	6973.9	3.0	2.2	0.0	-963.8	6298.0	1.6	1.0	0.3	-98.9
1	1513.4	2.1	1.1	0.5	-188.3	901.9	0.3	0.2	2.8	-151.0	7378.7	0.3	0.2	2.8	-151.0	1856.7	2.1	3.0	3.0	-1469.4
2	5615.7	1.6	2.1	0.0	-1229.9	4238.6	3.0	2.1	3.0	-1542.7	1620.0	3.0	2.1	3.0	-1542.7	3002.5	3.0	2.1	3.0	-1542.7
3	-6.9	2.5	2.1	1.9	-942.4	420.8	1.9	0.0	3.0	-948.5	6022.8	3.0	3.0	0.0	271.4	2891.4	3.0	3.0	0.0	271.4
4	6455.8	1.0	1.7	0.3	-998.6	903.3	0.1	3.0	3.0	-336.2	-6.0	0.1	3.0	3.0	-336.2	861.7	0.1	3.0	3.0	-336.2
5	4932.0	0.6	0.4	2.9	-574.9	6406.5	0.2	2.5	2.2	944.5	893.2	0.1	3.0	3.0	768.9	2012.7	1.9	1.1	1.0	245.3
6	4036.4	1.8	0.9	0.2	-1189.2	1183.7	1.1	1.6	0.9	-852.1	2000.7	2.6	2.2	0.8	-852.0	4910.0	2.6	2.2	0.8	-852.0
7	4989.9	0.5	0.8	1.5	-1174.4	7149.6	1.1	0.0	0.0	-625.6	3224.3	3.0	3.0	0.0	-420.7	7073.9	3.0	3.0	3.0	-150.3
8	1013.7	0.1	0.0	3.0	-698.5	5122.7	3.0	3.0	0.0	746.8	-0.8	3.0	3.0	0.0	746.8	7215.3	3.0	3.0	2.5	-888.1
9	5693.1	3.0	0.0	0.0	-230.2	923.3	3.0	3.0	0.1	48.9	2420.4	0.8	2.3	2.5	-1042.7	3157.8	0.1	2.2	0.1	-663.1
mean	3636.2				-784.0	3291.0				-363.2	3052.7				-352.2	3928.0				-548.4
std	2278.6				384.2	2669.8				782.1	2785.7				775.8	2289.3				646.8

Table 19: Ant 2 worst parameters for each iteration over 10 seeds

	$J^{\pi_0}_{T_0}$	ψ^1_1	ψ^1_2	$J^{\pi_0}_{T_1}$	$J^{\pi_1}_{T_1}$	ψ^2_1	ψ^2_2	$J^{\pi_1}_{T_2}$	$J^{\pi_2}_{T_2}$	ψ^3_1	ψ^3_2	$J^{\pi_2}_{T_3}$
0	334.6	0.1	1.0	-530.0	5991.2	3.0	3.0	4926.7	5675.8	3.0	3.0	4926.7
1	3814.7	0.3	0.4	-1217.9	4217.9	0.8	2.7	-91.6	3002.6	1.9	2.5	812.6
2	90.0	1.3	0.2	-284.3	4617.7	2.3	2.8	-130.4	2558.6	2.3	2.8	-130.4
3	7028.6	0.8	0.4	-1111.3	5489.5	0.1	0.4	4867.3	563.4	0.2	0.6	4987.1
4	2692.2	1.2	2.4	1765.6	11.0	0.7	0.9	-448.8	5211.9	1.0	0.8	-361.8
5	189.2	0.8	0.5	-550.0	6698.5	0.6	0.3	2513.5	4153.0	0.2	1.7	126.8
6	5507.9	1.2	0.6	-317.4	5366.6	0.9	0.6	-1738.6	5327.7	0.9	0.6	-1738.6
7	7135.3	0.3	0.1	-1889.3	1077.5	2.6	1.8	409.8	6587.1	1.8	1.4	-214.7
8	60.2	0.7	1.6	-381.2	3493.4	0.3	1.9	630.1	6718.4	0.3	1.9	630.1
9	5846.8	0.6	1.8	120.7	5218.9	1.0	0.6	911.9	7165.3	0.4	0.2	1762.9
mean	3270.0			-439.5	4218.2			1185.0	4696.4			1080.1
std	2979.0			966.0	2146.9			2233.2	2113.4			2233.0

Table 20: Humanoid 3 worst parameters for each iteration over 10 seeds

	$J^{\pi_0}_{T_0}$	ψ^1_1	ψ^1_2	ψ^1_3	$J^{\pi_0}_{T_1}$	$J^{\pi_1}_{T_1}$	ψ^2_1	ψ^2_2	ψ^2_3	$J^{\pi_1}_{T_2}$	$J^{\pi_2}_{T_2}$	ψ^3_1	ψ^3_2	ψ^3_3	$J^{\pi_2}_{T_3}$	$J^{\pi_3}_{T_3}$	ψ^4_1	ψ^4_2	ψ^4_3	$J^{\pi_3}_{T_4}$
0	162787.3	11.6	2.2	3.7	63777.7	55351.3	12.5	2.2	4.0	55424.1	150555.6	14.7	1.0	2.6	67996.1	165042.5	5.6	1.7	2.3	69635.1
1	69734.9	13.3	4.4	2.2	50795.2	128973.5	6.8	2.4	0.1	48109.7	141463.0	6.8	2.4	0.1	48109.7	76982.5	6.8	2.4	0.1	48109.7
2	114519.9	14.2	3.9	0.1	34651.3	147848.4	13.3	3.8	2.6	34657.0	159586.1	13.3	3.8	2.6	34657.0	159965.9	13.3	3.8	2.6	34657.0
3	135133.6	3.4	3.6	7.1	35652.5	66055.7	15.2	3.6	0.4	42160.9	139086.8	2.4	0.2	6.3	44978.4	121269.4	2.4	0.2	6.3	44978.4
4	63038.4	14.7	4.2	3.8	26449.9	208955.3	3.8	4.5	4.7	29254.3	152341.9	3.8	4.5	4.7	29254.3	154287.1	14.3	0.4	0.4	29789.3
5	30299.9	13.1	1.9	0.8	25853.1	166377.0	9.6	2.6	7.7	35389.8	218770.6	16.0	5.0	8.0	58698.7	171419.5	4.9	3.0	3.3	29918.3
6	97248.9	14.8	3.8	0.3	44581.9	89458.9	13.4	4.9	6.0	56668.7	101301.8	10.1	3.4	1.8	61737.6	143184.0	10.1	3.4	1.8	61737.6
7	53534.5	14.3	1.8	1.3	43299.9	61420.6	8.6	3.0	1.4	40005.2	151359.1	6.1	1.2	7.1	62190.9	92987.9	7.7	0.5	0.5	41136.6
8	76997.8	15.6	2.7	3.0	37371.1	117196.9	12.0	0.5	4.0	56593.5	113955.7	12.0	0.5	4.0	56593.5	149949.0	13.3	0.5	0.7	54796.9
9	202835.2	14.6	4.2	3.9	153604.2	39919.8	10.6	4.6	0.7	141281.9	142541.7	10.6	4.6	0.7	150635.1	139585.8	10.6	4.6	0.7	150635.1
mean	100613.0				51603.7	108155.7				53954.5	147096.2				61485.1	137467.4				56539.4
std	53535.8				37581.6	55167.9				32210.3	31025.0				33718.3	31259.5				35591.2

Table 21: Humanoid 2 worst parameters for each iteration over 10 seeds

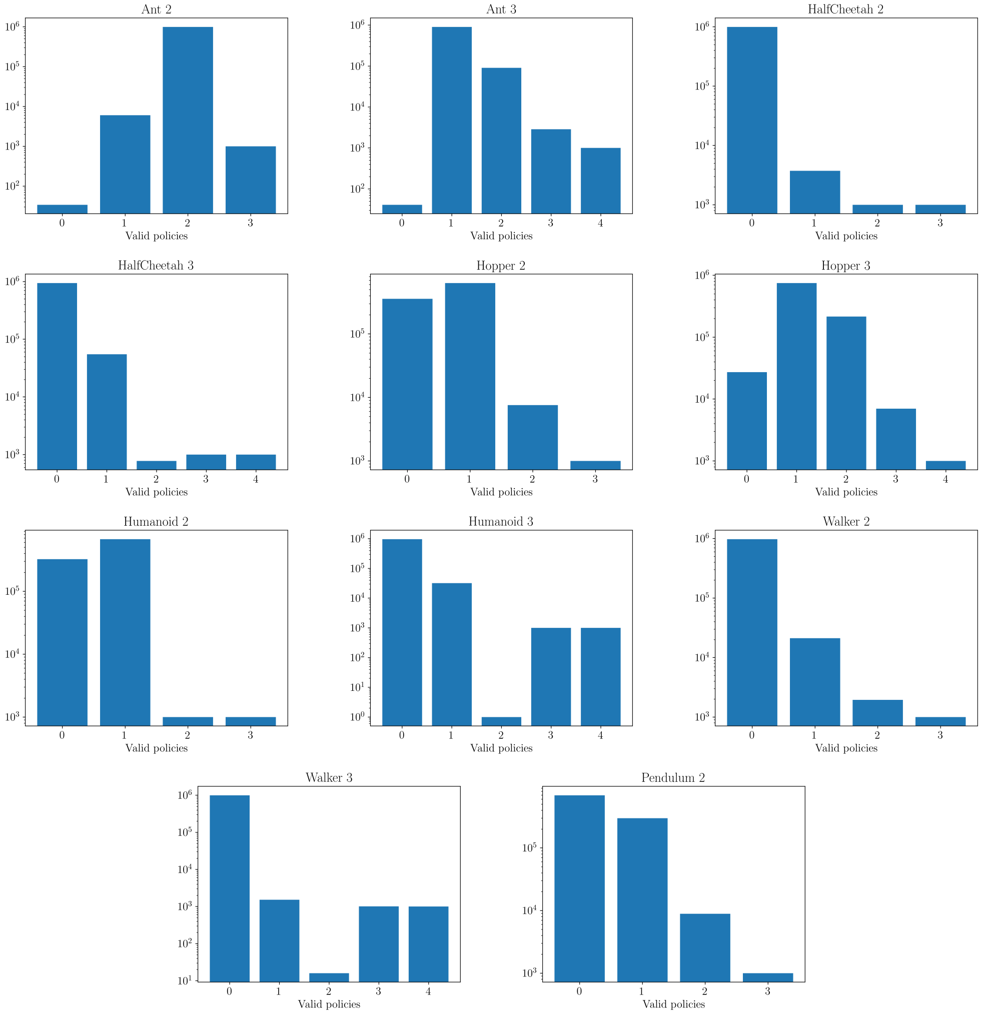
	$J_{T_0}^{\pi_0}$	ψ_1^1	ψ_2^1	$J_{T_1}^{\pi_0}$	$J_{T_1}^{\pi_1}$	ψ_1^2	ψ_2^2	$J_{T_2}^{\pi_1}$	$J_{T_2}^{\pi_2}$	ψ_1^3	ψ_2^3	$J_{T_3}^{\pi_2}$
0	78293.6	13.3	2.7	56554.6	122494.6	12.1	4.8	58367.5	151787.6	11.7	6.7	58896.8
1	107439.5	13.6	5.8	32057.0	147989.1	12.3	4.0	34210.3	140336.8	2.2	0.3	44547.7
2	119721.4	12.6	0.1	42117.4	158982.0	7.5	4.7	53170.1	78200.1	7.5	4.7	53170.1
3	50555.7	15.7	5.2	59878.2	143796.3	14.1	3.7	70100.0	87890.5	14.1	3.7	70100.0
4	75409.0	13.2	2.3	39404.6	94033.2	1.4	0.1	71365.4	238079.6	1.4	0.1	71365.4
5	67846.3	13.7	0.9	43642.5	151772.1	11.4	4.6	63760.4	215882.7	13.1	2.4	60936.7
6	89674.2	3.2	0.3	29934.6	108442.9	10.0	5.7	40954.1	147999.7	10.4	0.2	43334.3
7	160686.9	8.6	1.6	55290.8	47858.6	8.7	0.2	51295.9	147361.0	8.7	0.2	51295.9
8	57969.2	14.7	5.4	36889.2	111322.2	14.0	0.1	52196.3	117439.7	14.0	0.1	52196.3
9	88735.1	13.9	0.3	32775.8	156550.1	9.5	5.7	38304.1	161132.0	13.9	0.3	40780.7
mean	89633.1			42854.5	124324.1			53372.4	148611.0			54662.4
std	32671.8			10882.7	35152.7			12869.5	49870.6			10633.2

Table 22: Median of the worst-case scores across 10 independent runs.

Environment	**Median Performance**
Ant 2	378
Ant 3	-499
Halfcheetah 2	3826
Halfcheetah 3	2177
Hopper 2	3019
Hopper 3	2598
Humanoid 2	52683
Humanoid 3	46544
InvertedPendulum 2	1000
Walker 2	4200
Walker 3	3351

Table 23: Median of the average (on transition functions) scores across 10 independent runs.

Environment	**Median Performance**
Ant 2	3827
Ant 3	1296
Halfcheetah 2	5690
Halfcheetah 3	5030
Hopper 2	3396
Hopper 3	3012
Humanoid 2	123401
Humanoid 3	111418
InvertedPendulum 2	1000
Walker 2	5084
Walker 3	4566



Ant 2

Ant 3

HalfCheetah 2

HalfCheetah 3

Hopper 2

Hopper 3

Humanoid 2

Humanoid 3

Walker 2

Walker 3

Pendulum 2

Figure 5: Counting how many policies are valid in each state

Table 24: Avg. of normalized worst-case performance over 10 seeds for each method with standard deviations. For the sake of brevity: HC = HalfCheetah, HS = HumanoidStandup, IP = InvertedPendulum, IWOCS = IWOCS with CMA-ES, IWOCS* = IWOCS with grid-search.

Environment	IWOCS	IWOCS*	M2TD3	SoftM2TD3	M3DDPG	RARL (DDPG)	DR (TD3)
Ant 2	-0.23 ± 0.43	-0.29 ± 0.29	1.00 ± 0.04	0.92 ± 0.06	-0.72 ± 0.05	-1.32 ± 0.04	0.02 ± 0.05
Ant 3	0.17 ± 0.55	-	1.00 ± 0.09	0.97 ± 0.18	-0.36 ± 0.20	-1.28 ± 0.06	0.61 ± 0.03
HC 2	$\mathbf{1.12 \pm 0.20}$	1.06 ± 0.25	1.00 ± 0.05	1.07 ± 0.05	-0.02 ± 0.02	-0.05 ± 0.02	0.84 ± 0.04
HC 3	$\mathbf{1.63 \pm 0.36}$	-	1.00 ± 0.14	1.39 ± 0.15	-0.03 ± 0.05	-0.13 ± 0.05	1.10 ± 0.04
Hopper 2	$\mathbf{5.74 \pm 0.05}$	5.70 ± 0.00	1.00 ± 0.05	1.09 ± 0.06	0.46 ± 0.06	0.61 ± 0.17	0.87 ± 0.03
Hopper 3	$\mathbf{6.87 \pm 1.70}$	-	1.00 ± 0.09	0.68 ± 0.08	0.22 ± 0.04	0.56 ± 0.17	0.73 ± 0.13
HS 2	0.80 ± 0.14	1.97 ± 0.62	1.00 ± 0.12	1.25 ± 0.16	0.98 ± 0.12	0.88 ± 0.13	1.14 ± 0.14
HS 3	0.80 ± 0.28	-	1.00 ± 0.11	0.96 ± 0.07	0.97 ± 0.07	0.88 ± 0.13	0.86 ± 0.06
IP 2	$\mathbf{2.82 \pm 0.00}$	-	1.00 ± 0.37	0.38 ± 0.08	-0.00 ± 0.00	-0.00 ± 0.00	0.15 ± 0.01
Walker 2	$\mathbf{1.16 \pm 0.42}$	1.19 ± 0.17	1.00 ± 0.14	0.83 ± 0.15	0.04 ± 0.04	-0.08 ± 0.01	0.71 ± 0.17
Walker 3	$\mathbf{1.60 \pm 0.34}$	-	1.00 ± 0.23	1.03 ± 0.20	0.06 ± 0.05	-0.10 ± 0.01	0.65 ± 0.19
Aggregated	$\mathbf{2.04 \pm 0.18}$	-	1.0 ± 0.13	0.96 ± 0.05	0.14 ± 0.06	0.0 ± 0.07	0.70 ± 0.08

Table 25: Avg. ± std. error of worst-case performance over 10 seeds for each method. For the sake of brevity: HC = HalfCheetah, HS = HumanoidStandup, IP = InvertedPendulum, IWOCS = IWOCS with CMA-ES, IWOCS* = IWOCS with grid-search.

Environment	IWOCS	IWOCS*	M2TD3	SoftM2TD3	M2-DDPG	M3DDPG	RARL (DDPG)	DR (TD3)	TD3 (ref)
Ant 2 $(\times 10^3)$	1.0 ± 1.1	0.85 ± 0.73	$\mathbf{4.13} \pm 0.11$	3.92 ± 0.14	0.95 ± 0.20	-0.25 ± 0.13	-1.77 ± 0.09	1.64 ± 0.13	1.59 ± 0.08
Ant 3 $(\times 10^3)$	-0.8 ± 0.6	-	$\mathbf{0.10} \pm 0.10$	0.07 ± 0.20	-1.13 ± 0.28	-1.38 ± 0.22	-2.38 ± 0.07	-0.32 ± 0.03	-0.99 ± 1.13
HC 2 $(\times 10^3)$	$\mathbf{3.0} \pm 0.63$	2.8 ± 0.77	2.61 ± 0.16	2.82 ± 0.16	2.54 ± 0.23	-0.58 ± 0.06	-0.70 ± 0.05	2.12 ± 0.13	-0.53 ± 0.06
HC 3 $(\times 10^3)$	$\mathbf{1.9} \pm 0.55$	-	0.93 ± 0.21	1.53 ± 0.23	1.20 ± 0.22	-0.66 ± 0.08	-0.81 ± 0.07	1.09 ± 0.06	-0.61 ± 0.08
Hopper 2 $(\times 10^2)$	$\mathbf{29.6} \pm 0.25$	29.4 ± 0.02	5.33 ± 0.28	5.79 ± 0.29	4.30 ± 0.57	2.58 ± 0.29	3.34 ± 0.89	4.68 ± 0.15	0.21 ± 0.04
Hopper 3 $(\times 10^2)$	$\mathbf{18.7} \pm 4.6$	-	2.84 ± 0.25	1.98 ± 0.22	2.25 ± 0.29	0.73 ± 0.11	1.64 ± 0.46	2.10 ± 0.35	0.14 ± 0.03
HS 2 $(\times 10^4)$	5.32 ± 0.8	$\mathbf{12.1} \pm 3.6$	6.50 ± 0.70	7.94 ± 0.90	6.24 ± 0.54	6.37 ± 0.72	5.78 ± 0.73	7.31 ± 0.78	0.73 ± 0.07
HS 3 $(\times 10^4)$	5.06 ± 1.6	-	$\mathbf{6.20} \pm 0.64$	5.99 ± 0.37	5.96 ± 0.58	6.01 ± 0.38	5.54 ± 0.76	5.41 ± 0.34	0.57 ± 0.04
IP 2 $(\times 10^2)$	$\mathbf{10} \pm 0$	-	3.56 ± 1.32	1.36 ± 0.30	1.10 ± 0.62	0.02 ± 0.00	0.02 ± 0.00	0.57 ± 0.02	0.03 ± 0.00
Walker 2 $(\times 10^3)$	$\mathbf{3.6} \pm 1.2$	3.7 ± 0.5	3.14 ± 0.39	2.64 ± 0.43	0.85 ± 0.12	0.39 ± 0.11	0.06 ± 0.04	2.31 ± 0.50	0.28 ± 0.07
Walker 3 $(\times 10^3)$	$\mathbf{3.0} \pm 0.6$	-	1.94 ± 0.40	2.00 ± 0.35	0.82 ± 0.13	0.28 ± 0.09	0.00 ± 0.02	1.32 ± 0.34	$0.17 + 0.06$