Time-Constrained Robust MDPs

Adil Zouitine [*,1,2], **David Bertoin** [*,1,2,3], **Pierre Clavier** [*,4,5]

[1]ISAE-SUPAERO, [2]IRT Saint-Exupery, [3]INSA Toulouse
[4]Ecole polytechnique, CMAP ; [5]INRIA Paris, HeKA.

{adil.zouitine, david.bertoin}@irt.saintexupery.com,
pierre.clavier@polytechnique.edu.

Matthieu Geist[6], Emmanuel Rachelson[1]

[6]Cohere

Abstract

Robust reinforcement learning is essential for deploying reinforcement learning algorithms in real-world scenarios where environmental uncertainty predominates. Traditional robust reinforcement learning often depends on rectangularity assumptions, where adverse probability measures of outcome states are assumed to be independent across different states and actions. This assumption, rarely fulfilled in practice, leads to overly conservative policies. To address this problem, we introduce a new time-constrained robust MDP (TC-RMDP) formulation that considers multifactorial, correlated, and time-dependent disturbances, thus more accurately reflecting real-world dynamics. This formulation goes beyond the conventional rectangularity paradigm, offering new perspectives and expanding the analytical framework for robust RL. We propose three distinct algorithms, each using varying levels of environmental information, and evaluate them extensively on continuous control benchmarks. Our results demonstrate that these algorithms yield an efficient tradeoff between performance and robustness, outperforming traditional deep robust RL methods in time-constrained environments while preserving robustness in classical benchmarks. This study revisits the prevailing assumptions in robust RL and opens new avenues for developing more practical and realistic RL applications.

1 Introduction

Robust MDPs capture the problem of finding a control policy for a dynamical system whose transition kernel is only known to belong to a defined uncertainty set. The most common framework for analyzing and deriving algorithms for robust MDPs is that of sa-rectangularity [1, 2], where probability measures on outcome states are picked independently in different source states and actions (in formal notation, $\mathbb{P}(s'|s,a)$ and $\mathbb{P}(s'|\bar{s},\bar{a})$ are independent of each other). This provides an appreciable decoupling of worst transition kernel search across time steps and enables sound algorithms like robust value iteration (RVI). But policies obtained for such sa-rectangular MDPs are by nature very conservative [3, 4], as they enable drastic changes in environment properties from one time step to the next, and the algorithms derived from RVI tend to yield very conservative policies even when applied to non-sa-rectangular robust MDP problems.

In this paper, we depart from the rectangularity assumption and turn towards a family of robust MDPs whose transition kernels are parameterized by a vector ψ. This parameter vector couples together the outcome probabilities in different (s,a) pairs, hence breaking the independence assumption that

17th European Workshop on Reinforcement Learning (EWRL 2024).

is problematic, especially in large dimension [3]. This enables accounting for the notion of transition model consistency across states and actions: outcome probabilities are not picked independently anymore but are rather set across the state and action spaces by drawing a parameter vector. In turn, we examine algorithms for solving such parameter-based robust MDPs when the parameter is constrained to follow a bounded evolution throughout time steps. Our contributions are the following.

1. We introduce a formal definition for parametric robust MDPs and time-constrained robust MDPs, discuss their properties and derive a generic algorithmic framework (Sec. 2).
2. We propose three algorithmic variants for solving time-constrained MDPs, named vanilla `TC`, `Stacked-TC` and `Oracle-TC` (Sec. 4), which use different levels of information in the state space, and come with theoretical guaranties (Sec. 6).
3. These algorithms are extensively evaluated in MuJoCo [5] benchmarks, demonstrating they lead to non-conservative and robust policies (Sec. 5).

2 Problem statement

(Robust) MDPs. A Markov Decision Process (MDP) [6] is a model of a discrete-time, sequential decision making task. At each time step, from a state $s_t \in S$ of the MDP, an action $a_t \in A$ is taken and the state changes to s_{t+1} according to a stationary Markov transition kernel $p(s_{t+1}|s_t, a_t)$, while concurrently receiving a reward $r(s_t, a_t)$. S and A are measurable sets and we write Δ_S and Δ_A the set of corresponding probability distributions. A stationary policy $\pi(\cdot|s)$ is a mapping from states to distributions over actions, prescribing which action should be taken in s. The value function v_p^π of policy π maps state s to the expected discounted sum of rewards $\mathbb{E}_{p,\pi}[\sum_t \gamma^t r_t]$ when applying π from s for an infinite number of steps. An optimal policy for an MDP is one whose value function is maximal in any state. In a Robust MDP (RMDP) [1, 2], the transition kernel p is not set exactly and can be picked in an adversarial manner at each time step, from an uncertainty set $\mathcal{P}$. Then, the pessimistic value function of a policy is $v_{\mathcal{P}}^\pi(s) = \min_{p\in\mathcal{P}} v_p^\pi(s)$. An optimal robust policy is one that has the largest possible pessimistic value function $v_{\mathcal{P}}^*$ in any state, hence yielding an adversarial $\max_\pi \min_p$ optimization problem. Robust Value Iteration (RVI) [1, 7] solves this problem by iteratively computing the one-step lookahead best pessimistic value:

$$v_{n+1}(s) = T_{\mathcal{P}}^* v_n(s) := \max_{\pi(s)\in\Delta_A} \min_{p\in\mathcal{P}} \mathbb{E}_{a\sim\pi(s)}[r(s,a) + \mathbb{E}_p[v_n(s')]].$$

The $T_{\mathcal{P}}^*$ operator is called the robust Bellman operator and the sequence of v_n functions converges to the robust value function $v_{\mathcal{P}}^*$ as long as the adversarial transition kernel belongs to the simplex of Δ_S.

Zero-sum Markov Games. Robust MDPs can be cast as zero-sum two-players Markov games [8, 9] where B is the action set of the adversarial player. Writing $\bar{\pi} : S \times A \to \Delta_B$ the policy of this adversary, the robust MDP problem turns to $\max_\pi \min_{\bar{\pi}} v^{\pi,\bar{\pi}}$, where $v^{\pi,\bar{\pi}}(s)$ is the expected sum of discounted rewards obtained when playing π (agent actions) against $\bar{\pi}$ (transition models) at each time step from s. This enables introducing the robust value iteration sequence of functions

$$v_{n+1}(s) := T^{**} v_n(s) := \max_{\pi(s)\in\Delta_A} \min_{\bar{\pi}(s,a)\in\Delta_S} (T^{\pi,\bar{\pi}} v_n)(s)$$

where $T^{\pi,\bar{\pi}} := \mathbb{E}_{a\sim\pi(s)}[r(s,a) + \gamma \mathbb{E}_{s'\sim\bar{\pi}(s,a)} v_n(s')]$ is a zero-sum Markov game operator. These operators are also $\gamma-$contractions and converge to their respective fixed point $v^{\pi,\bar{\pi}}$ and $v^{**} = v_{\mathcal{P}}^*$ [9]. This formulation will be useful to derive a practical algorithm in Section 4.

Often, this convergence is analyzed under the assumption of sa-rectangularity, stating that the uncertainty set $\mathcal{P}$ is a set product of independent subsets of Δ_S in each s, a pair. Quoting [1], rectangularity is a sort of independence assumption and is a minimal requirement for most theoretical results to hold. Within robust value iteration, rectangularity enables picking $\bar{\pi}(s_t, a_t)$ completely independently of $\bar{\pi}(s_{t-1}, a_{t-1})$. To set ideas, let us consider the robust MDP of a pendulum, described by its mass and rod length. Varying this mass and rod length spans the uncertainty set of transition models. The rectangularity assumption induces that $\bar{\pi}(s_t, a_t)$ can pick a measure in Δ_S corresponding to a mass and a length that are completely independent from the ones picked in the previous time step. While this might be a good representation in some cases, in general it yields policies that are very conservative as they optimize for adversarial configurations which might not occur in practice.

We first step away from the rectangularity assumption and define a parametric robust MDP as an RMDP whose transition kernels are spanned by varying a parameter vector ψ (typically the mass

and rod length in the previous example). Choosing such a vector couples together the probability measures on successor states from two distinct (s, a) and (s', a') pairs. The main current robust deep RL algorithms actually optimize policies for such parametric robust MDPs but still allow the parameter value at each time step to be picked independently of the previous time step.

Parametric MDPs. A parametric RMDP is given by the tuple (S, A, Ψ, p_ψ, r) where the transition kernel $p_\psi(s, a) \in \Delta_S$ is parameterized by ψ, and Ψ is the set of values ψ can take, equipped with an appropriate metric. This yields the robust value iteration update :

$$v_{n+1}(s) = \max_{\pi(s)\in\Delta_A} \min_{\psi\in\Psi} (T^\pi_\psi v_n)(s) := \max_{\pi(s)\in\Delta_A} \min_{\psi\in\Psi} \mathbb{E}_{a\sim\pi(s)}[r(s, a) + \gamma\mathbb{E}_{s'\sim p_\psi(s,a)} v_n(s')].$$

A parametric RMDP remains a Markov game and the Bellman operator remains a contraction mapping as long as p_ψ can reach only elements in the simplex of Δ_S, where the adversary's action set is the set of parameters instead of a (possibly sa-rectangular) set of transition kernels.

Time-constrained RMDPs (TC-RMDPs). We introduce TC-RMDPs as the family of parametric RMDPs whose parameter's evolution is constrained to be Lipschitz with respect to time. More formally a TC-RMDP is given by the tuple $(S, A, \Psi, p_\psi, r, L)$, where $\|\psi_{t+1} - \psi_t\| \leq L$, that is the parameter change is bounded through time. In the previous pendulum example, this might represent the wear of the rod which might lose mass or stretch length. Similarly, and for a larger scale illustration, TC-RMDPs enable representing the possible evolutions of traffic conditions in a path planning problem through a busy town. Starting from an initial parameter value ψ_{-1}, the pessimistic value function of a policy π is non-stationary, as ψ_0 is constrained to lay at most L-far away from ψ_{-1}, ψ_1 from ψ_0, and so on. Generally, this yields non-stationary value functions as the uncertainty set at each time step depends on the previous uncertainty parameter. To regain stationarity without changing the TC-RMDP definition, we first change the definition of the adversary's action set. The adversary picks its actions in the constant set $B = \mathcal{B}(0_\Psi, L)$, which is the ball of radius L centered in the null element in Ψ. In turn, the state of the Markov game becomes the pair s, ψ and the Markov game itself is given by the tuple $((S \times \Psi), A, B, p_\psi, r)$, where the Lipschitz constant L is included in B. Thus, given an action $b_t \in B$ and a previous parameter value ψ_{t-1}, the parameter value at time t is $\psi_t = \psi_{t-1} + b_t$. Then, we define the pessimistic value function of a policy as a function of both the state s and parameter ψ:

$$v^\pi_B(s, \psi) := \min_{\substack{(b_t)_{t\in\mathbb{N}},\\ b_t\in B}} \mathbb{E}\Big[\sum \gamma^t r_t | \psi_{-1} = \psi, s_0 = s, b_t \in B, \psi_t = \psi_{t-1} + b_t, a \sim \pi, s_t \sim p_{\psi_t}\Big],$$

$$v^*_B(s, \psi) = \max_{\pi(s,\psi)\in\Delta_A} v^\pi_B(s, \psi).$$

In turn, an optimal robust policy is a function of s and ψ and the TC robust Bellman operators are:

$$\begin{aligned} v_{n+1}(s, \psi) &:= T^*_B v_n(s, \psi) := \max_{\pi(s,\psi)\in\Delta_A} T^\pi_B v_n(s, \psi), \\ &:= \max_{\pi(s,\psi)\in\Delta_A} \min_{b\in B} \mathbb{E}_{a\sim\pi(s)}[r(s, a) + \gamma\mathbb{E}_{s'\sim p_{\psi+b}(s,a)} v_n(s', \psi + b)]. \end{aligned}$$

This iteration scheme converges to a fixed point according to Th. 2.1.

Theorem 2.1. *The time-constrained (TC) Bellman operators T^π_B and T^*_B are contraction mappings. Thus the sequences $v_{n+1} = T^\pi_B v_n$ and $v_{n+1} = T^*_B v_n$, converge to their respective fixed points v^π_B and v^*_B.*

Proof of Th. 2.1 can be found in Appendix B. We refer to this formulation as algorithm `Oracle-TC` (see Section 4 for implementation details) since an oracle makes the current parameter ψ visible to the agent. Therefore, it is possible to derive optimal policies for TC-RMDPs by iterated application of this TC Bellman operator. These policies have the form $\pi(s, \psi)$. In the remainder of this paper, we extend state-of-the-art robust deep RL algorithms to the TC-RMDP framework. In particular, we compare their performance and robustness properties with respect to classical robust MDP formulations, we also discuss their relation with the $\pi(s)$ robust policies of classical robust MDPs.

If the agent is unable to observe the state variable ψ, it is not possible to guarantee the existence of a stationary optimal policy of the form $\pi(s)$. Similarly, there is no guarantee of convergence of value functions to a fixed point. Nonetheless, this scenario, in which access to the ψ parameter is not available, is more realistic in practice. It turns the two-player Markov game into a partially observable

Markov game, where one can still apply the TC Bellman operator but without these guarantees of convergence. We call vanilla TC the repeated application of the TC Bellman operator in this partially observable case. Vanilla TC will be tested in practice, and some theoretical properties of the objective function will be derived using the Lipschitz properties (Sec 6).

3 Related works

Since our method is a non-rectangular, Deep Robust RL algorithm, (possibly non-stationary for Stacked-TC and TC), we discuss the following related work.

Non-stationary MDPs. First, non-stationarity has been studied in the Bandits setting in [10]. Then, for episodic, non-stationary MDPs [11, 12, 13] have explored and provided regret bounds for algorithms that use oracle access to the current reward and transition functions. More recently [14, 15] have facilitated oracle access by performing a count-based estimation of the reward and transition functions based on the recent history of interactions. Finally, for tabular MDPs, past data from a non-stationary MDP can be used to construct a full Bayesian model [16] or a maximum likelihood model [17] of the transition dynamics. We focus on the setting not restricted to tabular representations

Non-rectangular RMDPs. While rectangularity in practice is very conservative, it can be demonstrated that, in an asymptotic sense, non-rectangular ellipsoidal uncertainty sets around the maximum likelihood estimator of the transition kernel constitute the smallest possible confidence sets for the ground truth transition kernel, as implied by classical Cramér-Rao bounds. This is in accordance with the findings presented in § 5 and Appendix A of [7]. More recently, [3] extends the rectangular assumptions using a factored uncertainty model, where all transition probabilities depend on a small number of underlying factors denoted $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_r \in \mathbb{R}^{\mathbb{S}}$, such that each transition probability P_{sa} for every (s, a) is a linear (convex) combination of these r factors. Finally, [4] use policy gradient algorithms for non-rectangular robust MDPs. While this work presents nice theoretical guarantees of convergence, there is no practical Deep RL algorithms for learning optimal robust policies.

Deep Robust RL Methods. Many Deep Robust algorithms exist such as M2TD3 [18], M3DDPG [19], or RARL [20], which are all based on the two player zero-sum game presented in 2. We will compare our method against these algorithms, except [19] which is outperformed by [18] in general. We also compare our algorithm to Domain randomization (DR) [21] that learns a value function $V(s) = \max_\pi \mathbb{E}_{p\sim\mathcal{U}(\mathcal{P})} V_p^\pi(s)$ which maximizes the expected return on average across a fixed (generally uniform) distribution on $\mathcal{P}$. As such, DR approaches do not optimize the worst-case performance but still have good performance on average. Nonetheless, DR has been used convincingly in applications [22, 23]. Finally, the zero-sum game formulation has lead to the introduction of action robustness [9] which is a specific case of rectangular MDPs, in scenarios where the adversary shares the same action space as the agent and interferes with the agent's actions. Several strategies based on this idea have been proposed. One approach, the Game-theoretic Response Approach for Adversarial Defense (GRAD) [24] builds on the Probabilistic Action Robust MDP (PR-MDP) [9]. This method introduces time-constrained perturbations in both the action and state spaces and employs a game-theoretic approach with a population of adversaries. In contrast to GRAD, where temporal disturbances affect the transition kernel around a nominal kernel, our method is part of a broader setting in which the transition kernel is included in a larger uncertainty set. Robustness via Adversary Populations (RAP) [25] introduces a population of adversaries. This approach ensures that the agent develops robustness against a wide range of potential perturbations, rather than just a single one, which helps prevent convergence to suboptimal stationary points. Similarly, State Adversarial MDPs [26, 27, 28, 24] address adversarial attacks on state observations, effectively creating a partially observable MDP. Finally, using rectangularity assumptions, [29, 30] use Wasserstein and χ^2 balls respectively for the uncertainty set in Robust RL.

4 Time-constrained robust MDP algorithms

The TC-RMDP framework addresses the limitations of traditional robust reinforcement learning by considering multifactorial, correlated, and time-dependent disturbances. Traditional robust reinforcement learning often relies on rectangularity assumptions, which are rarely met in real-

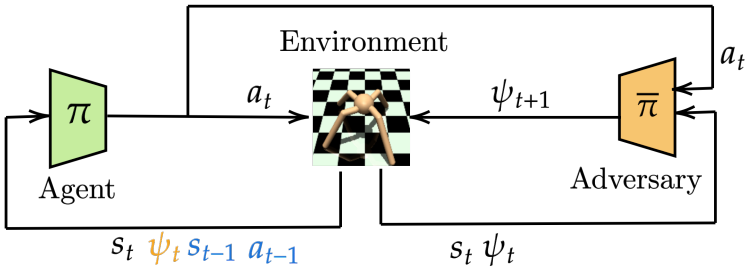


Figure 1: TC-RMDP training involves a temporally-constrained adversary aiming to maximize the effect of temporally-coupled perturbations. Conversely, the agent aims to optimize its performance against this time-constrained adversary. In orange, the oracle observation, and in blue the stacked observation.

world scenarios, leading to overly conservative policies. The TC-RMDP framework provides a more accurate reflection of real-world dynamics, moving beyond the conventional rectangularity paradigm.

We cast the TC-RMDP problem as a two-player zero-sum game, where the agent interacts with the environment, and the adversary (nature) changes the MDP parameters ψ. Our approach is generic and can be derived within any robust value iteration scheme, performing $\max_{\pi(s)\in\Delta_A} \min_{\psi\in\Psi} \mathbb{E}_{a\sim\pi(s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p_\psi(s,a)} v_n(s')]$ updates, by modifying the adversary's action space and potentially the agent's state space to obtain updates of the form $\max_{\pi(s,\psi)\in\Delta_A} \min_{b\in B} \mathbb{E}_{a\sim\pi(s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p_{\psi+b}(s,a)} v_n(s')]$. In Section 5, we will introduce time constraints within two specific robust value iteration algorithms, namely RARL [20] and M2TD3 [18] by simply limiting the search space for worst-case ψ at each step. This specific implementation extends the original actor-critic algorithms. For the sake of conciseness, we refer the reader to Appendix E.1 for details regarding the loss functions and algorithmic details.

Three variations of the algorithm are provided (illustrated in Figure 1) but all fall within the training loop of Algorithm 1.

Algorithm 1 Time-constrained robust training

Input: Time-constrained MDP: $(S, A, \Psi, p_\psi, r, L)$, Agent π, Adversary $\bar{\pi}$
for *each interaction time step* t **do**

$a_t \sim \pi_t(s_t, \psi_t)$ // Sample an action with Oracle-TC
or $a_t \sim \pi_t(s_t, a_{t-1}, s_{t-1})$ // Sample an action with Stacked-TC
or $a_t \sim \pi_t(s_t)$ // Sample an action with TC
$\psi_{t+1} \sim \bar{\pi}_t(s_t, a_t, \psi_t)$ // Sample the worst TC parameter
$s_{t+1} \sim p_{\psi_{t+1}}(s_t, a_t)$ // Sample a transition
$\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r(s_t, a_t), \psi_t, \psi_{t+1}, s_{t+1})\}$ // Add transition to replay buffer
$\{s_i, a_i, r(s_i, a_i), \psi_i, \psi_{i+1}, s_{i+1}\}_{i\in[1,N]} \sim \mathcal{B}$ // Sample a mini-batch of transitions
$\pi_{t+1} \leftarrow UpdatePolicy(\pi_t)$ // Update Agent
$\bar{\pi}_{t+1} \leftarrow UpdatePolicy(\bar{\pi}_t)$ // Update Adversary

Oracle-TC . As discussed in Section 2, the Oracle-TC version includes the MDP state and parameter value as input, $\pi : \mathcal{S} \times \Psi \to \mathcal{A}$. This method assumes that the agent has access to the true parameters of the environment, allowing it to make the most informed decisions and possibly reach the true robust value function. However, these parameters ψ are sometimes non-observable in practical scenarios, making this method not always feasible.

Stacked-TC . Since ψ might not be observable but may be approximately identified by the last transitions, the Stacked-TC policy uses the previous state and action as additional inputs in an attempt to replace ψ, $\pi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{A}$. This approach leverages the information in the transitions, even though it might be insufficient for a perfect estimate of ψ. It aims to retain (approximately) the convergence properties of the Oracle-TC algorithm.

Vanilla TC . Finally, the vanilla TC version takes only the state, $\pi : \mathcal{S} \to \mathcal{A}$, as input, similar to standard robust MDP policies. This method does not attempt to infer the environmental parameters or the transition dynamics explicitly. Instead, it relies on the current state information to guide the agent's actions. While this version is the most straightforward and computationally efficient, it may not perform as robustly as the Oracle-TC or Stacked-TC versions in environments with significant temporal disturbances, since it attempts to solve a partially observable Markov game, for which there may not exist a stationary optimal policy based only on the observation. Despite this, it remains a viable option in scenarios where computational simplicity and quick decision-making are prioritized.

5 Results

Experimental settings. This section evaluates the robust time-constrained algorithm's performance under severe time constraints and in the static settings. Experimental validation was conducted in continuous control scenarios using the MuJoCo simulation environments [5]. The approach was categorized into three variants. The Oracle-TC , where the agent accessed environmental parameters $\pi(s_t, \psi)$; the Stacked-TC , where the agent took in input $\pi(s_t, s_{t-1}, a_{t-1})$; and the vanilla TC , which did not receive any additional inputs $\pi(s)$. For each variant of the time-constrained algorithms, we applied them to RARL [20], and M2TD3 [18], renaming them TC-RARL and TC-M2TD3, respectively. The algorithms were tested against two state-of-the-art robust reinforcement learning algorithms, M2TD3 and RARL. Additionally, the Oracle versions of M2TD3 and RARL, where the agent's policy included ψ in the input $\pi : \mathcal{S} \times \Psi \to \mathcal{A}$, were evaluated for a more comprehensive assessment. Comparisons were also made with Domain Randomization (DR) [21] and vanilla TD3. [31] to ensure a thorough analysis. A 3D uncertainty set is defined in each environment $\mathcal{P}$ normalized between $[0, 1]^3$. Appendix G provides detailed descriptions of uncertainty parameters. Performance metrics were gathered after five million steps to ensure a fair comparison. All baselines were constructed using TD3, and a consistent architecture was maintained across all TD3 variants. The results presented below were obtained by averaging over ten distinct random seeds. Appendices E.4, E.3, E.2, and H.2 discuss further details on hyperparameters, network architectures, and implementation choices, including training curves for our methods and baseline comparisons. In the following tables 1, 2, 3, the best performances are shown in bold. Oracle methods, with access to optimal information, are shown in black. Items in bold and green represent the best performances with limited information on ψ, making them more easily usable in many scenarios. When there is only one element in bold and green, this implies that the best overall method is a non-oracle method.

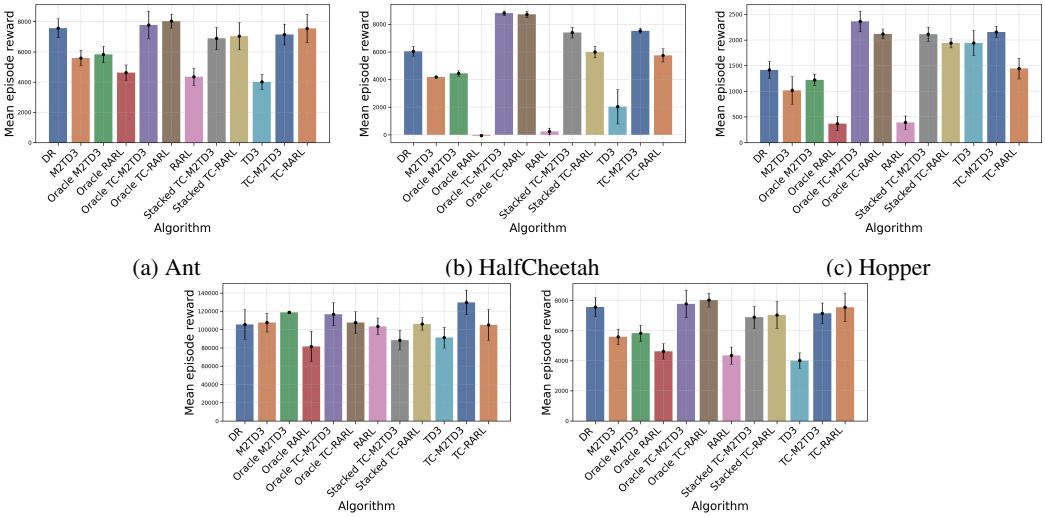
Table 1: Avg. of normalized time-coupled worst-case performance over 10 seeds for each method

	Ant	HalfCheetah	Hopper	Humanoid	Walker	Agg.
Oracle M2TD3	1.11 ± 0.07	0.95 ± 0.1	1.51 ± 0.84	2.07 ± 0.19	1.31 ± 0.36	1.39 ± 0.31
Oracle RARL	0.72 ± 0.18	-0.71 ± 0.05	-1.3 ± 0.28	-2.8 ± 1.62	-0.19 ± 0.2	-0.86 ± 0.47
Oracle-TC -M2TD3	1.61 ± 0.32	$\mathbf{2.76 \pm 0.16}$	$\mathbf{7.79 \pm 1.0}$	1.69 ± 2.14	1.49 ± 0.41	$\mathbf{3.07 \pm 0.81}$
Oracle-TC -RARL	$\mathbf{1.66 \pm 0.32}$	2.63 ± 0.12	6.86 ± 1.46	0.19 ± 1.68	1.34 ± 0.11	2.54 ± 0.74
Stacked-TC -M2TD3	1.33 ± 0.21	$\mathbf{2.4 \pm 0.19}$	$\mathbf{6.51 \pm 0.59}$	-1.42 ± 1.44	$\mathbf{1.69 \pm 0.33}$	2.1 ± 0.55
Stacked-TC -RARL	1.48 ± 0.22	1.76 ± 0.08	3.28 ± 0.27	1.39 ± 0.57	1.01 ± 0.21	1.78 ± 0.27
TC -M2TD3	1.52 ± 0.2	$\mathbf{2.42 \pm 0.1}$	5.16 ± 0.2	$\mathbf{4.02 \pm 1.23}$	1.38 ± 0.25	$\mathbf{2.9 \pm 0.4}$
TC -RARL	1.57 ± 0.26	1.54 ± 0.15	2.04 ± 0.49	1.25 ± 1.91	0.89 ± 0.2	1.46 ± 0.6
TD3	0.0 ± 0.19	0.0 ± 0.27	0.0 ± 1.27	0.0 ± 1.18	0.0 ± 0.23	0.0 ± 0.63
DR	$\mathbf{1.58 \pm 0.2}$	1.59 ± 0.12	2.28 ± 0.42	0.87 ± 1.79	1.03 ± 0.19	1.47 ± 0.54
M2TD3	1.0 ± 0.19	1.0 ± 0.14	1.0 ± 0.96	1.0 ± 1.31	1.0 ± 0.31	1.0 ± 0.58
RARL	0.63 ± 0.2	-0.61 ± 0.18	-1.5 ± 0.33	0.8 ± 0.88	0.27 ± 0.25	-0.08 ± 0.37

Performance of TCRMDPs in worst-case time-constrained. Table 1 reports the worst-case time-constrained perturbation. To address the worst-case time-constrained perturbations for each trained agent π^*, we utilized a time-constrained adversary using TD3 algorithm $\bar{\pi}^* = \min_{b \in B} \mathbb{E}_{a \sim \pi^*(s), b \sim \bar{\pi}(s,a,\psi)}[r(s,a) + \gamma \mathbb{E} s' \sim p_{\psi+b}(s,a) v_n(s')]$ within a perturbation radius of $L = 0.001$ for a total of 5 million steps. The sum of episode rewards was averaged over 10 episodes. To compare metrics across different environments, each method's score v was standardized relative to the reference score of TD3. TD3 was trained on the environment using default transition function parameters, with its score denoted as v_{TD3}. The M2TD3 score, v_{M2TD3}, was used as the comparison target. The formula applied was $(v - v_{TD3})/(|v_{M2TD3} - v_{TD3}|)$. This positioned v_{TD3} as the minimal baseline and v_{M2TD3} as the target score. This standardisation provides a metric that quantifies the improvement of each method over TD3 in relation to the improvement of M2TD3 over

TD3. In each evaluation environment, agents trained with the time-constrained framework (indicated by TC in the method name) demonstrated significantly superior performance compared to those trained using alternative robust reinforcement learning approaches, including M2TD3 and RARL. Furthermore, they outperformed those trained through domain randomisation (DR). Notably, even without directly conditioning the policy with ψ, the time-constrained trained policies excelled against all baselines, achieving up to a 2.9-fold improvement. The non-normalized scores are reported in Appendix H. Additionally, when policies were directly conditioned by ψ and trained within the robust reinforcement learning framework, they tended to be overly conservative in the time-constrained framework. This is depicted in Table 1, comparing the performances of Oracle RARL, Oracle M2TD3, Oracle TC-RARL, and Oracle TC-M2TD3. Both policies also observe ψ. The only difference is that Oracle RARL and Oracle M2TD3 were trained in the robust reinforcement learning framework, while Oracle TC-RARL and Oracle TC-M2TD3 were trained in the time-constrained framework. The performance differences under worst-case time-coupled perturbation are as follows: for Oracle RARL (resp. M2TD3) and Oracle TC-RARL (resp. M2TD3), the values are -0.86 (1.39) vs. 2.54 (3.07). This observation highlights the need for a balance between robust training and flexibility in dynamic conditions. A natural question arises regarding the worst-case time-constrained perturbation. Was the adversary in the loop adequately trained, or might its suboptimal performance lead to overestimating the trained agent's reward against the worst-case perturbation? The adversary's performance was monitored during its training against all fixed-trained agents. The results in Appendix F show that our adversary converged.

Robust Time-Constrained Training under various time fixed adversaries. The method was evaluated against various fixed adversaries, focusing on the random fixed adversary shown in Figure 2. This evaluation shows that robustly trained agents can handle dynamic and unpredictable conditions. The random fixed adversary simulates stochastic changes by selecting a parameter ψ_t at each timestep within a radius of $L = 0.1$. This radius is 100 times larger than in our training methods. At the start of each episode, ψ_0 is uniformly sampled from the uncertainty set $\psi_0 \sim \mathcal{U}(\mathcal{P})$. This tests the agents' adaptability to unexpected changes. Figures 2a through 2e show our agents' performance. Agents trained with our robust framework consistently outperformed those trained with standard methods. The policy was also assessed against five other fixed adversaries: cosine, exponential, linear, and logarithmic. Detailed results are provided in the Appendix. H.1.



(a) Ant (b) HalfCheetah (c) Hopper

(d) Humanoid (e) Walker

Figure 2: Evaluation against a random fixed adversary, with a radius $L = 0.1$

Performance of Robust Time-Constrained MDPs in the static setting. In static environments, the Robust Time-Constrained algorithms were evaluated for worst-case and average performance metrics, shown in Tables 2 and 3. A fixed uncertainty set $\mathcal{P}$ was used, dividing each dimension of Ψ into ten segments, creating a grid of 1000 points (10^3). Each agent ran five episodes at each grid point, and the rewards were averaged. The scores were normalized as described for the time-constrained adversary analysis in Table 1. The raw data is provided in Appendix 9 and 10.

Performance scores were adjusted relative to the baseline v_{TD3} and v_{M2TD3}. As a result, normalized results reveal distinct trends among agent configurations within the TC-RMDP framework. The Oracle TC-M2TD3 variant achieved an average score of 3.12 3, while the Stacked TC-M2TD3 scored 2.23, indicating its resilience. Furthermore, in the worst-case scenario, the TC-RARL and Stacked TC-RARL variants demonstrated adaptability, with TC-RARL scoring 0.92 and TC-M2TD3 scoring 1.02 2. This performance highlights its reliability in challenging static environments.

Table 2: Avg. of normalized static worst-case performance over 10 seeds for each method

	Ant	HalfCheetah	Hopper	Humanoid	Walker	Agg
Oracle M2TD3	$\mathbf{1.02 \pm 0.19}$	0.34 ± 0.23	0.97 ± 0.55	$\mathbf{3.9 \pm 3.65}$	0.3 ± 0.45	$\mathbf{1.31 \pm 1.01}$
Oracle RARL	0.62 ± 0.32	0.1 ± 0.02	0.48 ± 0.19	-2.59 ± 2.18	0.16 ± 0.21	-0.25 ± 0.58
Oracle-TC -M2TD3	0.1 ± 0.25	$\mathbf{1.87 \pm 0.1}$	0.49 ± 1.07	-0.8 ± 3.05	0.28 ± 0.38	0.39 ± 0.97
Oracle-TC -RARL	0.59 ± 0.36	1.55 ± 0.35	0.4 ± 0.16	1.19 ± 1.24	0.56 ± 0.39	0.86 ± 0.5
Stacked-TC -M2TD3	-0.05 ± 0.09	$\mathbf{1.56 \pm 0.16}$	1.08 ± 0.89	-0.83 ± 2.62	1.12 ± 0.5	0.58 ± 0.85
Stacked-TC -RARL	0.07 ± 0.13	0.76 ± 0.34	$\mathbf{1.35 \pm 0.93}$	$\mathbf{1.75 \pm 2.48}$	0.67 ± 0.32	0.92 ± 0.84
TC -M2TD3	-0.06 ± 0.08	1.49 ± 0.23	1.29 ± 0.29	1.21 ± 2.44	$\mathbf{1.19 \pm 0.34}$	$\mathbf{1.02 \pm 0.68}$
TC -RARL	0.14 ± 0.24	0.89 ± 0.3	1.5 ± 0.76	1.4 ± 4.57	0.67 ± 0.59	0.92 ± 1.29
TD3	0.0 ± 0.34	0.0 ± 0.06	0.0 ± 0.21	0.0 ± 2.27	0.0 ± 0.1	0.0 ± 0.6
DR	0.06 ± 0.16	1.07 ± 0.36	0.86 ± 0.82	0.04 ± 4.1	0.57 ± 0.37	0.52 ± 1.16
M2TD3	$\mathbf{1.0 \pm 0.27}$	1.0 ± 0.16	1.0 ± 0.65	1.0 ± 3.32	1.0 ± 0.63	1.0 ± 1.01
RARL	0.44 ± 0.3	0.13 ± 0.08	0.5 ± 0.22	0.44 ± 2.94	0.12 ± 0.09	0.33 ± 0.73

Table 3: Avg. of normalized static average case performance over 10 seeds for each method

	Ant	HalfCheetah	Hopper	Humanoid	Walker	Agg
Oracle M2TD3	1.13 ± 0.08	1.56 ± 0.24	1.12 ± 0.46	1.96 ± 1.53	1.23 ± 0.3	1.4 ± 0.52
Oracle RARL	0.7 ± 0.22	-1.4 ± 0.13	-0.77 ± 0.24	-2.6 ± 2.88	-1.13 ± 0.84	-1.04 ± 0.86
Oracle-TC -M2TD3	1.73 ± 0.09	$\mathbf{4.35 \pm 0.26}$	$\mathbf{5.54 \pm 0.13}$	2.12 ± 1.4	1.84 ± 0.37	$\mathbf{3.12 \pm 0.45}$
Oracle-TC -RARL	$\mathbf{1.78 \pm 0.02}$	4.32 ± 0.21	5.08 ± 0.48	0.42 ± 2.9	1.68 ± 0.24	2.66 ± 0.77
Stacked-TC -M2TD3	1.45 ± 0.38	$\mathbf{3.78 \pm 0.29}$	$\mathbf{5.2 \pm 0.29}$	-1.38 ± 1.67	$\mathbf{2.11 \pm 0.52}$	2.23 ± 0.63
Stacked-TC -RARL	1.52 ± 0.11	2.29 ± 0.23	2.91 ± 0.67	1.14 ± 2.19	1.21 ± 0.46	1.81 ± 0.73
TC -M2TD3	1.6 ± 0.06	3.71 ± 0.24	4.4 ± 0.6	$\mathbf{3.28 \pm 2.52}$	1.56 ± 0.23	$\mathbf{2.91 \pm 0.73}$
TC -RARL	$\mathbf{1.67 \pm 0.07}$	2.27 ± 0.22	1.79 ± 0.53	0.89 ± 2.19	1.01 ± 0.21	1.53 ± 0.64
TD3	0.0 ± 0.49	0.0 ± 0.22	0.0 ± 0.83	0.0 ± 1.36	0.0 ± 0.51	0.0 ± 0.68
DR	1.65 ± 0.05	2.31 ± 0.27	2.08 ± 0.49	1.15 ± 2.47	1.22 ± 0.34	1.68 ± 0.72
M2TD3	1.0 ± 0.11	1.0 ± 0.19	1.0 ± 0.55	1.0 ± 1.43	1.0 ± 0.65	1.0 ± 0.59
RARL	0.69 ± 0.13	-1.3 ± 0.54	-0.99 ± 0.11	0.47 ± 1.92	-0.35 ± 0.83	-0.3 ± 0.71

6 Some Theoretical properties of TC-MDPS.

6.1 On the optimal policy of TC

Following Lemma 3.3 of [1], it is known that in the rectangular case, there exists an optimal policy of the adversary that is stationary, provided that the actor policy is stationary. The TC-RMDP definition enforces a limitation on the temporal variation of the transition kernel. Consequently, all stationary adversarial policies are constrained by this stipulation. In turn, this guarantees that (under the hypothesis of sa-rectangularity) there always exists a solution to the TC-RMDP that is also a solution to the original RMDP. In other words: optimizing policies for TC-RMDPs do not exclude optimal solutions to the underlying RMDP. This sheds an interesting light on the search for robust optimal policies, since TC-RMDPs shrink the search space of optimal adversarial policies. In practice, this is confirmed by the previous experimental results (Figure 2) where the optimal agent policy found by either Oracle-TC , Stacked-TC , or vanilla TC actually outperforms the one found by M2TD3 or RARL in the non time-constrained setting.

6.2 Some Lipchitz-properties for non-stationary TC-RMPDS.

In this subsection we slightly depart from the framework defined in Section 2 and study the smoothness of the robust objective for vanilla TC or Stacked-TC . Th. 2.1 is no longer applicable as ψ is not observed. However, we can still give smoothness of the objective starting from Lipchichz conditions

on the evolution of the parameter that leads to smoothness on reward and transition kernel in the following definition 6.1.

Definition 6.1 (Reward/Kernel Lipchitz TC-RMDPs [13])**.** *We say that a parametric RDMPs is time constrained if the parameter change is bounded through time ie.* $\|\psi_{t+1} - \psi_t\| \leq L$*. Moreover, we assume that this variation in parameter implies a variation in the reward and transition kernel of*

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \|P_t(\cdot \mid s,a) - P_{t+1}(\cdot \mid s,a)\|_1 \leq L_P \quad ; |r_t(s,a) - \mathbb{E}\left[r_{t+1}(s,a)\right]| \leq L_r$$

From a theoretical point of view, a TC-RMDP can be seen as a sequence of stationary MDPs with time indexed reward and transition kernel r_t, P_t that have continuity. More formally for $M_t = (S, A, \Psi, p_{\psi_t}, r_t, L = (L_p, L_r))$, we can then define the sequence of stationary MDPs with Lipchitz variation :

$$\mathcal{M}_t^L = \Big\{ \{M_t\}_{t'=t_0}^t ; \exists L_r \in \mathbb{R} \forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \|P_t'(\cdot \mid s,a) - P_{t'+1}(\cdot \mid s,a)\|_1 \leq L_P \quad ; \\ |r_{t'}(s,a) - r_{t'+1}(s,a)| \leq L_r \Big\} \tag{1}$$

Defining r_t^k as the random variable corresponding to the reward function at time step t for stationary MDPs, but iterating with index k, the stationary rollout return at time t is $G(\pi, M_t) = \sum_{k \geq 0} \gamma^k r_t^k$. Assuming that at a fixed t the reward and transition kernel r_t, P_t are fixed, the robust objective function is:

$$J^R(\pi, t) := \min_{m = \{m_t'\}_{t'=t_0}^t \in \mathcal{M}_t^L} \mathbb{E}\left[G\left(\pi, m\right)\right]$$

This leads to the following guarantee for vanilla `TC` and `Stacked-TC` algorithms.

Theorem 6.2. *Assume TC-RMPDS with* $L = (L_r, L_P)$ *smoothness. Then* $\forall t \in \mathbb{N}, r_t \in [0,1]$*,*

$$\forall t \in \mathbb{N}^+, \forall t_0 \in \mathbb{N}^+, \quad |J^R(\pi, t_0) - J^R(\pi, t_0 + t)| \leq L' t.$$

with $L' := \left(\frac{\gamma}{(1-\gamma)^2} L_P + \frac{1}{1-\gamma} L_r \right)$

This theorem states that a small variation of the Kernel and reward function will not affect too much the robust objective. In other terms, despite the fact that the TC Bellman operator may not admit a fixed point and yield a non-stationary sequence of value functions, variations of the expected return remain bounded. Proof of the Th. 6.2 can be found in Appendix C.

7 Conclusion

This paper presents a novel framework for robust reinforcement learning, which addresses the limitations of traditional methods that rely on rectangularity assumptions. These assumptions often result in overly conservative policies, which are not suitable for real-world applications where environmental disturbances are multifactorial, correlated, and time-constrained. In order to overcome these challenges, we proposed a new formulation, the Time-Constrained Robust Markov Decision Process (TC-RMDP). The TC-RMDP framework is capable of accurately capturing the dynamics of real-world environments, due to its consideration of the temporal continuity and correlation of disturbances. This approach resulted in the development of three algorithms: The three algorithms, `Oracle-TC` , `Stacked-TC` , vanilla `TC` which differ in the extent to which environmental information is incorporated into the decision-making process. A comprehensive evaluation of continuous control benchmarks using MuJoCo environments has demonstrated that the proposed TC-RMDP algorithms outperform traditional robust RL methods and domain randomization techniques. These algorithms achieved a superior balance between performance and robustness in both time-constrained and static settings. The results confirmed the effectiveness of the TC-RMDP framework in reducing the conservatism of policies while maintaining robustness. Moreover, we provided theoretical guaranties for `Oracle-TC` in Th. 2.1 and for `Stacked-TC` and vanilla `TC` in Th. 6.2. This study contributes to the field of robust reinforcement learning by introducing a time-constrained framework that more accurately reflects the dynamics observed in real-world settings. The proposed algorithms and theoretical contributions offer new avenues for the development of more effective and practical RL applications in environments with complex, time-constrained uncertainties.

References

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

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

[3] Vineet Goyal and Julien Grand-Clement. Robust markov decision process: Beyond rectangularity. *arXiv preprint arXiv:1811.00215*, 2018.

[4] Mengmeng Li, Tobias Sutter, and Daniel Kuhn. Policy gradient algorithms for robust mdps with non-rectangular uncertainty sets. *arXiv preprint arXiv:2305.19004*, 2023.

[5] 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.

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

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

[8] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[9] 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.

[10] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

[11] Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. Experts in a markov decision process. *Advances in neural information processing systems*, 17, 2004.

[12] Yasin Abbasi Yadkori, Peter L Bartlett, Varun Kanade, Yevgeny Seldin, and Csaba Szepesvári. Online learning in markov decision processes with adversarially chosen transition probability distributions. *Advances in neural information processing systems*, 26, 2013.

[13] 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.

[14] Pratik Gajane, Ronald Ortner, and Peter Auer. A sliding-window algorithm for Markov decision processes with arbitrarily changing rewards and transitions. *arXiv preprint arXiv:1805.10066*, 2018.

[15] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Reinforcement learning under drift. *arXiv preprint arXiv:1906.02922*, 2019.

[16] Nicholas K Jong and Peter Stone. Bayesian models of nonstationary Markov decision processes. *Planning and Learning in A Priori Unknown or Dynamic Domains*, page 132, 2005.

[17] Melkior Ornik and Ufuk Topcu. Learning and planning for time-varying mdps using maximum likelihood estimation. *arXiv preprint arXiv:1911.12976*, 2019.

[18] 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.

[19] 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.

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

[21] 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.

[22] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[23] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[24] Yongyuan Liang, Yanchao Sun, Ruijie Zheng, Xiangyu Liu, Tuomas Sandholm, Furong Huang, and Stephen Marcus McAleer. Game-theoretic robust reinforcement learning handles temporally-coupled perturbations. In *The Second Workshop on New Frontiers in Adversarial Machine Learning*, 2023.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] Mohammed Amin Abdullah, Hang Ren, Haitham Bou Ammar, Vladimir Milenkovic, Rui Luo, Mingtian Zhang, and Jun Wang. Wasserstein robust reinforcement learning. *arXiv preprint arXiv:1907.13196*, 2019.

[30] Pierre Clavier, Stéphanie Allassonière, and Erwan Le Pennec. Robust reinforcement learning with distributional risk-averse formulation. *arXiv preprint arXiv:2206.06841*, 2022.

[31] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[32] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

A Appendix

The Appendix is structured as follow :

- In Appendix B, proof for fix point of `Oracle-TC` algorithm for can be found.
- In Appendix C, proof for algorithm Vanilla `TC` and `Stacked-TC` can found about robust objective.
- In Appendix F, the adversary training was sanity-checked within the time-constrained evaluation.
- In Appendix E, all implementation details are provided.
- In Appendix H, all raw results are presented.
- In Appendix I, the computer resources and training wall clock time are detailed.
- In Appendix J, the broader impact and limitations are discussed.

B Proof of Theorem 2.1

Proof. The Proof is similar to [1], using the fact that $p_{\psi+b}$ belongs to the simplex, we get contraction of the operator and convergence to a fix point v_B^*. Not that to converge to the fix point, there is no need of rectangularity. □

Recall the recursion

$$v_{n+1}(s,\psi) = \max_{\pi(s,\psi)\in\Delta_A} \min_{b\in B} T_b^\pi v_n(s,\psi) := \max_{\pi(s,\psi)\in\Delta_A} \min_{b\in B} \mathbb{E}_{a\sim\pi(s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p_{\psi+b}} v_n(s',\psi')] \tag{2}$$

First we prove that the TC Robust Operator T_B^π is a contraction. Let $V_1, V_2 \in \mathbb{R}^n$. Fix $s \in S$, and assume that $T_B^\pi V_1(s,\psi) \geq T_B^\pi V_2(s,\psi)$. Then fix $\epsilon > 0$ and pick π s.t given $s \in S$,

$$\inf_{b\in B} \mathbb{E}_{p_{\psi+b}} \left[r\left(s,\pi(s)\right) + \gamma V_1\left(s',\psi'\right)\right] \geq T_B^\pi V_1(s,\psi') - \epsilon. \tag{3}$$

First we pick a probability measure p' such that $p' = p_{\psi+b}, b \in B$, such that

$$\mathbb{E}_{p'} \left[r\left(s,\pi(s)\right) + \gamma V_2\left(s',\psi'\right)\right] \leq \inf_{b\in B} \mathbb{E}_{p'} \left[r\left(s,\pi(s)\right) + \gamma V_2\left(s',\psi'\right)\right] + \epsilon. \tag{4}$$

Then it lead to

$$\begin{aligned}
0 \leq T_B^\pi V_1(s,\psi) - T_B^\pi V_2(s,\psi) &\leq \left(\inf_{p\in B} \mathbb{E}_p \left[r\left(s,\pi(s)\right) + \gamma V_1\left(s',\psi'\right)\right] + \epsilon\right) && (5)\\
&- \left(\inf_{p\in B} \mathbb{E}_p \left[r\left(s,\pi(s)\right) + \gamma V_2\left(s',\psi'\right)\right]\right) && (6)\\
&\leq \left(\mathbb{E}_{p'} \left[r\left(s,\pi(s)\right) + \gamma V_1\left(s',\psi'\right)\right] + \epsilon\right) - && (7)\\
&\left(\mathbb{E}_{p'} \left[r\left(s,\pi(s)\right) + \gamma V_2\left(s',\psi'\right)\right] - \epsilon\right), && (8)\\
&= \gamma\mathbb{E}_{p'}[V_1 - V_2] + 2\epsilon, && (9)\\
&\leq \gamma\mathbb{E}_{p'}|V_1 - V_2| + 2\epsilon && (10)\\
&\leq \gamma\|V_1 - V_2\|_\infty + 2\epsilon. && (11)
\end{aligned}$$

where last inequality is Holder's inequality between L_1 and L_∞ norms, use probability measure in the simplex such as $\|p'\|_1 = 1$. Doing the same thing but in the case where $T_B^\pi V_1(s) \leq T_B^\pi V_2(s)$, it holds

$$\forall s \in S, |T_B^\pi V_1(s) - T_B^\pi V_2(s)| \leq \gamma\|V_1 - V_2\|_\infty + 2\epsilon, \tag{12}$$

i.e. $\|T_B^\pi V_1 - T_B^\pi V_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty + 2\epsilon$. As we can choose ϵ arbitrary small, this establishes that the TC Bellman operator is a γ-contraction. Since T_B^π is a contraction operator on a Banach space, the Banach fixed point theorem implies that the operator equation $T_B^\pi V = V$ has a unique solution $V = v_B^\pi$. A similar proof can be done for optimal operator T_B^*. The only difference is the maximum operator which is $1-$Lipschitz. So T_B^* is also a contraction. Then, once proved that operators are $\gamma-$ contraction, following [1] (Th. 5), we have that the fixed point of this recursion is exactly :

$$v_B^\pi(s,\psi) := \min_{\substack{(b_t)_{t\in\mathbb{N}},\\ b_t\in B}} \mathbb{E}\Big[\sum \gamma^t r_t | \psi_{-1} = \psi, s_0 = s, b_t \in B, \psi_t = \psi_{t-1} + b_t, a \sim \pi, s_t \sim p_{\psi_t}\Big], \tag{13}$$

$$v_B^*(s,\psi) = \max_{\pi(s,\psi)\in\Delta_A} v_B^\pi(s,\psi). \tag{14}$$

for (optimal) TC Bellman Operator.

C Guaranties for non-stationary Robust MDPS

Recall that we represent a non-stationary robust MDPs (NS-RMDP) as a stochastic sequence, $\{\mathcal{M} = \{M_i\}_{t=t_0}^\infty$, of stationary MDPs $M_t \in \mathcal{M}$, where $\mathcal{M}$ is the set of all stationary MDPs. Each M_t is a tuple, $\big(\mathcal{S}, \mathcal{A}, P_t, r_t, \gamma, \rho^0\big)$, where $\mathcal{S}$ The set of possible states is denoted by $\mathcal{S}$, the set of actions by $\mathcal{A}$, the discounting factor by γ, the start-state distribution by ρ^0, and the reward distribution by r_t. The reward distribution, denoted by $r_t : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$, is the probability distribution of rewards. The transition function, represented by $P_t : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, is the probability distribution of transitions between states. The symbol Δ denotes the simplex. For all $M_t \in \mathcal{M}$, we assume that the state space, action space, discount factor, and initial distribution remain fixed. A policy is represented as a function $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. In general, we will use subscripts t to denote the time evolution during an episode and superscripts k to denote the time step assuming reward or kernel t which is stationary, assuming that the reward function is not changing as it is at time step t stationary. That r_t^k is the random variables corresponding to the state, action, and reward at time step t for stationary, but iterating with index k.

Definition C.1 (Lipschitz of sequence of MDPs)**.** *We denote the sequence of kernel and reward function $\mathcal{P} = \{P_t\}_{t=t_0}^\infty$ and $\mathcal{R} = \{r_t\}_{t=t_0}^\infty$. We define a sequence of MDP is $L = (L_r, L_P)$-Lipchitz if $m = \{m_t\}_{t=t_0}^\infty \in \mathcal{M}^L$ with*

$$\mathcal{M}_t^L = \Big\{ \{M_t\}_{t'=t_0}^t ; \exists (L_r, L_P) \in \mathbb{R}_+^2 \forall t \in \mathbb{N}, \forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \|P_{t'}(\cdot \mid s,a) - P_{t'+1}(\cdot \mid s,a)\|_1 \leq L_P \\ ; |r_t'(s,a) - r_{t'+1}(s,a)| \leq L_r \Big\}$$

Assuming that for a time steps the reward function is stationary, we can compute the average return as:

Definition C.2. *Non-robust objective function, assuming that $G(\pi, M_t) = \sum_{k\geq 0} \gamma^k r_t^k$, the return is we assume stationary with reward function r_t*

$$J(\pi, t) = \mathbb{E}[G(\pi, M_t)] = (1-\gamma)^{-1} \sum_{s\in\mathcal{S}} d^\pi(s, M_t) \sum_{a\in\mathcal{A}} \pi(a \mid s) r_t(s,a). \tag{15}$$

with d^π the state occupancy measure defined in (16)*.*

Definition C.3 (Robust (optimal) Return of NS-RMDPs)**.** *Let a return of π for any $m_t \in M_t$ be $G(\pi, M_t) := \sum_{k=0}^\infty \gamma^k r_t^k$ with kernel transition P_t following π, with $\forall k, t, r_t^k \in [0,1]$, and the Robust non-stationary expected return with variation of kernel*

Let the robust performance of π for episode t be

$$J^R(\pi, t) := \min_{m = \{m_t'\}_{t'=t_0}^t \in \mathcal{M}_t^L} \mathbb{E}\left[G(\pi, m)\right]$$

D Proof Theom 6.2

$$\forall t \in \mathbb{N}^+, \forall t_0 \in \mathbb{N}^+, \quad |J^R(\pi, t_0) - J^R(\pi, t_0 + t)| \leq L' t.$$

with $L' := \left(\frac{\gamma}{(1-\gamma)^2} L_P + \frac{1}{1-\gamma} L_r\right)$

Proof of Theorem 6.2. First, this difference can be upper bounded in the non robust case as:
By definition, we can rewrite non-robust objective function and occupancy measure as.

$$d^\pi(s, M_t) = (1-\gamma) \sum_{k=0}^{\infty} \gamma^k \Pr(S_t = s \mid \pi, M_t), \tag{16}$$

$$J(\pi, M_t) = (1-\gamma)^{-1} \sum_{s \in \mathcal{S}} d^\pi(s, M_t) \sum_{a \in \mathcal{A}} \pi(a \mid s) r_t(s, a). \tag{17}$$

First, we can decompose the problem into sub-problems such that

$$\forall t \in \mathbb{N}^+, \forall t_0 \in \mathbb{N}^+, \quad |J(\pi, t_0) - J(\pi, t_0 + t)| \leq |\sum_{t'=t_0}^{t_0+t-1} |J(\pi, M_{t'}) - J(\pi, M_{t'+1})| \tag{18}$$

using triangular inequality. Looking at differences between two time steps:

$$\begin{aligned}
&(1-\gamma)|J(\pi, M_t) - J(\pi, M_{t+1})| \\
&= \left| \sum_{s \in \mathcal{S}} d^\pi(s, M_t) \sum_{a \in \mathcal{A}} \pi(a \mid s) r_t(s, a) - \sum_{s \in \mathcal{S}} d^\pi(s, M_{t+1}) \sum_{a \in \mathcal{A}} \pi(a \mid s) r_{t+1}(s, a) \right| \\
&= \left| \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(d^\pi(s, M_t) r_t(s, a) - d^\pi(s, M_{t+1}) r_{t+1}(s, a) \right) \right| \\
&= \left| \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(d^\pi(s, M_t) \left(r_{t+1}(s, a) + (r_t(s, a) - R_{t+1}(s, a)) \right) - d^\pi(s, M_{t+1}) r_{t+1}(s, a) \right) \right| \\
&= \Big| \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(d^\pi(s, M_t) - d^\pi(s, M_{t+1}) \right) r_{t+1}(s, a) \\
&\quad + \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a \mid s) d^\pi(s, M_t) \left(r_t(s, a) - r_{t+1}(s, a) \right) \Big| \\
&\overset{(a)}{\leq} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a \mid s) \left| d^\pi(s, M_t) - d^\pi(s, M_{t+1}) \right| |r_{t+1}(s, a)| \\
&\quad + \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a \mid s) d^\pi(s, M_t) |r_t(s, a) - r_{t+1}(s, a)| \\
&\overset{(b)}{\leq} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a \mid s) \left| d^\pi(s, M_t) - d^\pi(s, M_{t+1}) \right| + L_R \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a \mid s) d^\pi(s, M_t) \\
&= \sum_{s \in \mathcal{S}} \left| d^\pi(s, M_t) - d^\pi(s, M_{t+1}) \right| + L_r
\end{aligned}$$

where (a) is triangular inequality, (b) is definition of of supremum of reward in the assumptions and reward bounded by 1. Then, let $P_t^\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ be the transition matrix (s' in rows and s in columns) resulting due to π and P_t, i.e., $\forall t, P_t^\pi(s', s) := \Pr(S_{t+1} = s' \mid S_t = s, \pi, M_t)$, and let $d^\pi(\cdot, M_t) \in \mathbb{R}^{|\mathcal{S}|}$ denote the vector of probabilities for each state, then Finally we can easily bound the difference of occupation measure as :

$$\sum_{s\in\mathcal{S}} \left|d^{\pi}\left(s, M_t\right) - d^{\pi}\left(s, M_{t+1}\right)\right| \tag{19}$$

$$\overset{(d)}{\leq} \gamma(1-\gamma)^{-1} \sum_{s'\in\mathcal{S}} \left| \sum_{s\in\mathcal{S}} \left(P_t^{\pi}\left(s', s\right) - P_{t+1}^{\pi}\left(s', s\right)\right) d^{\pi}\left(s, M_t\right) \right| \tag{20}$$

$$\leq \gamma(1-\gamma)^{-1} \sum_{s'\in\mathcal{S}} \sum_{s\in\mathcal{S}} \left|P_t^{\pi}\left(s', s\right) - P_{t+1}^{\pi}\left(s', s\right)\right| d^{\pi}\left(s, M_t\right) \tag{21}$$

$$= \gamma(1-\gamma)^{-1} \sum_{s'\in\mathcal{S}} \sum_{s\in\mathcal{S}} \left| \sum_{a\in\mathcal{A}} \pi(a \mid s) \left(\Pr\left(s' \mid s, a, M_t\right) - \Pr\left(s' \mid s, a, M_{t+1}\right)\right) \right| d^{\pi}\left(s, M_t\right) \tag{22}$$

$$\leq \gamma(1-\gamma)^{-1} \sum_{s'\in\mathcal{S}} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \pi(a \mid s) \left|\Pr\left(s' \mid s, a, M_t\right) - \Pr\left(s' \mid s, a, M_{t+1}\right)\right| d^{\pi}\left(s, M_t\right) \tag{23}$$

$$= \gamma(1-\gamma)^{-1} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \pi(a \mid s) d^{\pi}\left(s, M_t\right) \sum_{s'\in\mathcal{S}} \left|\Pr\left(s' \mid s, a, M_t\right) - \Pr\left(s' \mid s, a, M_{t+1}\right)\right| \tag{24}$$

$$\leq \gamma(1-\gamma)^{-1} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \pi(a \mid s) d^{\pi}\left(s, M_t\right) L_P \tag{25}$$

$$= \frac{\gamma L_P}{(1-\gamma)}, \tag{26}$$

which gives regrouping all terms:

$$\left|J(\pi, M_t) - J(\pi, M_{t+1})\right| \leq \frac{L_r}{1-\gamma} + \frac{\gamma L_P}{(1-\gamma)^2}. \tag{27}$$

where the stationary MDP M_{t+1} can be chosen as the minimum over the previous MDPs at time step t such as $\left|\Pr\left(s' \mid s, a, M_t\right) - \Pr\left(s' \mid s, a, M_{t+1}\right)\right| \leq L_p$. Rewriting previous equation (27), it holds that

$$\left| \left[\mathbb{E}_{\pi,P}\left[G\left(\pi, m\right)\right] - \min_{m=\{m'_t\}_{t'=t}^{t+1}} \mathbb{E}_{\pi,P}\left[G\left(\pi, m\right)\right]\right] \right| \leq \frac{L_r}{1-\gamma} + \frac{\gamma L_P}{(1-\gamma)^2} = L'. \tag{28}$$

Now considering non robust objective :

$$\left|J^R\left(\pi, t\right) - J^R\left(\pi, t+1\right)\right| \tag{29}$$

$$= \left| \min_{m=\{m'_t\}_{t'=t_0}^{t} \in \mathcal{M}^L} \mathbb{E}\left[G\left(\pi, m\right)\right] - \min_{m=\{m'_t\}_{t'=t_0}^{t+1} \in \mathcal{M}^L_{t+1}} \mathbb{E}\left[G\left(\pi, m\right)\right] \right| \tag{30}$$

$$= \left| \min_{m=\{m'_t\}_{t'=t_0}^{t} \in \mathcal{M}^L_t} \left[\mathbb{E}\left[G\left(\pi, m\right)\right] - \min_{m=\{m'_t\}_{t'=t_0}^{t} \in \mathcal{M}^L_t} \min_{m=\{m'_t\}_{t'=t}^{t+1}} \mathbb{E}\left[G\left(\pi, m\right)\right]\right] \right| \tag{31}$$

$$\leq \max_{m=\{m_t\}_{t=t_0}^{t} \in \mathcal{M}^L_t} \left| \left[\mathbb{E}\left[G\left(\pi, m\right)\right] - \min_{m=\{m'_t\}_{t'=t}^{t+1}} \mathbb{E}\left[G\left(\pi, m\right)\right]\right] \right| \tag{32}$$

where first equality is the definition of the robust objective, second equality is decomposition of minimum across time steps and final inequality is simply a property of the $\min$ such as $|\min a - \min b| \leq \sup |a - b|$.

Finally plugging 28 in (32), it holds that

$$\left|J^R(\pi,t) - J^R(\pi,t+1)\right| \tag{33}$$

$$= \left| \min_{m=\{m'_t\}_{t'=t_0}^{t} \in \mathcal{M}^L} \mathbb{E}_{\pi,P}\left[G(\pi,m)\right] - \min_{m=\{m'_t\}_{t'=t_0}^{t+1} \in \mathcal{M}^L} \mathbb{E}_{\pi,P}\left[G(\pi,m)\right] \right| \leq \frac{L_r}{1-\gamma} + \frac{\gamma L_P}{(1-\gamma)^2}. \tag{34}$$

$$:= L'. \tag{35}$$

Combining t times the previous equation gives the result:

$$\forall t \in \mathbb{N}^+, \forall t_0 \in \mathbb{N}^+, \quad |J^R(\pi,t_0) - J^R(\pi,t_0+t)| \leq L't.$$

with $L' := \left(\frac{\gamma}{(1-\gamma)^2} L_P + \frac{1}{1-\gamma} L_r\right)$ □

E Implementation details

E.1 Algorithm

Algorithm 2 Time-constrained robust training

Input: Time-constrained MDP: $(S, A, \Psi, p_\psi, r, L)$, Agent π, Adversary $\bar{\pi}$

for *each interaction time step* t **do**

- $a_t \sim \pi_t(s_t, \psi_t)$ // Sample an action with Oracle-TC
- $a_t \sim \pi_t(s_t, a_{t-1}, s_{t-1})$ // Sample an action with Stacked-TC
- $a_t \sim \pi_t(s_t)$ // Sample an action with TC
- $\psi_{t+1} \sim \bar{\pi}_\phi(s_t, a_t, \psi_t)$ // Sample the worst TC parameter
- $s_{t+1} \sim p_{\psi_{t+1}}(s_t, a_t)$ // Sample a transition
- $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r(s_t, a_t), \psi_t, \psi_{t+1}, s_{t+1})\}$ // Add transition to replay buffer
- $\{s_i, a_i, r(s_i, a_i), \psi_i, \psi_{i+1}, s_{i+1}\}_{i \in [1,N]} \sim \mathcal{B}$ // Sample a mini-batch of transitions
- $\theta_c \leftarrow \theta_c - \alpha \nabla_{\theta_c} L_Q(\theta_c)$ // Critic update phase
- $\theta_a \leftarrow \theta_a - \alpha \nabla_{\theta_a} L_\pi(\theta_a)$ // Actor update
- $\phi_c \leftarrow \phi_c + \alpha \nabla_{\phi_c} L_{\bar{Q}}(\phi_c)$ // Adversary Critic update phase
- $\phi_a \leftarrow \phi_a + \alpha \nabla_{\phi_a} L_{\bar{\pi}}(\phi_a)$ // Adversary update

Note that in Time-constrained robust training Algorithm in section E.1, L_Q and L_π are as defined by [31] double critics and target network updates are omitted here for clarity

In Table 4, for the stack algorithm, s_i is defined as $s_i \leftarrow s_i \cup s_{i-1} \cup a_{i-1}$ for Stacked-TC , and for the Oracle-TC version, $s_i \leftarrow s_i \cup \psi_i$.

Loss Function	Equation
$L_{Q_{\theta_c}}$ (TC-RARL)	$\mathbb{E}\left[Q_{\theta_c}(s_i,a_i) - r(s_i,a_i) + \gamma \min_{j=1,2} Q_{\theta_c}(s_{i+1}, \pi(s_{i+1}))\right]$
$L_\pi(\theta_a)$ (TC-RARL)	$-\mathbb{E}\left[Q_{\theta_c}(s_i, \pi_{\theta_a}(s_i))\right]$
$L_{\bar{\pi}}(\theta_a)$ (TC-RARL)	$\mathbb{E}\left[\bar{Q}_{\theta_c}(s_i, a_i, \bar{\pi}(s_i,a_i), \psi_i)\right]$
$L_{\bar{Q}}(\theta_c)$ (TC-RARL)	$\mathbb{E}\left[\bar{Q}_{\theta_c}(s_i,a_i) - r(s_i,a_i) + \gamma \min_{j=1,2} \bar{Q}_{\theta_c}(s_{i+1}, \pi_{\theta_a}(s_{i+1}), \bar{\pi}_{\theta_a}(s_{i+1}, a_{i+1}, \psi_{i+1}))\right]$
$L_{Q_{\theta_c}}$ Shared (TC-M2TD3)	$\mathbb{E}\left[Q_{\theta_c}(s_i,a_i) - r(s_i,a_i) + \gamma \min_{j=1,2} Q_{\theta_c}(s_{i+1}, \pi_{\theta_a}(s_{i+1}), \bar{\pi}_{\theta_a}(s_{i+1}, a_{i+1}, \psi_{i+1}))\right]$
$L_\pi(\theta_a)$ (TC-M2TD3)	$\mathbb{E}\left[Q_{\theta_c}(s_i, a_i, \bar{\pi}_{\theta_a}(s_i,a_i), \psi_i)\right]$
$L_{\bar{\pi}}(\theta_a)$ (TC-M2TD3)	$-\mathbb{E}\left[\bar{Q}_{\theta_c}(s_i, a_i, \bar{\pi}_{\theta_a}(s_i,a_i,\psi_i))\right]$

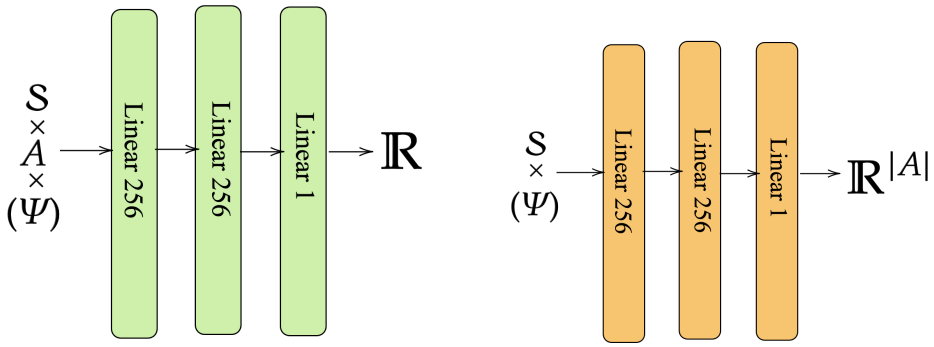
Table 4: Summary of Loss Functions for TD3 in TC-RARL and TC-M2TD3

E.2 Neural network architecture

We employ a consistent neural network architecture for both the baseline and our proposed methods for the actor and the critic components. The architecture's design ensures uniformity and comparability across different models.

The critic network is structured with three layers, as depicted in Figure 3a, the critic begins with an input layer that takes the state and action as inputs, which then passes through two fully connected linear layers of 256 units each. The final layer is a single linear unit that outputs a real-valued function, representing the estimated value of the state-action pair.

The actor neural network, shown in Figure 3b, also utilizes a three-layer design. It begins with an input layer that accepts the state as input. This is followed by two linear layers, each consisting of 256 units. The output layer of the actor neural network has a dimensionality equal to the number of dimensions of the action space.



(a) Critic neural network architecture

(b) Actor neural network architecture

Figure 3: Actor critic neural network architecture

E.3 M2TD3

We utilized the official M2TD3 [18] implementation provided by the original authors, accessible via the GitHub repository for M2TD3 and Oracle M2TD3.

For the TC-M2TD3 or variants, we implemented the M2TD3 algorithm as specified. To simplify our approach, we omitted the implementation of the multiple $\hat{\psi}$ network and the system for resetting $\hat{\psi}$. We replace with an adversary which $\bar{\pi} : \mathcal{S} \times \mathcal{A} \times \Psi \to \Psi$ which minimize $Q(s, a, \psi)$.

E.4 TD3

We adopted the TD3 implementation from the CleanRL library, as detailed in [32].

F Sanity check on the adversary training in the time-constrained evaluation

A natural question arises regarding the worst time-constrained perturbation. Whether we adequately trained the adversary in the loop, or its suboptimal performance might lead to overestimating the trained agent reward against the worst-case time-constrained perturbation. We monitored the adversary's performance during its training against a fixed agent to address this. The attached figure shows the episodic reward (from the agent's perspective) during the adversary's training over 5 million timesteps, with a perturbation radius of $L = 0.001$. Each curve is an average of over 10 seeds. The plots show a rapid decline in reward during the initial stages of training, followed by quick stabilization. The episodic reward stabilizes early in the Ant (Figure 4a) environment, indicating

Hyperparameter	Default Value
Policy Std Rate	0.1
Policy Noise Rate	0.2
Noise Clip Policy Rate	0.5
Noise Clip Omega Rate	0.5
Omega Std Rate	1.0
Min Omega Std Rate	0.1
Maximum Steps	5e6
Batch Size	100
Hatomega Number	5
Replay Size	1e6
Policy Hidden Size	256
Critic Hidden Size	256
Policy Learning Rate	3e-4
Critic Learning Rate	3e-4
Policy Frequency	2
Gamma	0.99
Polyak	5e-3
Hatomega Parameter Distance	0.1
Minimum Probability	5e-2
Hatomega Learning Rate (ho_lr)	3e-4
Optimizer	Adam

Table 5: Hyperparameters for the M2TD3 Agent

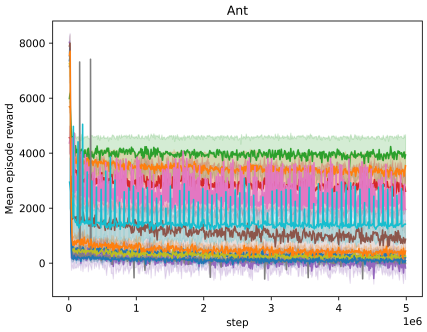
Hyperparameter	Default Value
Maximum Steps	5e6
Buffer Size	1×10^6
Learning Rate	3×10^{-4}
Gamma	0.99
Tau	0.005
Policy Noise	0.2
Exploration Noise	0.1
Learning Starts	2.5×10^4
Policy Frequency	2
Batch Size	256
Noise Clip	0.5
Action Min	-1
Action Max	1
Optimizer	Adam

Table 6: Hyperparameters for the TD3 Agent

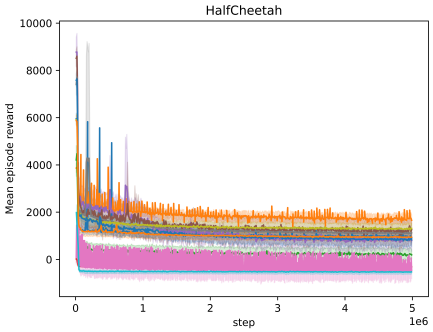
quick convergence. Similarly, in the HalfCheetah (Figure 4b) environment, the reward shows a sharp initial decline and stabilizes, suggesting effective training. For Hopper (Figure 4c), the reward decreases and then levels off, reflecting adversary convergence. Although the reward is more variable in the HumanoidStandup (Figure 4d) environment, it ultimately reaches a steady state, confirming adequate training. Finally, in the Walker environment, the reward pattern demonstrates a quick drop followed by stabilization, indicating convergence. These observations confirm that the adversaries were not undertrained. The rapid convergence to a stable performance across all environments ensures the accuracy of the worst time-constrained perturbations estimated during training.

G Uncertainty set in MuJoCo environments

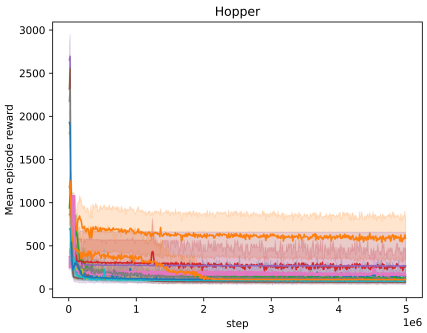
The experiments of Section 5 follow the evaluation protocol proposed by [18] and based on MuJoCo environments [5]. These environments are designed with a 3D uncertainty sets. Table 7 lists all environments evaluated and their uncertainty sets. The uncertainty sets column defines the ranges of



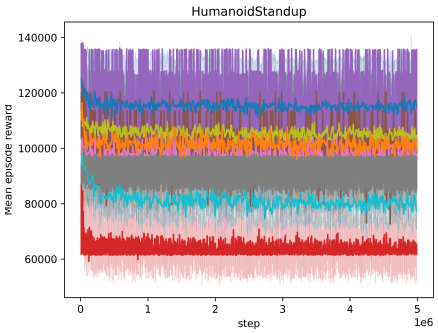
(a) Ant: Episodic reward of the trained agent during adversary training



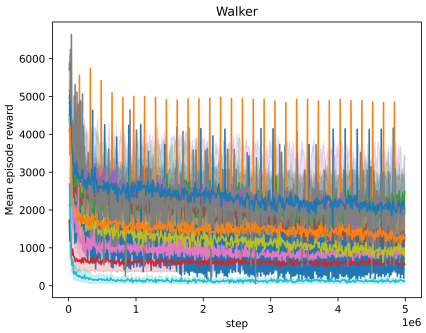
(b) HalfCheetah: Episodic reward of the trained agent during adversary training



(c) Hopper: Episodic reward of the trained agent during adversary training



(d) HumanoidStandup: Episodic reward of the trained agent during adversary training



(e) Walker: Episodic reward of the trained agent during adversary training

(f) Legend for algorithm

Figure 4: Episodic reward of the trained agent during the training of the adversary across different environments. Each plot represents the performance over 5 million timesteps, with rewards averaged across 10 seeds. The perturbation radius is set to $L = 0.001$ for all adversaries.

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.

H Raw results

Table 8 reports the non-normalized time-constrained (with a radius of $L = 0.001$) worst-case scores, averaged across 10 independent runs for each benchmark. Table 9 reports the static worst case score obtained by each agent across a grid of environments, also averaged across 10 independent runs for

Table 7: List of environment and parameters for the experiments

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

Table 8: Avg. of time-constrained worst-case performance over 10 seeds for each method

Environment Method	Ant	HalfCheetah	Hopper	HumanoidStandup	Walker
Oracle M2TD3	5768 ± 395	3521 ± 187	1241 ± 125	116232 ± 1454	4559 ± 757
Oracle RARL	4387 ± 667	-50 ± 99	344 ± 113	68979 ± 10641	1811 ± 342
Oracle TC-M2TD3	7268 ± 704	7507 ± 284	$\mathbf{3386 \pm 323}$	114411 ± 16973	5344 ± 536
Oracle TC-RARL	$\mathbf{7534 \pm 781}$	$\mathbf{7526 \pm 311}$	3169 ± 311	101182 ± 12083	4783 ± 382
Stacked TC-M2TD3	6502 ± 450	6377 ± 517	3047 ± 394	85524 ± 11448	$\mathbf{5724 \pm 828}$
Stacked TC-RARL	6955 ± 690	5319 ± 223	1747 ± 153	107913 ± 5514	4152 ± 483
TC-M2TD3	7181 ± 591	6516 ± 232	2511 ± 45	$\mathbf{129183 \pm 9120}$	4964 ± 531
TC-RARL	7473 ± 361	4989 ± 284	1475 ± 158	108669 ± 17764	3971 ± 351
DR	7247 ± 925	4986 ± 363	1642 ± 104	109618 ± 11479	4380 ± 488
M2TD3	5622 ± 435	3671 ± 405	1120 ± 220	102839 ± 12987	4078 ± 644
RARL	4348 ± 574	382 ± 366	240 ± 104	106768 ± 4051	2388 ± 559
TD3	2259 ± 424	1808 ± 503	777 ± 407	104877 ± 12063	1893 ± 361

each benchmark. Table 10 reports the static average case score obtained by each agent across a grid of environments, also averaged across 10 independent runs for each benchmark.

H.1 Fixed adversary evaluation

At the beginning of each episode, $\psi_0 \sim \mathcal{U}(\Psi)$ is selected for every fixed adversary. The episode length is 1000 steps. To begin with, the random fixed adversary simulates stochastic changes. It selects a parameter ψ_t at each timestep within a radius of $L = 0.1$, which is 100 times larger than in our training methods. This tests the agents' adaptability to unexpected changes. In contrast, the cosine fixed adversary introduces deterministic changes using a cosine function. The radius

Table 9: Avg. of raw static worst-case performance over 10 seeds for each method

	Ant	HalfCheetah	Hopper	Humanoid	Walker
dr	19.78 ± 394.84	2211.48 ± 915.64	245.01 ± 167.21	64886.87 ± 30048.79	1318.36 ± 777.51
m2td3	2322.73 ± 649.3	2031.9 ± 409.7	273.6 ± 131.9	71900.97 ± 24317.35	2214.16 ± 1330.4
oracle m2td3	2370.93 ± 473.56	319.67 ± 599.26	267.41 ± 111.47	93123.84 ± 26696.17	736.59 ± 944.76
oracle rarl	1396.88 ± 777.46	-278.84 ± 54.36	167.5 ± 38.2	45635.24 ± 15974.44	459.74 ± 437.02
oracle tc m2td3	120.74 ± 618.23	4273.31 ± 246.91	168.7 ± 217.94	58687.26 ± 22321.77	710.99 ± 799.08
oracle tc rarl	1328.27 ± 890.49	3458.52 ± 893.22	150.54 ± 33.12	73276.78 ± 9110.33	1299.88 ± 812.63
rarl	960.11 ± 744.01	-211.8 ± 218.73	170.46 ± 45.73	67821.86 ± 21555.24	360.31 ± 186.06
stacked tc m2td3	-242.98 ± 212.98	3467.34 ± 418.64	289.37 ± 182.18	58515.04 ± 19186.25	2475.58 ± 1057.03
stacked tc rarl	37.77 ± 320.71	1414.37 ± 876.91	344.37 ± 190.1	77357.17 ± 18186.34	1518.86 ± 668.13
td3	-123.64 ± 824.35	-546.21 ± 158.81	69.3 ± 42.77	64577.24 ± 16606.51	114.41 ± 211.05
tc m2td3	-271.34 ± 191.15	3286.67 ± 603.14	333.36 ± 60.04	73428.2 ± 17879.28	2603.59 ± 706.63
tc rarl	209.04 ± 575.89	1738.59 ± 782.71	376.01 ± 155.4	74840.68 ± 33496.45	1513.65 ± 1239.3

Table 10: Avg. of raw static average case performance over 10 seeds for each method

env name algo-name	Ant	HalfCheetah	Hopper	HumanoidStandup	Walker
dr	7500.88 ± 143.38	6170.33 ± 442.57	1688.36 ± 225.59	110939.89 ± 22396.41	4611.24 ± 463.42
m2td3	5577.41 ± 316.95	4000.98 ± 314.76	1193.32 ± 254.9	109598.43 ± 12992.35	4311.2 ± 877.89
oracle m2td3	5958.21 ± 237.32	4930.18 ± 390.96	1249.62 ± 212.74	118273.54 ± 13891.06	4616.05 ± 407.94
oracle rarl	4684.83 ± 648.14	36.19 ± 216.52	380.39 ± 110.14	76920.58 ± 26135.3	1451.39 ± 1132.87
oracle-tc m2td3	7739.65 ± 254.65	9536.92 ± 429.14	3281.92 ± 61.79	119737.21 ± 12697.2	5442.85 ± 499.78
oracle-tc-rarl	7889.1 ± 56.0	9474.0 ± 341.69	3071.17 ± 220.39	104348.01 ± 26249.98	5220.2 ± 318.07
rarl	4650.55 ± 395.03	206.71 ± 887.25	276.37 ± 52.42	104764.87 ± 17400.85	2493.26 ± 1113.74
stacked tc m2td3	6912.76 ± 1116.81	8583.55 ± 479.97	3124.06 ± 133.27	88039.74 ± 15138.11	5809.54 ± 703.92
stacked-tc-rarl	7123.07 ± 332.33	6130.71 ± 384.05	2072.75 ± 306.48	110843.2 ± 19887.32	4596.79 ± 619.2
vanilla	2600.43 ± 1468.87	2350.58 ± 357.12	733.18 ± 382.06	100533.0 ± 12298.37	2965.47 ± 685.39
vanilla-tcm2td3	7366.9 ± 169.58	8467.64 ± 397.42	2756.5 ± 273.91	130305.38 ± 22865.1	5070.71 ± 315.7
vanilla-tc-rarl	7558.58 ± 198.37	6092.61 ± 365.68	1558.26 ± 242.17	108635.71 ± 19848.21	4325.42 ± 283.04

Environment Method	Ant	HalfCheetah	Hopper	HumanoidStandup	Walker
Oracle TC-M2TD3	7782 ± 915	$\mathbf{8805 \pm 165}$	$\mathbf{2365 \pm 199}$	116791 ± 12572	5148 ± 558
Oracle TC-RARL	$\mathbf{8041 \pm 470}$	8727 ± 227	2120 ± 96	107733 ± 11975	4896 ± 326
Oracle M2TD3	5830 ± 542	4445 ± 186	1222 ± 111	118861 ± 1365	4584 ± 787
Oracle RARL	4628 ± 514	-51 ± 60	370 ± 141	81583 ± 16526	1829 ± 356
Stacked TC-M2TD3	6888 ± 738	7400 ± 385	2114 ± 138	88436 ± 10750	$\mathbf{5278 \pm 845}$
Stacked TC-RARL	7045 ± 904	5992 ± 427	1940 ± 93	106213 ± 6770	4430 ± 389
TC-M2TD3	7156 ± 692	7530 ± 185	2157 ± 112	$\mathbf{129599 \pm 13556}$	4931 ± 568
TC-RARL	7554 ± 948	5751 ± 482	1445 ± 203	105144 ± 16813	4112 ± 329
DR	7572 ± 629	6048 ± 349	1416 ± 168	105677 ± 16333	4371 ± 431
M2TD3	5588 ± 516	4180 ± 70	1018 ± 271	107692 ± 10414	4176 ± 783
RARL	4347 ± 567	240 ± 250	390 ± 130	103583 ± 9217	1925 ± 501
TD3	4017 ± 518	2028 ± 1250	1944 ± 246	91205 ± 11350	2860 ± 419

Table 11: Avg. performance against time-constrained fixed random adversary with a radius $L = 0.1$ over 10 seeds for each method

of $L = 0.1$ scales the frequency of the cosine function, ensuring smooth and periodic variations. Additionally, a phase shift at the start of each episode ensures different starting points. Meanwhile, the linear fixed adversary employs a linear function. The parameters change linearly from the initial value to either one of a vertex of the uncertainty set Ψ over 1000 steps. Furthermore, the exponential fixed adversary uses an exponential function. Parameters change exponentially from the initial value to either of a vertex of the uncertainty set Ψ over 1000 steps. This ensures smooth and predictable variations. Similarly, the logarithmic fixed adversary uses a logarithmic function. Parameters change logarithmically from the initial value to either of a vertex of the uncertainty of the uncertainty set Ψ over 1000 steps, ensuring smooth and predictable variations. Agents trained under the time-constrained framework outperform all baselines in all environments for each fixed adversary, except when compared to the oracle TC method, which has access to ψ. In this case, the stacked-TC or TC methods outperform all baselines in all environments for the cosine, logarithmic, and exponential adversaries and outperform the fixed adversary baseline in 4 out of 5 instances for the random and linear fixed adversaries.

H.2 Agents training curve

We conducted training for each agent over a duration of 5 million steps, closely monitoring the cumulative rewards obtained over a trajectory spanning 1,000 steps. To enhance the reliability of our results, we averaged the performance curves across 10 different seeds. The graphs in Figures 5 to 15 illustrate how different training methods, including Domain Randomization, M2TD3, RARL, Oracle RARL ,Oracle M2TD3, TC RARL, TC M2TD3, Stacked TC RARL and Stacked TC M2TD3, impact agent performance across various environments.

Table 12: Avg. performance against time-constrained fixed cosine adversary with a radius $L = 0.1$ over 10 seeds for each method

Environment Method	Ant	HalfCheetah	Hopper	HumanoidStandup	Walker
Oracle M2TD3	5528 ± 637	3453 ± 266	1016 ± 48	119813 ± 3281	3589 ± 863
Oracle RARL	4550 ± 626	-79 ± 34	371 ± 140	74116 ± 7890	1593 ± 326
Oracle TC-M2TD3	$\mathbf{7586 \pm 1345}$	$\mathbf{8174 \pm 383}$	$\mathbf{1946 \pm 104}$	115506 ± 12470	4464 ± 781
Oracle TC-RARL	7522 ± 1435	7838 ± 810	1735 ± 138	110535 ± 12702	4442 ± 591
Stacked TC-M2TD3	6269 ± 849	7173 ± 509	1734 ± 157	88157 ± 10654	$\mathbf{4888 \pm 567}$
Stacked TC-RARL	6510 ± 1395	5385 ± 445	1519 ± 118	105696 ± 5243	3848 ± 404
TC-M2TD3	6350 ± 769	6797 ± 609	1413 ± 167	$\mathbf{130892 \pm 11544}$	4611 ± 632
TC-RARL	7124 ± 912	5109 ± 348	1172 ± 129	102864 ± 13308	3548 ± 545
DR	6975 ± 992	5490 ± 384	1091 ± 169	109227 ± 17068	3851 ± 612
M2TD3	5330 ± 684	3634 ± 321	938 ± 158	108136 ± 9755	4126 ± 644
RARL	4153 ± 602	154 ± 261	363 ± 58	103366 ± 7604	1689 ± 465
TD3	4025 ± 557	2784 ± 370	1317 ± 189	94352 ± 10101	2020 ± 355

Table 13: Avg. performance against a fixed linear adversary over 10 seeds for each method

Environment Method	Ant	HalfCheetah	Hopper	HumanoidStandup	Walker
Oracle M2TD3	5811 ± 121	3560 ± 167	1216 ± 326	118829 ± 846	4431 ± 615
Oracle RARL	4447 ± 600	-122 ± 64	308 ± 62	81498 ± 12860	1503 ± 450
Oracle TC-M2TD3	7919 ± 595	$\mathbf{7495 \pm 268}$	$\mathbf{2983 \pm 252}$	117610 ± 11682	4952 ± 415
Oracle TC-RARL	$\mathbf{8069 \pm 151}$	7443 ± 236	2805 ± 352	110314 ± 9354	4613 ± 257
Stacked TC-M2TD3	7003 ± 812	6365 ± 335	2714 ± 198	89556 ± 11115	$\mathbf{5256 \pm 675}$
Stacked TC-RARL	7328 ± 251	5301 ± 86	1616 ± 137	105137 ± 7903	4234 ± 385
TC-M2TD3	7622 ± 413	6451 ± 246	2228 ± 131	$\mathbf{129501 \pm 10326}$	4844 ± 417
TC-RARL	7675 ± 143	4881 ± 251	1277 ± 288	105566 ± 15551	3906 ± 381
DR	7713 ± 412	5290 ± 103	1419 ± 122	108711 ± 16696	4307 ± 309
M2TD3	5444 ± 225	3810 ± 69	970 ± 323	106311 ± 9771	4128 ± 727
RARL	4651 ± 446	218 ± 138	346 ± 22	101477 ± 8947	1894 ± 515
TD3	3493 ± 475	1462 ± 1246	1722 ± 366	89934 ± 10644	2396 ± 416

Table 14: Avg. performance against a fixed logarithmic adversary over 10 seeds for each method

Environment Method	Ant	HalfCheetah	Hopper	HumanoidStandup	Walker
Oracle M2TD3	5561 ± 580	3086 ± 163	957 ± 165	119214 ± 2525	4148 ± 630
Oracle RARL	4911 ± 177	-145 ± 67	293 ± 49	79522 ± 13470	1618 ± 142
Oracle TC-M2TD3	7963 ± 796	$\mathbf{6625 \pm 204}$	$\mathbf{2577 \pm 171}$	116664 ± 11798	4818 ± 451
Oracle TC-RARL	$\mathbf{8061 \pm 821}$	6532 ± 304	2572 ± 177	108213 ± 10684	4375 ± 382
Stacked TC-M2TD3	7315 ± 478	5863 ± 290	2283 ± 122	87691 ± 11133	$\mathbf{4931 \pm 735}$
Stacked TC-RARL	7514 ± 62	4770 ± 145	1426 ± 197	104193 ± 8030	3939 ± 369
TC-M2TD3	7910 ± 90	5657 ± 280	1702 ± 226	$\mathbf{128467 \pm 10762}$	4664 ± 412
TC-RARL	7686 ± 208	4475 ± 238	1082 ± 298	104835 ± 16040	3636 ± 428
DR	7883 ± 67	4721 ± 146	1166 ± 332	106171 ± 16867	3995 ± 313
M2TD3	5371 ± 279	3565 ± 105	802 ± 271	104002 ± 11606	4206 ± 712
RARL	4620 ± 763	231 ± 110	340 ± 44	102004 ± 9925	1919 ± 499
TD3	3678 ± 623	576 ± 983	1389 ± 327	88952 ± 11367	1956 ± 360

Table 15: Avg. performance against a fixed exponential adversary over 10 seeds for each method

Environment Method	Ant	HalfCheetah	Hopper	HumanoidStandup	Walker
Oracle M2TD3	5860 ± 93	3780 ± 137	1271 ± 224	119205 ± 1217	4767 ± 815
Oracle RARL	4585 ± 674	-88 ± 79	302 ± 41	82063 ± 13274	1611 ± 342
Oracle TC-M2TD3	7491 ± 624	$\mathbf{8256 \pm 269}$	2894 ± 244	118476 ± 11683	5161 ± 289
Oracle TC-RARL	$\mathbf{7724 \pm 368}$	8000 ± 250	$\mathbf{3036 \pm 293}$	110092 ± 10754	4650 ± 503
Stacked TC-M2TD3	6903 ± 365	7041 ± 302	2721 ± 214	91077 ± 11945	$\mathbf{5310 \pm 882}$
Stacked TC-RARL	7061 ± 222	5741 ± 249	1825 ± 145	104793 ± 6758	4376 ± 342
TC-M2TD3	7318 ± 299	7139 ± 387	2408 ± 113	$\mathbf{129966 \pm 10823}$	4910 ± 663
TC-RARL	7441 ± 133	5326 ± 220	1457 ± 163	106491 ± 14605	4017 ± 439
DR	7389 ± 206	5691 ± 121	1564 ± 99	106290 ± 17502	4224 ± 660
M2TD3	5466 ± 318	3909 ± 332	1062 ± 272	107097 ± 9551	4274 ± 582
RARL	4556 ± 729	228 ± 181	351 ± 24	102096 ± 8291	2053 ± 493
TD3	3771 ± 228	2302 ± 343	2201 ± 219	90496 ± 9487	2768 ± 538

Table 16: Average wall-clock time for each algorithm

	Wall-clock time
TD3	14h
M2TD3	16h
RARL	18h
TC	16h
Stacked TC	16h
Oracle TC	16h

I Computer ressources

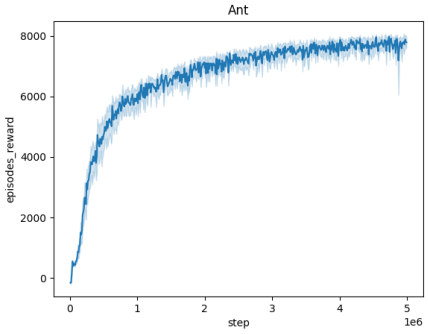
All experiments were run on a desktop machine (Intel i9, 10th generation processor, 64GB RAM) with a single NVIDIA RTX 4090 GPU. Averages and standard deviations were computed from 10 independent repetitions of each experiment.

J Broader impact

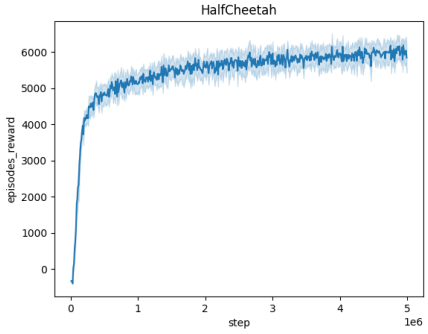
This paper aims to advance robust reinforcement learning. It addresses general mathematical and computational challenges. These challenges may have societal and technological impacts, but we do not find it necessary to highlight them here.

J.1 Limitations

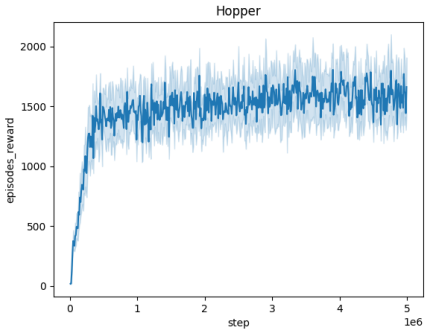
While our proposed Time-Constrained Robust Markov Decision Process (TC-RMDP) framework significantly advances robust reinforcement learning by addressing multifactorial, correlated, and time-dependent disturbances, several limitations must be acknowledged. The TC-RMDP framework assumes that the parameter vector ψ that governs environmental disturbances is known during training. In real-world applications, obtaining such detailed information may not always be feasible. This reliance on precise parameter knowledge limits the practical deployment of our algorithms in environments where ψ cannot be accurately measured or inferred. Our approach assumes that the environment's dynamics can be accurately parameterized and that these parameters remain within a predefined uncertainty set Ψ. This assumption might not hold in more complex or highly dynamic environments where disturbances are not easily parameterized or when the uncertainty set Ψ cannot comprehensively capture all possible variations. Consequently, the robustness of the learned policies might degrade when facing disturbances outside the considered parameter space. Addressing these limitations in future work.



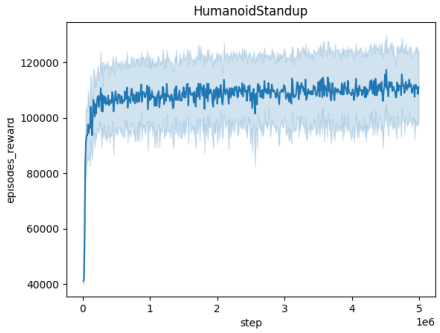
(a) Training curve on Ant with Domain Randomization



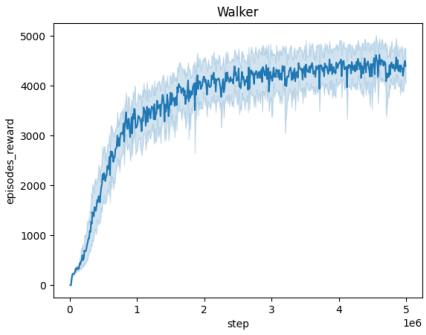
(b) Training curve on HalfCheetah with Domain Randomization



(c) Training curve on Hopper with Domain Randomization

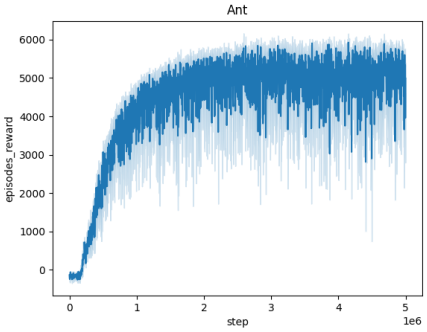


(d) Training curve on HumanoidStandup with Domain Randomization

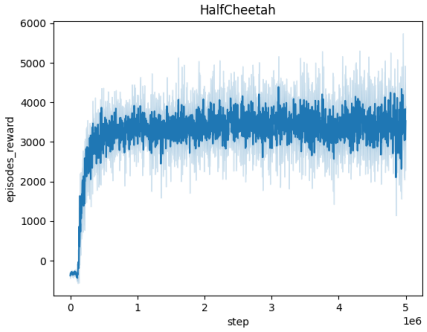


(e) Training curve on Walker with Domain Randomization

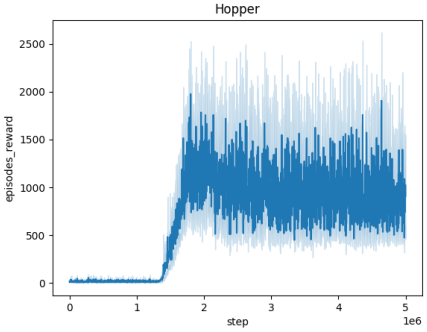
Figure 5: Averaged training curves for the Domain Randomization method over 10 seeds



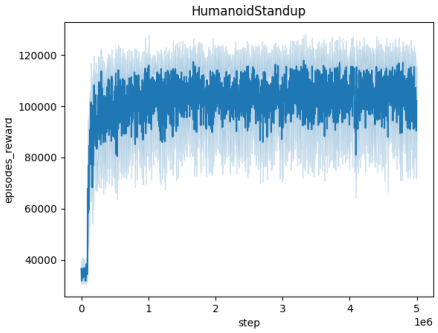
(a) Training curve on Ant with M2TD3



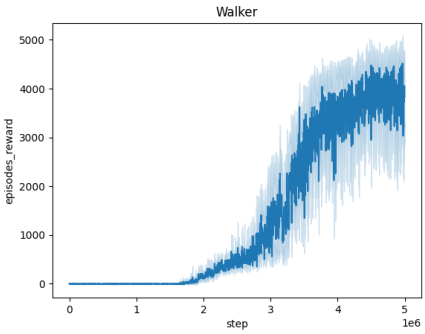
(b) Training curve on HalfCheetah with M2TD3



(c) Training curve on Hopper with M2TD3

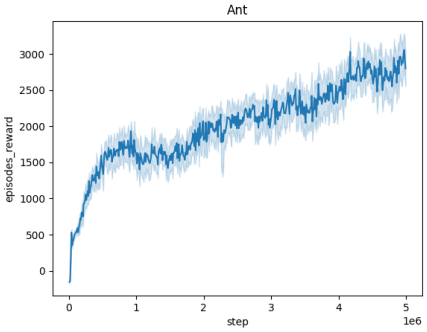


(d) Training curve on HumanoidStandup with M2TD3

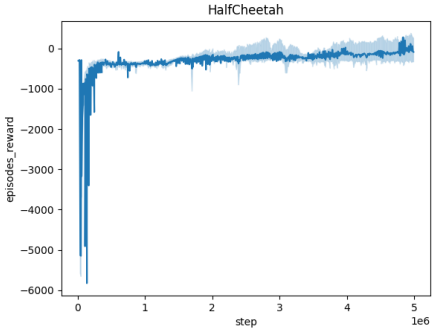


(e) Training curve on Walker with M2TD3

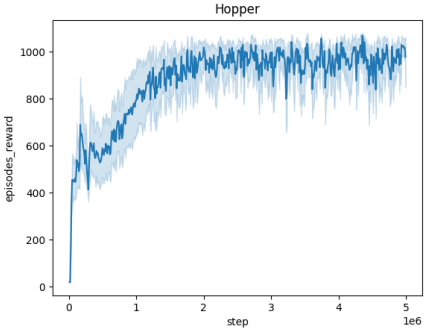
Figure 6: Averaged training curves for the M2TD3 method over 10 seeds



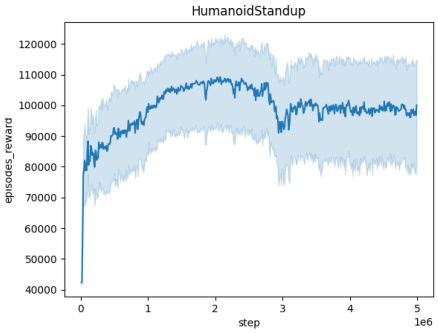
(a) Training curve on Ant with RARL



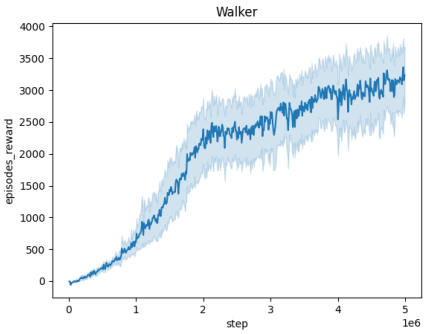
(b) Training curve on HalfCheetah with RARL



(c) Training curve on Hopper with RARL

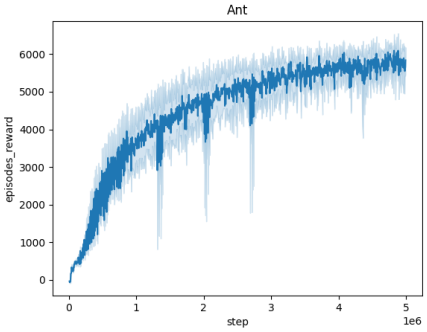


(d) Training curve on HumanoidStandup with RARL

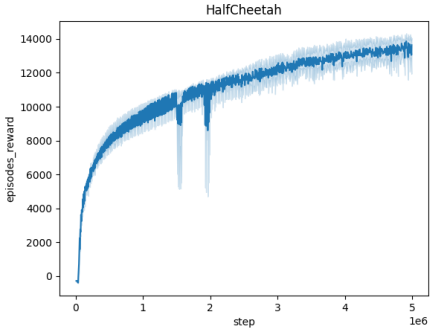


(e) Training curve on Walker with RARL

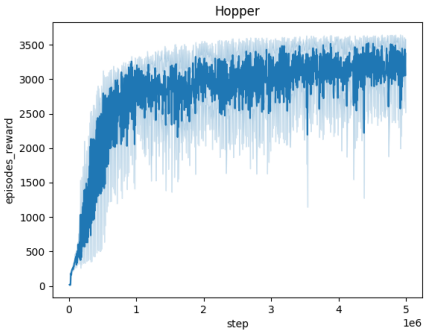
Figure 7: Averaged training curves for the RARL method over 10 seeds



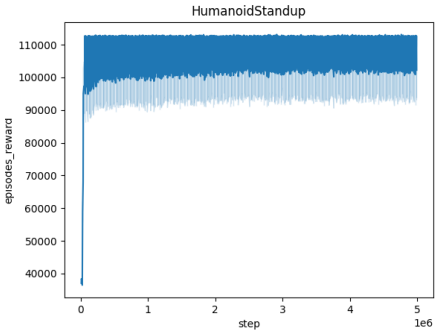
(a) Training curve on Ant with TD3



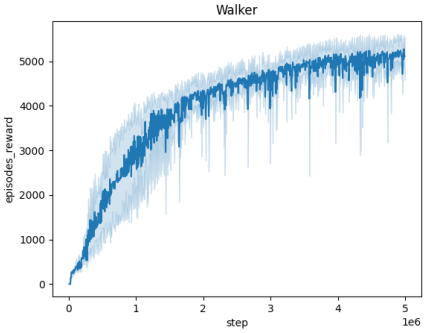
(b) Training curve on HalfCheetah with TD3



(c) Training curve on Hopper with TD3

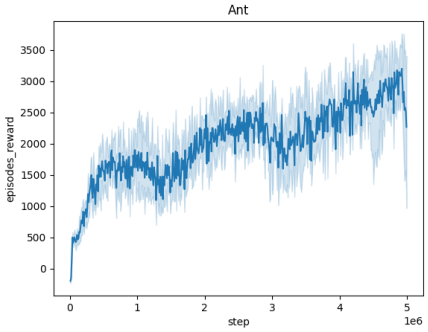


(d) Training curve on HumanoidStandup with TD3

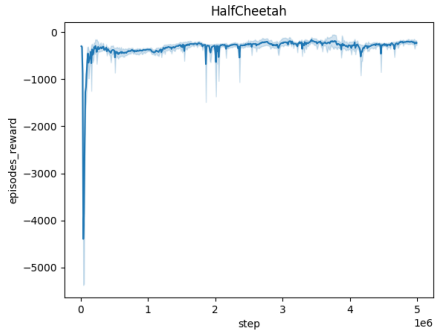


(e) Training curve on Walker with TD3

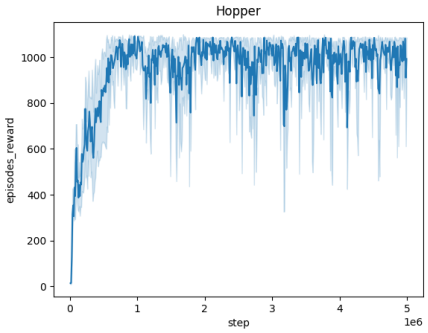
Figure 8: Averaged training curves for the TD3 method over 10 seeds



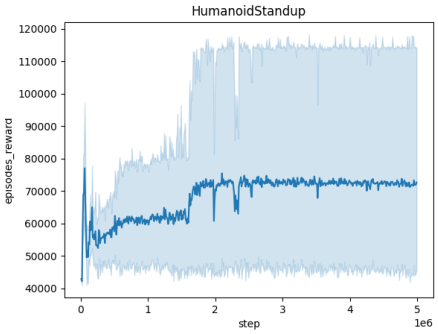
(a) Training curve on Ant with Oracle RARL



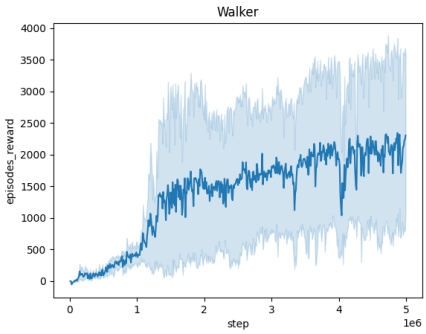
(b) Training curve on HalfCheetah with Oracle RARL



(c) Training curve on Hopper with Oracle RARL

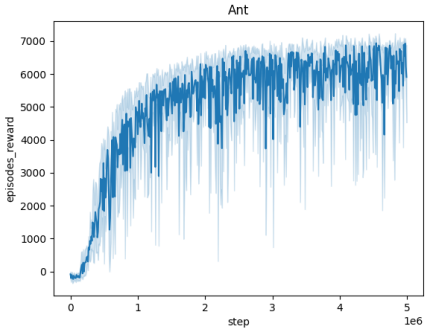


(d) Training curve on HumanoidStandup with Oracle RARL

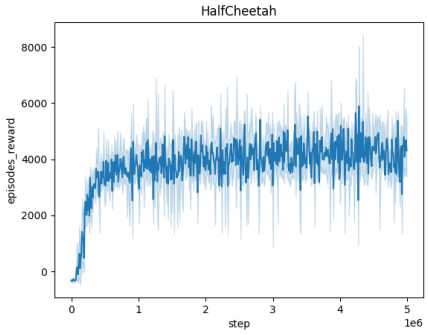


(e) Training curve on Walker with Oracle RARL

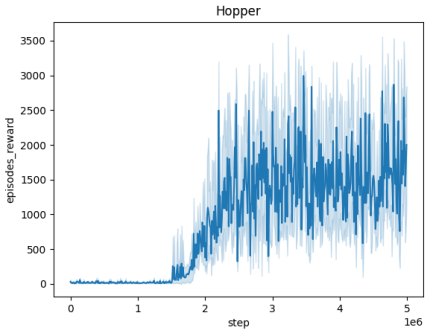
Figure 9: Averaged training curves for the Oracle RARL method over 10 seeds



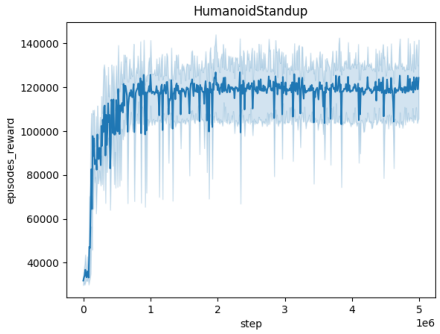
(a) Training curve on Ant with Oracle M2TD3



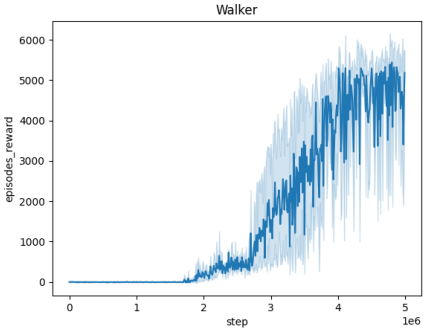
(b) Training curve on HalfCheetah with Oracle M2TD3



(c) Training curve on Hopper with Oracle M2TD3

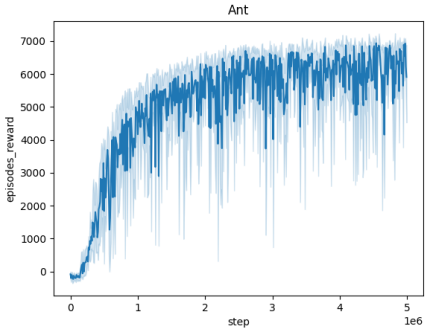


(d) Training curve on HumanoidStandup with Oracle M2TD3

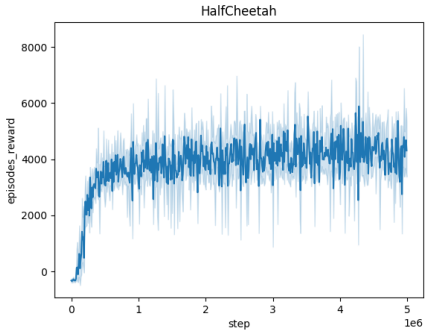


(e) Training curve on Walker with Oracle M2TD3

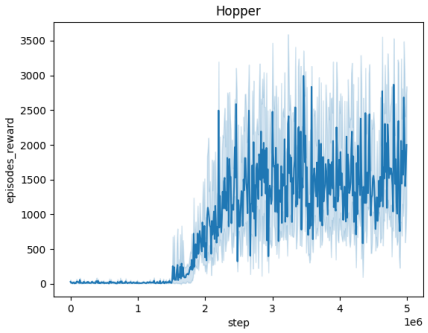
Figure 10: Averaged training curves for the Oracle M2TD3 method over 10 seeds



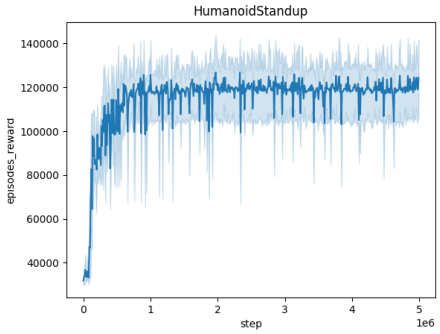
(a) Training curve on Ant with Oracle M2TD3



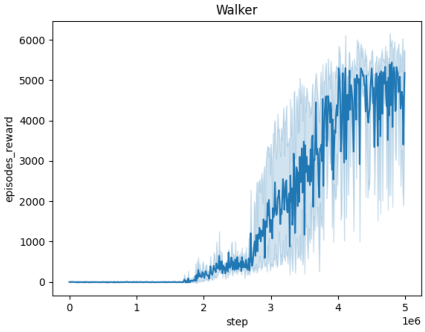
(b) Training curve on HalfCheetah with Oracle M2TD3



(c) Training curve on Hopper with Oracle M2TD3

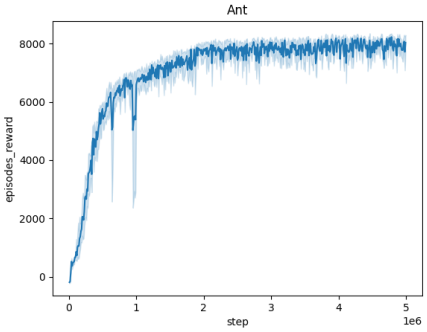


(d) Training curve on HumanoidStandup with Oracle M2TD3

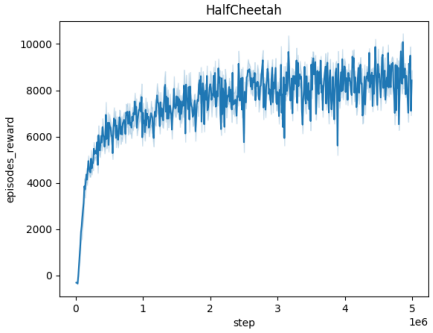


(e) Training curve on Walker with Oracle M2TD3

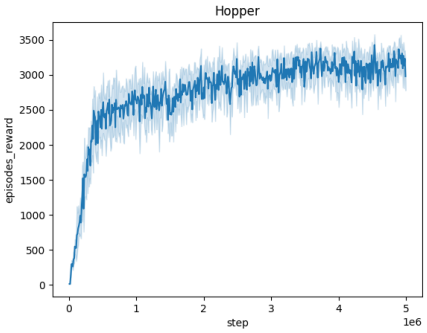
Figure 11: Averaged training curves for the Oracle M2TD3 method over 10 seeds



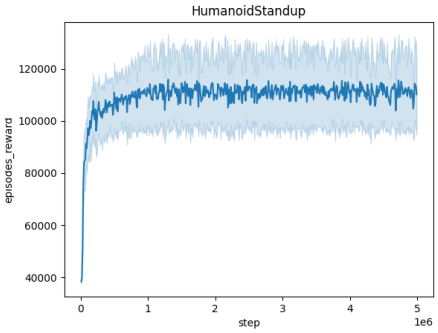
(a) Training curve on Ant with Oracle TC-RARL



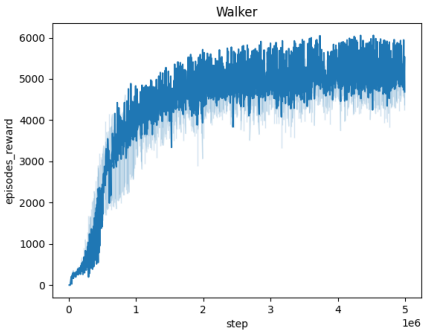
(b) Training curve on HalfCheetah with Oracle TC-RARL



(c) Training curve on Hopper with Oracle TC-RARL

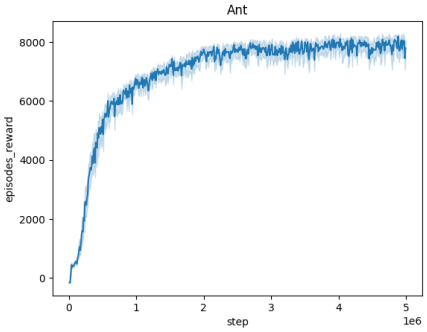


(d) Training curve on HumanoidStandup with Oracle TC-RARL

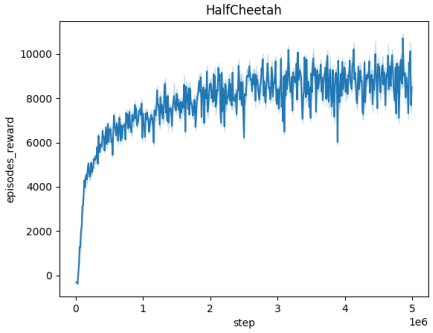


(e) Training curve on Walker with Oracle TC-RARL

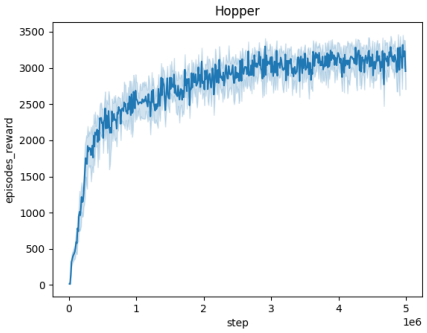
Figure 12: Averaged training curves for the Oracle TC-RARL method over 10 seeds



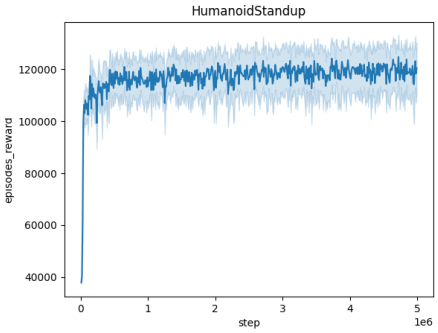
(a) Training curve on Ant with Oracle TC-M2TD3



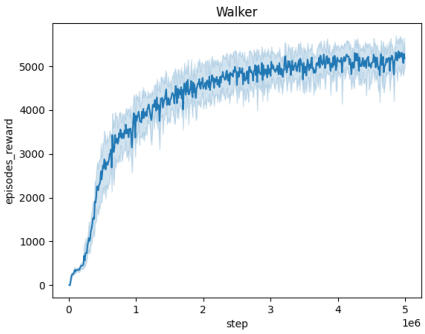
(b) Training curve on HalfCheetah with Oracle TC-M2TD3



(c) Training curve on Hopper with Oracle TC-M2TD3

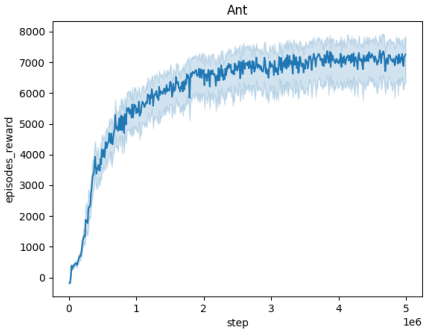


(d) Training curve on HumanoidStandup with Oracle TC-M2TD3

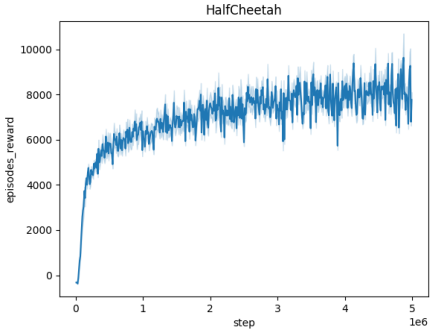


(e) Training curve on Walker with Oracle TC-M2TD3

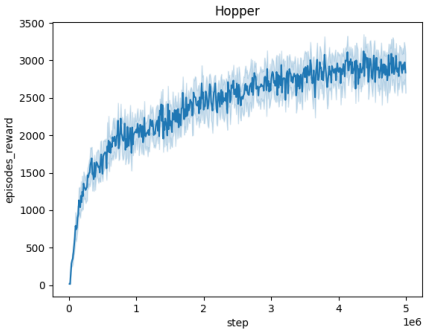
Figure 13: Averaged training curves for the Oracle TC-M2TD3 method over 10 seeds



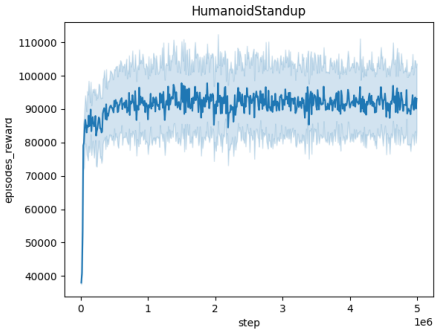
(a) Training curve on Ant with Stacked TC-M2TD3



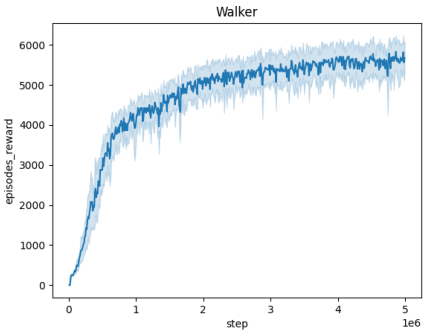
(b) Training curve on HalfCheetah with Stacked TC-M2TD3



(c) Training curve on Hopper with Stacked TC-M2TD3

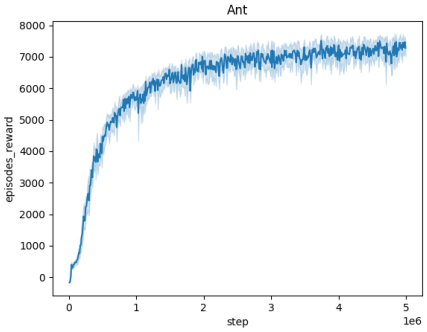


(d) Training curve on HumanoidStandup with Stacked TC-M2TD3

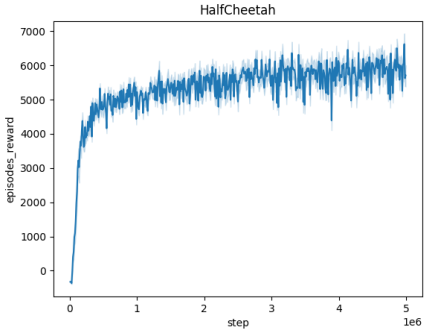


(e) Training curve on Walker with Stacked TC-M2TD3

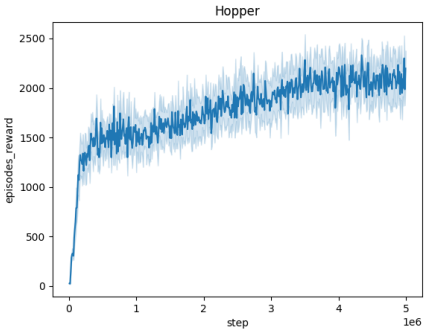
Figure 14: Averaged training curves for the Stacked TC-M2TD3 method over 10 seeds



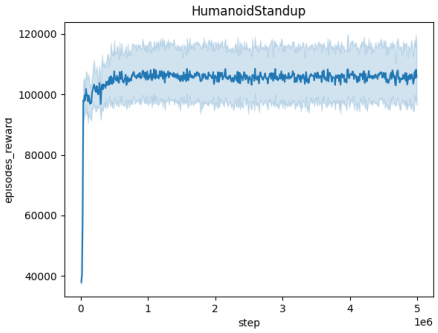
(a) Training curve on Ant with Stacked TC-RARL



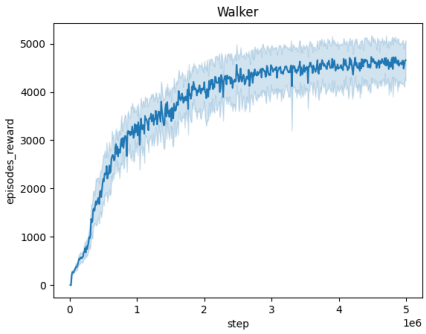
(b) Training curve on HalfCheetah with Stacked TC-RARL



(c) Training curve on Hopper with Stacked TC-RARL



(d) Training curve on HumanoidStandup with Stacked TC-RARL



(e) Training curve on Walker with Stacked TC-RARL

Figure 15: Averaged training curves for the Stacked TC-RARL method over 10 seeds