Reinforcement Learning in Configurable Continuous Environments

Alberto Maria Metelli [1] **Emanuele Ghelfi** [1] **Marcello Restelli** [1]

Abstract

Configurable Markov Decision Processes (Conf-MDPs) have been recently introduced as an extension of the usual MDP model to account for the possibility of *configuring* the environment to improve the agent's performance. Currently, there is still no suitable algorithm to solve the learning problem for real-world Conf-MDPs. In this paper, we fill this gap by proposing a trust-region method, Relative Entropy Model Policy Search (REMPS), able to learn both the policy and the MDP configuration in continuous domains without requiring the knowledge of the true model of the environment. After introducing our approach and providing a finite-sample analysis, we empirically evaluate REMPS on both benchmark and realistic environments by comparing our results with those of the gradient methods.

1. Introduction

The overall goal of Reinforcement Learning (RL, Sutton & Barto, 1998) is to make an agent learn a behavior that maximizes the amount of reward it collects during its interaction with the environment. Most of the problems tackled by RL are typically modeled as a *Markov Decision Process* (MDP, Puterman, 2014) in which the environment is considered a fixed entity and cannot be adjusted. Nevertheless, there exist several real-world motivational examples in which partial control over the environment can be exercised by the agent itself or by an external supervisor (Metelli et al., 2018). For instance, in a car racing problem, the vehicle can be set up to better suit the driver's needs. The entity that performs the configuration can be either the driver itself (agent) or a track engineer (supervisor). With the phrase *environment configuration*, we refer to the activity of altering some environmental parameters to improve the performance of the agent's policy. This scenario has been recently formalized as a *Configurable Markov Decision Process* (Conf-MDP, Metelli et al., 2018). As in traditional RL, in a Conf-MDP the agent looks for the best policy but, in the meantime, there exists an entity entitled to configure the environment with the shared goal of maximizing the final performance of the policy. The nature of this new kind of interaction with the environment cannot be modeled either within the agent's action space or with a multi-agent framework. Indeed, the configuration activity cannot be placed at the same level as the agent's learning process. Configuring the environment may be more expensive and dangerous than updating the agent's policy and may occur on a different time scale w.r.t. the agent's learning process. Furthermore, while the entity that configures the environment must be aware of the presence of the agent (in order to wisely choose the environment configuration), the agent may not be aware of the fact that the configuration is taking place, perceiving the changes in the environment just as a non-stationarity. It is worth noting that the configuration process is rather different from the idea of changing the environment just to encourage learning in the original environment. While in the Conf-MDP framework the environment is altered because we *can* decide to change it (e.g., when a Formula 1 driver selects a car that better fits their driving abilities), in other works, like Ciosek & Whiteson (2017) and Florensa et al. (2017), the configuration is limited to a simulator and does not affect the real environment. Recently, an approach similar to Conf-MDPs, including also and explicit cost for altering the environment, has been proposed (Silva et al., 2018).

Learning in a Conf-MDP, therefore, means finding an agent's policy π together with an environment configuration p that, jointly, maximize the total reward. In Metelli et al. (2018), a safe-learning algorithm, *Safe Policy Model Iteration* (SPMI), is presented to solve the learning problem in the Conf-MDP framework. The basic idea is to optimize a lower bound of the performance improvement to ensure a monotonic increase of the total reward (Kakade & Langford, 2002; Pirotta et al., 2013). Although this approach succeeded in showing the benefits of configuring the environment in some illustrative examples, it is quite far from being applicable to real-world scenarios. We believe there are two significant limitations of SPMI. First of all, it is only applicable to problems with a finite state-action space, while the most interesting Conf-MDP examples have, at

[1]Politecnico di Milano, 32, Piazza Leonardo da Vinci, Milan, Italy. Correspondence to: Alberto Maria Metelli <albertomaria.metelli@polimi.it>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

least, a continuous state space (e.g., the car configuration problem). Second, it requires full knowledge of the environment dynamics. This latter limitation is the most relevant as, in reality, we almost never know the true environment dynamics, and even if a model is available it could be too approximate or too complex and computationally expensive (e.g., the fluid-dynamic model of a car).

In this paper, we propose a new learning algorithm for the Conf-MDP problem that overcomes the main limitations of SPMI. *Relative Entropy Model Policy Search* (REMPS) belongs to the *trust-region* class of methods (Schulman et al., 2015) and takes inspiration from REPS (Peters et al., 2010). REMPS operates with parametric policies $\pi_{\boldsymbol{\theta}}$ and configurations $p_{\boldsymbol{\omega}}$ and can be endowed with an approximate configuration model $\widehat{p}_{\boldsymbol{\omega}}$ that can be estimated from interaction with the environment. At each iteration, REMPS performs two phases: *optimization* and *projection*. In the optimization phase, we aim at identifying a new stationary distribution for the Conf-MDP that maximizes the total reward in a neighborhood of the current stationary distribution. This notion of neighborhood is encoded in our approach as a KL–divergence constraint. However, this distribution may fall outside the space of representable distributions, given the parametrization of the policy and of the configuration. Thus, the projection phase performs a moment projection in order to find an approximation of this stationary distribution in terms of representable policies and configurations.

Our framework shares some aspects with Utility Maximizing Design (Keren et al., 2017); although it assumes that the environment and the applicable modifications are known to the planner, whereas in our setting the agent only knows the environment parameters but ignores their effect on the transition probabilities. Controlling the learning process by employing the KL-divergence was previously done in the Linearly Solvable MDPs (Todorov, 2007) and its extensions (Todorov, 2009; Guan et al., 2014; Busic & Meyn, 2018) considering a penalty, rather than a constraint, to account for the cost of changing the transition probabilities.

In principle, the learning process in a parametric Conf-MDP can be carried out by a standard stochastic gradient method (Sutton et al., 2000; Peters & Schaal, 2008). We can easily adapt the classic REINFORCE (Williams, 1992) and G(PO)MDP (Baxter & Bartlett, 2001) estimators for learning the configuration parameters (see Appendix B). However, we believe that a first-order method does not scale to relevant situations that are of motivating interest in the Conf-MDP framework. For instance, it may be convenient to select a new configuration that makes the performance of the current policy worse because, in this new configuration, we have a much better chance of learning high-performing policies. We argue that this behavior is impossible by using a gradient method, as the gradient update direction attempts to improve performance for all parameters, including those in the transition model. This example justifies the choice of a trust-region method that allows a closed-form optimization in a controlled region. It has been proved empirically that these methods, also in the policy-search framework, are able to overcome local maxima (Levine & Koltun, 2013).

The contribution of this paper is threefold: algorithmic, theoretical and empirical. We start in Section 2 by recalling the definition of MDP, Conf-MDP and some notions of RL. Section 3 introduces our algorithm, REMPS, whose theoretical analysis is provided in Section 4. Section 5 shows how to equip REMPS with an approximation of the environment, while Section 6 presents the experimental evaluation. Finally, in Section 7 we discuss the results and provide some future research directions. The proofs of all results can be found in Appendix A.

2. Preliminaries

A discrete-time Markov Decision Process (MDP, Puterman, 2014) is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \mu, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and the action space respectively, p is the transition model that provides, for every state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, a probability distribution over the next state $p(\cdot|s, a)$, r is the reward model defining the reward collected by the agent $r(s, a, s')$ when performing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ and landing on state $s' \in \mathcal{S}$, μ is the distribution of the initial state and $\gamma \in [0, 1]$ is the discount factor. The behavior of an agent is defined by means of a policy π that provides a probability distribution over the actions $\pi(\cdot|s)$ for every state $s \in \mathcal{S}$. A Configurable Markov Decision Process (Conf-MDP, Metelli et al., 2018) is defined as $\mathcal{CM} = (\mathcal{S}, \mathcal{A}, r, \mu, \gamma, \mathcal{P}, \Pi)$ and extends the MDP definition by considering a configuration space $\mathcal{P}$ instead of a single transition model p and adds a policy space Π to account for the possible limitations of the agent. The performance of a policy-configuration pair (π, p) is defined in terms of the expected return:

$$J_{\pi,p} = \mathop{\mathbb{E}}_{\substack{S_0 \sim \mu \\ A_t \sim \pi(\cdot|S_t) \\ S_{t+1} \sim p(\cdot|S_t, A_t)}} \left[\sum_{t=0}^{+\infty} \gamma^t r(S_t, A_t, S_{t+1}) \right]. \quad (1)$$

When $\gamma = 1$, the previous equation diverges; therefore, we resort to the expected average reward:

$$J_{\pi,p} = \liminf_{H \to +\infty} \mathop{\mathbb{E}}_{\substack{S_0 \sim \mu \\ A_t \sim \pi(\cdot|S_t) \\ S_{t+1} \sim p(\cdot|S_t, A_t)}} \left[\frac{1}{H} \sum_{t=0}^{H-1} r(S_t, A_t, S_{t+1}) \right]. \quad (2)$$

Sometimes we will refer to the state transition kernel $p^\pi(s'|s) = \int_{\mathcal{A}} \pi(a|s) p(s'|s, a) \mathrm{d}a$. For any policy $\pi \in \Pi$ and environment configuration $p \in \mathcal{P}$ we can define the (γ-*discounted*) *stationary distribution*:

$$d_{\pi,p}(s) = (1 - \gamma)\mu(s) + \gamma \int_{\mathcal{S}} d_{\pi,p}(s') p^\pi(s|s') \mathrm{d}s', \quad (3)$$

which represents the discounted number of times state $s \in \mathcal{S}$ is visited under policy π and configuration p. $d_{\pi,p}(s)$ surely exists for $\gamma < 1$. For the case $\gamma = 1$, we will assume that the Markov chain p^π is ergodic for any $\pi \in \Pi$ and $p \in \mathcal{P}$, so that a unique stationary distribution $d_{\pi,p}$ exists. We define the state-action stationary distribution as $d_{\pi,p}(s,a) = d_{\pi,p}(s)\pi(a|s)$ and the state-action-next-state stationary distribution as $d_{\pi,p}(s,a,s') = d_{\pi,p}(s)\pi(a|s)p(s'|s,a)$. In this way, we can unify Equations (1) and (2) as:

$$J_{\pi,p} = \mathop{\mathbb{E}}_{S,A,S' \sim d_{\pi,p}} [r(S,A,S')]. \tag{4}$$

Sometimes, given a stationary distribution d, we will indicate with J_d the corresponding performance. We will denote with $\mathcal{D}_{\Pi,\mathcal{P}}$ the set of all stationary distributions $d_{\pi,p}$ induced by Π and $\mathcal{P}$. In this paper, we assume that the agent policy belongs to a parametric policy space $\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p\}$ as well as the environment configuration $\mathcal{P}_\Omega = \{p_{\boldsymbol{\omega}} : \boldsymbol{\omega} \in \Omega \subseteq \mathbb{R}^q\}$. Thus, the learning problem in a Conf-MDP can be rephrased as finding the optimal policy and configuration parametrizations:

$$\boldsymbol{\theta}^*, \boldsymbol{\omega}^* = \mathop{\arg\max}_{\boldsymbol{\theta}\in\Theta, \boldsymbol{\omega}\in\Omega} J_{\pi_{\boldsymbol{\theta}}, p_{\boldsymbol{\omega}}}. \tag{5}$$

3. Relative Entropy Model Policy Search

In this section, we introduce an algorithm to solve the learning problem in the Conf-MDP framework that can be effectively applied to continuous state-action spaces. *Relative Entropy Model Policy Search* (REMPS), imports several ideas from the classic REPS (Peters et al., 2010); in particular, the use of a constraint to ensure that the resulting new stationary distribution is sufficiently close to the current one. REMPS consists of two subsequent phases: *optimization* and *projection*. In the optimization phase (Section 3.1) we look for the stationary distribution d' that optimizes the performance (4). This search is limited to the space of distributions that are not too dissimilar from the current stationary distribution $d_{\pi,p}$. The notion of dissimilarity is formalized in terms of a threshold $\kappa > 0$ on the KL-divergence. The resulting distribution d' may not fall within the space of the representable stationary distributions given our parametrization $\mathcal{D}_{\Pi_\Theta,\mathcal{P}_\Omega}$. Therefore, similarly to Daniel et al. (2012), in the projection phase (Section 3.2) we need to retrieve a policy $\pi_{\boldsymbol{\theta}}$ and a configuration $p_{\boldsymbol{\omega}}$ inducing a stationary distribution $d_{\pi_{\boldsymbol{\theta}}, p_{\boldsymbol{\omega}}} \in \mathcal{D}_{\Pi_\Theta,\mathcal{P}_\Omega}$ as close as possible to d'.

3.1. Optimization

The optimization problem can be stated in terms of stationary distributions only. Given a stationary distribution d (e.g., the one used to collect samples, i.e., $d_{\pi,p}$) and a KL–divergence threshold $\kappa > 0$, we look for a new stationary distribution d' solving the optimization problem PRIMAL$_\kappa$:

$$\max_{d' \in \Delta(\mathcal{S}\times\mathcal{A}\times\mathcal{S})} J_{d'} \qquad \text{s.t.} \quad D_{\mathrm{KL}}(d'\|d) \le \kappa,$$

where, for a given set $\mathcal{X}$, we have denoted with $\Delta(\mathcal{X})$ the set of all probability distributions over $\mathcal{X}$ and $D_{\mathrm{KL}}(d'\|d) = \mathbb{E}_{S,A,S'\sim d'}\left[\log \frac{d'(S,A,S')}{d(S,A,S')}\right]$ is the KL-divergence between d' and d. It is worth noting that, unlike REPS, we do not impose a constraint on the validity of the stationary distribution w.r.t. the transition model (constraint (7) in Peters et al. (2010)), as we have the possibility to change it. With similar mathematical tools we can solve PRIMAL$_\kappa$ in closed form.

Theorem 3.1. *Let d be a distribution over $\mathcal{S}\times\mathcal{A}\times\mathcal{S}$ and $\kappa > 0$ a KL-divergence threshold. The solution d' of the problem PRIMAL$_\kappa$, for $\kappa > 0$, is given by:*

$$d'(s,a,s') \propto d(s,a,s') \exp\left(\frac{1}{\eta} r(s,a,s')\right), \tag{6}$$

where η is the unique solution of the dual problem DUAL$_\kappa$:

$$\min_{\eta\in[0,+\infty)} g(\eta) = \eta \log \mathop{\mathbb{E}}_{S,A,S'\sim d} \left[\exp\left(\frac{1}{\eta} r(S,A,S') + \kappa\right)\right].$$

Thus, to find the optimal solution of PRIMAL$_\kappa$ we must first determine η, by solving DUAL$_\kappa$. It can be proved, as done in REPS, that with a change of variable $\overline{\eta} = 1/\eta$, we have that $g(\overline{\eta})$ is a convex function (Boyd & Vandenberghe, 2004), and therefore DUAL$_\kappa$ can be easily solved using standard optimization tools. Given a value of η, the new stationary distribution d' is defined by the exponential reweighting of each (s,a,s') triple with its reward $r(s,a,s')$. Moreover, given a stationary distribution d', we can derive a representation of a policy π' and a configuration p' inducing d'.

Corollary 3.1. *The solution d' of PRIMAL$_\kappa$ is induced by the configuration p' and the policy π' defined as:*

$$p'(s'|s,a) \propto p(s'|s,a) \exp\left(\frac{1}{\eta} r(s,a,s')\right),$$

$$\pi'(a|s) \propto \pi(a|s) \mathop{\mathbb{E}}_{S'\sim p(\cdot|s,a)} \left[\exp\left(\frac{1}{\eta} r(s,a,S')\right)\right].$$

In practice, we do not have access to the actual sampling distribution $d_{\pi,p}$, so we cannot compute the exact solution of the dual problem DUAL$_\kappa$. As in REPS, all expectations must be estimated from samples. Given a dataset $\{(s_i,a_i,s'_i,r_i)\}_{i=1}^N$ of N i.i.d. samples drawn from $d_{\pi,p}$, the empirical dual problem $\widetilde{\text{DUAL}}_\kappa$ becomes:

$$\min_{\eta\in[0,+\infty)} \widetilde{g}(\eta) = \eta \log \frac{1}{N} \sum_{i=1}^{N} \exp\left(\frac{1}{\eta} r_i + \kappa\right),$$

which yields the solution $\widetilde{\eta}$ inducing the distribution $\widetilde{d'}$ defined by Equation (6). We discuss the effect of the finite sample size in Section 4.3.

3.2. Projection

The solution d' of the PRIMAL$_\kappa$ problem does not belong, in general, to the class of stationary distributions $\mathcal{D}_{\Pi_\Theta,\mathcal{P}_\Omega}$

induced by Π_Θ and $\mathcal{P}_\Omega$. For this reason, we look for a parametric policy $\pi_{\boldsymbol{\theta}}$ and a parametric configuration $p_{\boldsymbol{\omega}}$ that induce a stationary distribution $d_{\pi_{\boldsymbol{\theta}},p_{\boldsymbol{\omega}}}$ as close as possible to d', by performing a moment projection (PROJ$_d$):[1]

$$\begin{aligned}\boldsymbol{\theta}', \boldsymbol{\omega}' &= \arg\min_{\boldsymbol{\theta}\in\Theta,\boldsymbol{\omega}\in\Omega} D_{\mathrm{KL}}\left(d' \| d_{\pi_{\boldsymbol{\theta}},p_{\boldsymbol{\omega}}}\right) \\ &= \arg\max_{\boldsymbol{\theta}\in\Theta,\boldsymbol{\omega}\in\Omega} \mathop{\mathbb{E}}_{S,A,S'\sim d'} \left[\log d_{\pi_{\boldsymbol{\theta}},p_{\boldsymbol{\omega}}}(S,A,S')\right].\end{aligned}$$

However, this problem is very hard to solve as computing the functional form of $d_{\pi_{\boldsymbol{\theta}},p_{\boldsymbol{\omega}}}$ is complex and cannot be performed in closed form for most cases. If the state space and the action space are finite, we can formulate the problem by defining a set of constraints $d'(s) = (1-\gamma)\mu(s) + \gamma \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} d'(s')\pi_{\boldsymbol{\theta}}(a|s')p_{\boldsymbol{\omega}}(s|s',a), \forall s \in \mathcal{S}$ to enforce the nature of the stationary distribution. Nevertheless, in most of the relevant cases, the problem remains intractable as the state space could be very large. Therefore, we consider more convenient projection approaches that we will justify from a theoretical standpoint in Section 4.1. A first relaxation consists in finding an approximation of the transition kernel $p'^{\pi'}$ induced by d' (PROJ$_{p^\pi}$):

$$\begin{aligned}\boldsymbol{\theta}', \boldsymbol{\omega}' &= \arg\min_{\boldsymbol{\theta}\in\Theta,\boldsymbol{\omega}\in\Omega} \mathop{\mathbb{E}}_{S\sim d'} \left[D_{\mathrm{KL}}\left(p'^{\pi'}(\cdot|S) \| p^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\omega}}(\cdot|S)\right)\right] \\ &= \arg\max_{\boldsymbol{\theta}\in\Theta,\boldsymbol{\omega}\in\Omega} \mathop{\mathbb{E}}_{S,A,S'\sim d'} \left[\log p^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\omega}}(S'|S)\right].\end{aligned}$$

Clearly, we need to be able to compute the functional form of the state transition kernel $p^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\omega}}$, which is only possible when considering finite action spaces. Indeed, in such case, we just have to marginalize over the (finite) action space as: $p^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\omega}}(s'|s) = \sum_{a\in\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s)p_{\boldsymbol{\omega}}(s'|s,a)$. When also the action space is infinite, we resort to separate projections for the policy and the transition model (PROJ$_{\pi,p}$):

$$\begin{aligned}\boldsymbol{\theta}' &= \arg\min_{\boldsymbol{\theta}\in\Theta} \mathop{\mathbb{E}}_{S\sim d'} \left[D_{\mathrm{KL}}\left(\pi'(\cdot|S) \| \pi_{\boldsymbol{\theta}}(\cdot|S)\right)\right] \\ &= \arg\max_{\boldsymbol{\theta}\in\Theta} \mathop{\mathbb{E}}_{S,A,S'\sim d'} \left[\log \pi_{\boldsymbol{\theta}}(A|S)\right], \\ \boldsymbol{\omega}' &= \arg\min_{\boldsymbol{\omega}\in\Omega} \mathop{\mathbb{E}}_{S,A\sim d'} \left[D_{\mathrm{KL}}\left(p'(\cdot|S,A) \| p_{\boldsymbol{\omega}}(\cdot|S,A)\right)\right] \\ &= \arg\max_{\boldsymbol{\omega}\in\Omega} \mathop{\mathbb{E}}_{S,A,S'\sim d'} \left[\log p_{\boldsymbol{\omega}}(S'|S,A)\right].\end{aligned}$$

Similarly to what happens during the optimization phase, we only have access to a finite dataset of N samples. Moreover, here we are faced with an additional challenge, i.e., we need to compute expectations w.r.t. d', but our samples are collected with $d_{\pi,p}$. This can be cast as an off-distribution estimation problem and therefore we resort to importance weighting (Owen, 2013). In the importance weighting estimation, each sample (s_i, a_i, s'_i) is reweighted by the likelihood of being generated by d', i.e., by $w_i = \exp\left(r_i/\widetilde{\eta}\right)$.[2] In the following, we will denote the approximate projections with $\widetilde{\text{PROJ}}$. A summary of the objective functions for the different projection approaches, their applicability, and the corresponding estimators are reported in Table 1.

[1] When using samples, the moment projection is equivalent to the maximum likelihood estimation.

[2] The likelihood is given by the density ratio $\frac{d'(s,a,s')}{d_{\pi,p}(s,a,s')} \propto \exp\left(r(s,a,s')/\widetilde{\eta}\right)$.

Algorithm 1 Relative Entropy Model Policy Search

1: Initialize $\boldsymbol{\theta}_0, \boldsymbol{\omega}_0$ arbitrarily
2: **for** $t = 0, 1, \ldots$ until convergence[3] **do**
3: Collect N samples $\{(s_i, a_i, s'_i, r_i)\}_{i=1}^N$ with $d_{\pi_{\boldsymbol{\theta}_t},p_{\boldsymbol{\omega}_t}}$
4: (*Optimization*) Compute $\widetilde{\eta}$ and $\widetilde{d'}$ solving the $\widetilde{\text{DUAL}}_\kappa$
5: (*Projection*) Perform the projection of $\widetilde{d'}$ and obtain $\boldsymbol{\theta}_{t+1}$ and $\boldsymbol{\omega}_{t+1}$
6: **end for**

The full REMPS problem can be stated as the composition of optimization and projection, i.e., REMPS$_\kappa$ = PROJ $\circ$ PRIMAL$_\kappa$, and the corresponding problem from samples as $\widetilde{\text{REMPS}}_\kappa = \widetilde{\text{PROJ}} \circ \widetilde{\text{PRIMAL}}_\kappa$. Refer to Algorithm 1 for a high-level pseudocode of REMPS.

4. Theoretical Analysis

In this section, we elaborate on three theoretical aspects of REMPS. First of all, we provide three inequalities that bound the difference of performance when changing the policy and the model in terms of distributional divergences between stationary distributions, policies and models (Section 4.1). Secondly, we present a sensitivity study of the hyper-parameter κ (i.e., the KL-divergence threshold) of REMPS (Section 4.2). Finally, we discuss a finite-sample analysis of the single step of REMPS (Section 4.3). In the following, we will not constrain the policy and the configuration spaces to be parametric spaces, so we will omit the parameter space dependence in the symbols Π and $\mathcal{P}$. Furthermore, we will consider the following assumptions.

Assumption 4.1. *(Uniformly bounded reward) For any $s, s' \in \mathcal{S}, a \in \mathcal{A}$ it holds that: $|r(s,a,s')| \leq r_{\max} < +\infty$.*

Assumption 4.2. *(Ergodicity) Let $\pi \in \Pi$ and $p \in \mathcal{P}$, the* ergodicity coefficient *(Seneta, 1988) of the Markov chain induced by π and p is defined as:*

$$\tau\left(p^\pi\right) = \frac{1}{2} \sup_{s,s'\in\mathcal{S}} \left\| p^\pi(\cdot|s) - p^\pi(\cdot|s') \right\|.$$

If $\gamma = 1$, for any $\pi \in \Pi$ and $p \in \mathcal{P}$ we assume $\tau\left(p^\pi\right) \leq \tau_{\max}$.

4.1. Performance Bounds

We start with the following result that bounds the absolute difference of total reward with a dissimilarity index between the stationary distributions.

[3] The algorithm can be stopped after a fixed number of iterations $N_{\max}$ or if the performance improvement between two consecutive iterations is too small.

Table 1. Applicability, exact objective function and corresponding estimator for the three projections presented. w_i is the (non-normalized) importance weight defined as $w_i = \exp(r_i/\widetilde{\eta})$.

Projection	$\lvert\mathcal{S}\rvert = \infty$	$\lvert\mathcal{A}\rvert = \infty$	Exact objective	Estimated objective
PROJ_d	✗	✗	$\mathop{\mathbb{E}}_{S,A,S'\sim d'}\left[\log d_{\pi_{\boldsymbol{\theta}},p_{\boldsymbol{\omega}}}(S,A,S')\right]$	$\frac{1}{N}\sum_{i=1}^{N} w_i \log d_{\pi_{\boldsymbol{\theta}},p_{\boldsymbol{\omega}}}(s_i,a_i,s_i')$
PROJ_{p^π}	✓	✗	$\mathop{\mathbb{E}}_{S,A,S'\sim d'}\left[\log p_{\boldsymbol{\omega}}^{\pi_{\boldsymbol{\theta}}}(S'\lvert S)\right]$	$\frac{1}{N}\sum_{i=1}^{N} w_i \log p_{\boldsymbol{\omega}}^{\pi_{\boldsymbol{\theta}}}(s_i'\lvert s_i)$
$\text{PROJ}_{\pi,p}$	✓	✓	$\mathop{\mathbb{E}}_{S,A,S'\sim d'}\left[\log \pi_{\boldsymbol{\theta}}(A\lvert S)\right]$ $\mathop{\mathbb{E}}_{S,A,S'\sim d'}\left[\log p_{\boldsymbol{\omega}}(S'\lvert S,A)\right]$	$\frac{1}{N}\sum_{i=1}^{N} w_i \log \pi_{\boldsymbol{\theta}}(a_i\lvert s_i)$ $\frac{1}{N}\sum_{i=1}^{N} w_i \log p_{\boldsymbol{\omega}}(s_i'\lvert s_i,a_i)$

Proposition 4.1. *Let d and d' be two stationary distributions, then it holds that:*

$$\lvert J_{d'} - J_d\rvert \le r_{\max}\left\lVert d' - d\right\rVert_1 \le r_{\max}\sqrt{2D_{\text{KL}}\left(d'\lVert d\right)}.$$

This result justifies the use of the projection PROJ_d, since minimizing the KL-divergence between the stationary distributions allows controlling the performance difference. As we have seen in Section 3.2, the PROJ_d is typically intractable. Therefore, we now prove that performing the projection of the state transition kernel (PROJ_{p^π}) still allows controlling the performance difference.

Corollary 4.1. *Let p^π and $p'^{\pi'}$ two transition kernels, inducing the stationary distributions d and d' respectively, then, under Assumption 4.2, it holds that:*

$$\lvert J_{d'} - J_d\rvert \le r_{\max}\rho\sqrt{2\mathop{\mathbb{E}}_{S\sim d'}\left[D_{\text{KL}}\left(p'^{\pi'}(\cdot\lvert S)\lVert p^\pi(\cdot\lvert S)\right)\right]},$$

where $\rho = \frac{\gamma}{1-\gamma}$ if $\gamma < 1$ or $\rho = \frac{1}{1-\tau_{\max}}$ if $\gamma = 1$.

Finally, the following result provides a justification for the separate projections of policy and model ($\text{PROJ}_{\pi,p}$) .

Lemma 4.1. *Let (π, p) and (π', p') be two policy-configuration pairs and let p^π and $p'^{\pi'}$ the corresponding transition kernels, then for any state $s \in \mathcal{S}$, it holds that:*

$$D_{\text{KL}}\left(p'^{\pi'}(\cdot\lvert s)\lVert p^\pi(\cdot\lvert s)\right) \le D_{\text{KL}}\left(\pi'(\cdot\lvert s)\lVert\pi(\cdot\lvert s)\right) + \mathop{\mathbb{E}}_{A\sim\pi'(\cdot\lvert s)}\left[D_{\text{KL}}\left(p'(\cdot\lvert s,A)\lVert p(\cdot\lvert s,A)\right)\right].$$

As an immediate consequence, thanks to the monotonicity property, the inequality remains valid when taking the expectation w.r.t. $S \sim d'$. Thus, we are able to bound the right-hand side of Corollary 4.1 isolating the contribution of policy and configuration.

4.2. Sensitivity to the KL threshold

We analyze how the performance of the solution of PRIMAL_κ changes when the KL-divergence threshold κ varies. The following result upper bounds the reduction in performance between the optimal solution d of PRIMAL_κ and the optimal solution d' of $\text{PRIMAL}_{\kappa'}$ when $\kappa' \le \kappa$.

Proposition 4.2. *Let d and d' be the solutions of PRIMAL_κ and $\text{PRIMAL}_{\kappa'}$ respectively with $\kappa' \le \kappa$, having d_0 as sampling distribution. Then, it holds that:*

$$J_d - J_{d'} \le r_{\max}\lVert d - d_0\rVert_1\left(1 - \frac{\kappa'}{\kappa}\right). \tag{7}$$

This result is general and can be applied broadly to the class of trust-region methods, when using the KL-divergence as a constraint to define the trust-region.

4.3. Finite-sample Analysis

Now we present a finite-sample analysis of the single step of REMPS. In particular, our goal is to upper bound the difference $J_{d'} - J_{\widetilde{\pi}'',\widetilde{p}''}$ where d' is the solution of the exact problem PRIMAL_κ and $d_{\widetilde{\pi}'',\widetilde{p}''}$ is the solution obtained after projecting $\widetilde{d}'$ onto $\mathcal{D}_{\Pi,\mathcal{P}}$. Due to the similarities between the two algorithms, large part of our analysis applies to also REPS (Peters et al., 2010). We will denote $\mathcal{D}_d = \{d' \in \Delta(\mathcal{S}\times\mathcal{A}\times\mathcal{S}) : d' \propto d\exp(r/\eta) : \eta \in [0,+\infty)\}$ the set of possible solutions to the PRIMAL_κ problem.We will consider the following additional assumptions.

Assumption 4.3. *(Finite pseudo-dimension) Given a policy $\pi \in \Pi$ and a transition model $p \in \mathcal{P}$, the pseudo-dimensions of the hypothesis spaces $\{\frac{d}{d_{\pi,p}} : d \in \mathcal{D}_{d_{\pi,p}}\}$, $\{\frac{d}{d_{\pi,p}} r : d \in \mathcal{D}_{d_{\pi,p}}\}$, $\{\frac{d}{d_{\pi,p}}\log\frac{d}{d_{\pi,p}} : d \in \mathcal{D}_{d_{\pi,p}}\}$ and $\{\frac{d}{d_{\pi,p}}\log d' : d \in \mathcal{D}_{d_{\pi,p}}, d' \in \mathcal{D}_{\Pi,\mathcal{P}}\}$ are bounded by $v < +\infty$.*

Assumption 4.4. *(Finite β–moments) There exist $\beta \in (1,2)$, such that*

$$\mathop{\mathbb{E}}_{S,A,S'\sim d_{\pi,p}}\left[\left\lvert\frac{d(S,A,S')}{d_{\pi,p}(S,A,S')}\right\rvert^\beta\right]^{1/\beta} \quad \textit{and}$$

$$\mathop{\mathbb{E}}_{S,A,S'\sim d_{\pi,p}}\left[\left\lvert\frac{d(S,A,S')}{d_{\pi,p}(S,A,S')}\log d'(S,A,S')\right\rvert^\beta\right]^{1/\beta}$$

are bounded for all $d \in \mathcal{D}_{d_{\pi,p}}$ and $d' \in \mathcal{D}_{\Pi,\mathcal{P}}$.

Assumption 4.3 requires that all the involved hypothesis spaces (for the solution of the PRIMAL_κ and PROJ) are characterized by a finite pseudo-dimension. This assumption is necessary to state learning theory guarantees. Assumption 4.4 is more critical as it requires that the involved loss functions (used to solve the PRIMAL_κ and PROJ) have

a uniformly bounded (over the hypothesis space) moment of order $\beta \in (1, 2)$. In particular, the first line states that the exponentiated β-Rényi divergence (Cortes et al., 2010, see Equation (34)) between d and $d_{\pi,p}$ is finite for some $\beta \in (1, 2)$. This requirement allows an analysis based on Cortes et al. (2013) for unbounded loss function with bounded moments. A more straightforward analysis can be made by assuming that the involved loss functions are uniformely bounded and using more traditional tools (Mohri et al., 2012) (see Appendix A.4.4). However, we believe this latter requirement is too restrictive. Therefore, we report below the general statement, under Assumption 4.4.

Theorem 4.1. *(Finite–Sample Bound) Let $\pi \in \Pi$ and $p \in \mathcal{P}$ be the current policy and transition model. Let $\kappa > 0$ be the KL–divergence threshold. Let $d' \in \mathcal{D}_{d_{\pi,p}}$ be the solution of the PRIMAL$_\kappa$ problem and $d_{\widetilde{\pi}'',\widetilde{p}''} \in \mathcal{D}_{\Pi,\mathcal{P}}$ be the solution of the REMPS$_\kappa$ problem with PROJ$_d$ computed with $N > 0$ samples collected with $d_{\pi,p}$. Then, under Assumptions 4.1, 4.3 and 4.4, for any $\alpha \in (1, \beta)$, there exist two constants χ, ξ and a function $\zeta(N) = \mathcal{O}(\log N)$ depending on α, and on the samples, such that for any $\delta \in (0, 1)$, with probability at least $1 - 4\delta$ it holds that:*

$$J_{d'} - J_{\widetilde{\pi}'',\widetilde{p}''} \leq \underbrace{\sqrt{2} r_{\max} \sup_{d \in \mathcal{D}_{d_{\pi,p}}} \inf_{\overline{d} \in \mathcal{D}_{\Pi,\mathcal{P}}} \sqrt{D_{\mathrm{KL}}(d \| \overline{d})}}_{\text{approximation error}}$$
$$+ \underbrace{r_{\max} \chi \sqrt{\epsilon} + r_{\max} \zeta(N) \epsilon + r_{\max} \xi \epsilon^2}_{\text{estimation error}},$$

where $\epsilon = 2^{\frac{\alpha+2}{2\alpha}} \sqrt{\frac{v \log \frac{2eN}{v} + \log \frac{8}{\delta}}{N^{\frac{2(\alpha-1)}{\alpha}}}} \Gamma\left(\alpha, \sqrt{\frac{v \log \frac{2eN}{v} + \log \frac{8}{\delta}}{N^{\frac{2(\alpha-1)}{\alpha}}}}\right)$, *depending on the pseudo-dimension bound* $v < +\infty$ *and* $\Gamma(\alpha, \tau) = \frac{\alpha-1}{\alpha} + \frac{1}{\alpha}\left(\frac{\alpha}{\alpha-1}\right)^{\alpha-1}\left(1 + \left(\frac{\alpha-1}{\alpha}\right)^{\alpha-1} \log \frac{1}{\tau}\right)^{\frac{\alpha-1}{\alpha}}$.

Proof Sketch. The idea of the proof is to decouple the contributions of (i) PRIMAL$_\kappa$ and (ii) PROJ$_d$ to the final error:

$$J_{d'} - J_{\widetilde{\pi}'',\widetilde{p}''} = \underbrace{J_{d'} - J_{\widetilde{d}'}}_{\text{(i)}} + \underbrace{J_{\widetilde{d}'} - J_{\widetilde{\pi}'',\widetilde{p}''}}_{\text{(ii)}},$$

where $\widetilde{d}'$ is the solution of PRIMAL$_\kappa$. (i) is the contribution of the *estimation* error due to the finite number of samples used to solve PRIMAL$_\kappa$. It is analyzed in Lemma A.3, exploiting the sensitivity analysis of Lemma 4.2. (ii) includes the contribution of an *approximation* error due to the space in which we represent the stationary distributions $\mathcal{D}_{\Pi,\mathcal{P}}$ and an estimation error due to the finite number of samples used to perform the projection PROJ$_d$. This term can be decomposed using Lemma A.4. Lemmas A.7-A.10 are then employed to state learning theory bounds over the various terms that allow completing the proof of the main statement. □

The estimation error is dominated by $\sqrt{\epsilon}$. Ignoring logarithmic terms, we have that $J_{d'} - J_{\widetilde{\pi}'',\widetilde{p}''} = \widetilde{\mathcal{O}}\left(N^{-\frac{2(\alpha-1)}{4\alpha}}\right)$. In this analysis, we considered the case in which the projection is performed over the stationary distribution (PROJ$_d$).[4] The result can be easily extended to the case in which we resort to PROJ$_{p^\pi}$ or PROJ$_{\pi,p}$ (Corollary A.1).

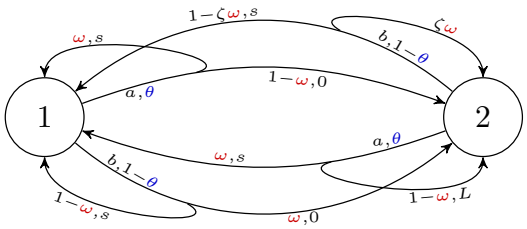

Figure 1. The Chain Domain.

5. Approximation of the Transition Model

The formulation of REMPS we presented above, requires access to a representation of the environment model $p_{\boldsymbol{\omega}}$, depending on a vector of parameters $\boldsymbol{\omega}$. Although the parameters that can be configured are usually known; the environment dynamics is unknown in a model-free scenario. Even when an environment model is available it may be too imprecise or too complex to be used effectively. In principle, we could resort to a general model-based RL approach to effectively approximate the transition model (Deisenroth & Rasmussen, 2011; Nagabandi et al., 2018). However, in our scenario, we need to learn a mapping from state-action-configuration triples to a new state. Our approach is based on a simple maximum likelihood estimation. Given a dataset of experience $\{(s_i, a_i, s'_i, \boldsymbol{\omega}_i)\}_{i=1}^N$ (possibly collected with different policies π_i and different configurations $\boldsymbol{\omega}_i$) and given an approximation space $\widehat{\mathcal{P}}_\Omega$ we solve the problem:

$$\max_{\widehat{p} \in \widehat{\mathcal{P}}_\Omega} \frac{1}{N} \sum_{i=1}^{N} \log \widehat{p}(s'_i | s_i, a_i, \boldsymbol{\omega}_i), \tag{8}$$

where we made explicit that the distribution of the next state s'_i depends also on the configuration.[5] Given the model approximation, we can run REMPS by replacing p with $\widehat{p}$. We do not impose any restriction on the specific model class $\widehat{\mathcal{P}}_\Omega$ (e.g., neural network, Gaussian process) and on the moment in which the fitting phase has to be performed (e.g., at the beginning of the training or every m iterations).

6. Experiments

In this section, we provide the experimental evaluation of REMPS on three domains: a simple chain domain (Section 6.1, Figure 1), the classical Cartpole (Section 6.2) and

[4] Note that Assumption 4.4 ensures that the approximation error is finite, since the KL-divergence is the 1-Rényi divergence and the Rényi divergence is non-decreasing in the order β (Van Erven & Harremos, 2014).

[5] Notice that the configuration parameters $\boldsymbol{\omega}$ are an input of the approximate model.

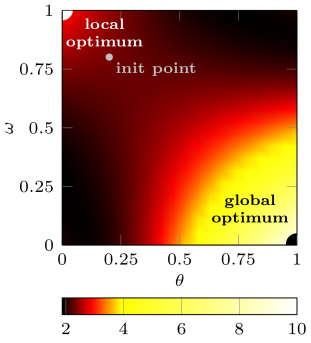

Figure 2. Return surface of the Chain domain.

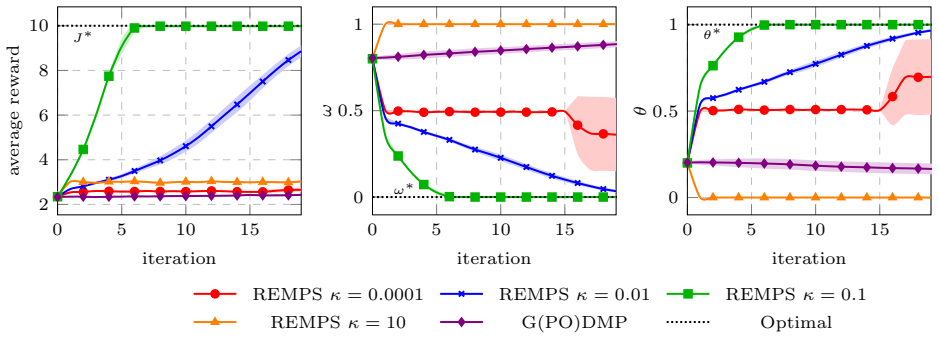

Figure 3. Average reward, configuration parameter ω, and policy parameter θ, as a function of the number of iterations for REMPS with different values of κ and G(PO)MDP. 20 runs, 95% c.i.

a more challenging car-configuration task based on TORCS (Section 6.3). In the first two experiments, we compare REMPS with the extension of G(PO)MDP to the policy-configuration learning (Appendix B), whereas in the last experiment we evaluate REMPS against REPS, the latter used for policy learning only. Full experimental details are reported in Appendix D.

6.1. Chain Domain

The Chain Domain (Figure 1) is an illustrative example of Conf-MDP. There are two states 1 and 2 and the agent can perform two actions a (forward) and b (backward). The agent is forced to play every action with the same probability in both states, i.e., $\pi_\theta(a|s) = \theta$ and $\pi_\theta(b|s) = 1 - \theta$ for all $s \in \{1, 2\}$ and $\theta \in [0, 1]$. The environment can be configured via the parameter $\omega \in [0, 1]$, that is the probability of action failure. Action a, if successful, takes the agent to state 2, whereas action b, if successful, takes the agent to state 1. When one action fails, the other is executed. The agent gets a high reward, $L > 0$, if, starting from state 1, it successfully executes action a, while it gets a smaller reward, l ($0 < l < L$) if it lands in state 2 starting from 1 but by performing action b. The agent gets an even smaller reward, s ($0 < s < l$), when it lands in state 1. The parameter $\zeta \in [0, 1]$ is not configurable and has been added to avoid symmetries in the return surface.

The main goal of this experiment is to show the benefits of our algorithm compared to a simple gradient method, assuming to know the exact environment model. The return surface is characterized by two local maxima (Figure 2). If the system is initialized in a suitable region (as in Figure 2), to reach the global maximum we need to change the model in order to worsen the current policy performance. In Figure 3, we compare our algorithm REMPS using PROJ$_{p^\pi}$ with different values of κ, against G(PO)MDP adapted to model learning (see Appendix B). We can see that G(PO)MDP, besides the slow convergence, moves in the direction of the local maximum. Instead, for some appropriate values of the hyperparameter (e.g., $\kappa \in \{0.1, 0.01\}$) REMPS is able to reach the global optimum. It is worth noting that too small a value of κ (e.g., $\kappa = 0.0001$) prevents escaping the basin of attraction of the local maximum. Likewise, for too large κ (e.g., $\kappa = 10$) the estimated quantities are too uncertain and therefore we are not able to reach the global optimum as well. Hyperparameter values and further experiments, including the effect of the different projection strategies, no-configuration cases, and the comparison with SPMI (Metelli et al., 2018), are reported in Appendix D.1.

6.2. Cartpole

The Cartpole domain (Widrow & Smith, 1964; Barto et al., 1983) is a continuous-state and finite-action environment. We add to the standard Cartpole domain the possibility to configure the cart force, via the parameter ω. We consider in the reward function an additional penalization proportional to the applied force, so that an optimal agent should find the smallest force that allows the pole to remain in a vertical position (details in Appendix D.2.1). The goal of this experiment is to test the ability of REMPS to learn jointly the policy and the environment configuration in a continuous state environment, as well as the effect of replacing the exact environment model with an approximator, trained just at the beginning of the learning process.

In Figure 4, we compare the performance of REMPS, with the two projection strategies PROJ$_{p^\pi}$ and PROJ$_{\pi,p}$, and G(PO)MDP, starting from a fixed value of the model parameter ($\omega_0 = 8$), both for the case of exact model and approximate model. In the exact case, the performance of REMPS are similar to those of G(PO)MDP. The latter is even faster to achieve a good performance, although it shows a larger variance across the runs. No significant difference can be found between PROJ$_{p^\pi}$ and PROJ$_{\pi,p}$ in this case. Instead, in the approximated scenario, REMPS notably outperforms G(PO)MDP, which shows a very unstable

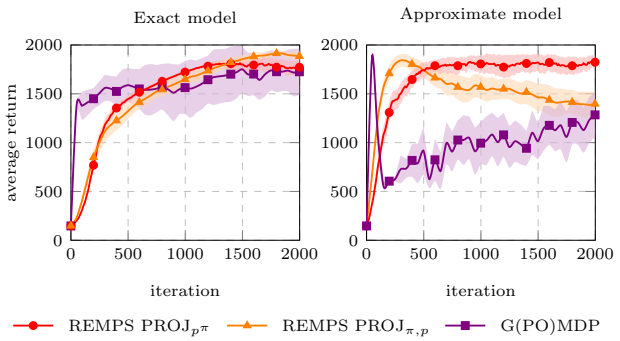

Figure 4. Average return as a function of the number of iterations for the Cartpole experiment when the environment model is exact (left) or approximated from samples (right) comparing REMPS with PROJ$_{p^\pi}$, PROJ$_{\pi,p}$ and G(PO)MDP. 20 runs, 95% c.i.

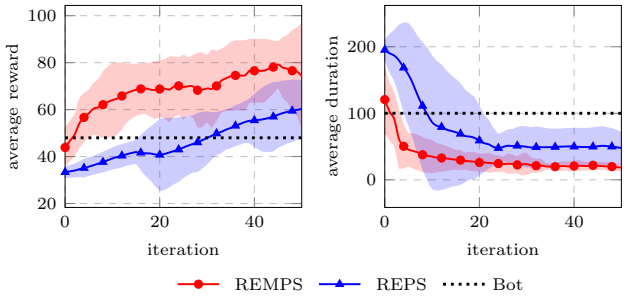

Figure 5. Average reward and episode duration as a function of the number of iterations for the TORCS experiment comparing REMPS, REPS and the bot. 10 runs, 80% c.i.

curve. Indeed, constraining the search in a trust-region, as REMPS does by means of κ, is even more important in the approximate case, since the estimated quantities are affected by further uncertainty (injected by the approximated model of the environment). It is worth noting that in this case the difference between PROJ$_{p^\pi}$ and PROJ$_{\pi,p}$ is more visible. Indeed, PROJ$_{\pi,p}$ is less precise than PROJ$_{p^\pi}$ (being a relaxation) and thus, when projecting d', it trusts the approximate model moving towards a suboptimal configuration. Further details on the Cartpole experiments are given in Appendix D.2.

6.3. Driving and Configuring with TORCS

The Open Racing Car Simulator TORCS (Wymann et al., 2000) is a car racing simulation that allows simulating driving races. TORCS has been used several times in RL (Loiacono et al., 2010; Koutník et al., 2013; Lillicrap et al., 2015; Mnih et al., 2016). We modified TORCS adding the possibility to configure the car parameters taking inspiration from the "Car Setup Competition" (Loiacono et al., 2013, details in Appendix D.3.1). The agent's observation is a low-dimensional representation of the car's sensors (including speed, focus and wheel speeds), while the action space is composed of steering and acceleration/braking (continuous).

The goal of this experiment is to show the ability of REMPS to learn policy and configuration in a continuous state-action space, like a car racing scenario. We consider a configuration space made of three parameters: rear and front wing orientation and brake repartition between front and rear. We start with a policy pretrained via behavioral cloning, using samples collected with a driving bot (snakeoil). Using the same bot, we collect a dataset of episodes with different parameter values, used to train an approximation of the environment. In Figure 5, we compare the average reward and the average lap time for REMPS (with PROJ$_{\pi,p}$), in which we act on both the policy and the model, and REPS, in which only policy learning is enabled. We can notice that REMPS is able to reach performances larger than those achievable without configuring the environment. In this experiment, we can appreciate another remarkable benefit of environment configurability: configuring the environment can also speed up the learning process (online performance), as clearly visible in Figure 5. Full experimental results can be found in Appendix D.3.

7. Discussion and Conclusions

Environment configurability is a relevant property of many real-world domains, with significant potential benefits when accounted by the RL algorithms. In this paper, we proposed a novel trust-region algorithm, REMPS, which takes advantage of this possibility to jointly learn an agent policy and an environment configuration. Unlike previous works, REMPS can be employed in continuous state-action spaces and does not require the knowledge of the exact environment dynamics. Furthermore, we derived several interesting properties of REMPS, especially we provided a finite-sample analysis for the single step of REMPS. Finally, the experimental evaluation showed that configuring the environment, on the one hand, allows the agent to learn highly performing policies; on the other hand, it might speed up the learning process itself. Moreover, REMPS showed the ability to overcome some of the limitations of gradient methods when employed to configure environments, even in the presence of approximate models. The future research directions include a more-in-depth analysis of REMPS, especially by studying the effect of dynamically modifying the KL-divergence threshold κ and the extension of the theoretical analysis for finite-time guarantees as well as for accounting of the approximate model of the environment. More generally, it is interesting to investigate diverse applications of the Conf-MDP framework, with particular attention of removing the knowledge of the agent policy space.

References

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Busic, A. and Meyn, S. Action-constrained markov decision processes with kullback-leibler cost. In *Conference On Learning Theory (COLT)*, 2018.

Ciosek, K. A. and Whiteson, S. Offer: Off-environment reinforcement learning. In *AAAI*, pp. 1819–1825, 2017.

Cortes, C., Mansour, Y., and Mohri, M. Learning bounds for importance weighting. In *Advances in Neural Information Processing Systems*, pp. 442–450, 2010.

Cortes, C., Greenberg, S., and Mohri, M. Relative deviation learning bounds and generalization with unbounded loss functions. *arXiv preprint arXiv:1310.5796*, 2013.

Daniel, C., Neumann, G., and Peters, J. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pp. 273–281, 2012.

Deisenroth, M. P. and Rasmussen, C. E. PILCO: A model-based and data-efficient approach to policy search. In Getoor, L. and Scheffer, T. (eds.), *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 465–472. Omnipress, 2011.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pp. 482–495, 2017.

Gelfand, I. M., Silverman, R. A., et al. *Calculus of variations*. Courier Corporation, 2000.

Guan, P., Raginsky, M., and Willett, R. M. Online markov decision processes with kullback–leibler control cost. *IEEE Transactions on Automatic Control*, 59(6):1423–1438, 2014.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In Sammut, C. and Hoffmann, A. G. (eds.), *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pp. 267–274. Morgan Kaufmann, 2002.

Keren, S., Pineda, L., Gal, A., Karpas, E., and Zilberstein, S. Equi-reward utility maximizing design in stochastic environments. *HSDIP 2017*, pp. 19, 2017.

Koutník, J., Cuccu, G., Schmidhuber, J., and Gomez, F. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pp. 1061–1068, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1963-8.

Levine, S. and Koltun, V. Guided policy search. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 1–9. JMLR.org, 2013.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Loiacono, D., Prete, A., Lanzi, P. L., and Cardamone, L. Learning to overtake in torcs using simple reinforcement learning. In *IEEE Congress on Evolutionary Computation*, pp. 1–8, July 2010. doi: 10.1109/CEC.2010.5586191.

Loiacono, D., Cardamone, L., and Lanzi, P. L. Simulated car racing championship: Competition software manual. *arXiv preprint arXiv:1304.1672*, 2013.

Metelli, A. M., Mutti, M., and Restelli, M. Configurable markov decision processes. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3488–3497. PMLR, 2018.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT press, 2012.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

Owen, A. B. *Monte Carlo theory, methods and examples*. 2013.

Peters, J. and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

Peters, J., Mülling, K., and Altun, Y. Relative entropy policy search. In *AAAI*, pp. 1607–1612. Atlanta, 2010.

Pirotta, M., Restelli, M., Pecorino, A., and Calandriello, D. Safe policy iteration. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 307–315. JMLR.org, 2013.

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

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015.

Seneta, E. Perturbation of the stationary distribution measured by ergodicity coefficients. *Advances in Applied Probability*, 20(1):228–230, 1988.

Silva, R., Melo, F. S., and Veloso, M. What if the world were different? gradient-based exploration for new optimal policies. *EPiC Series in Computing*, 55:229–242, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

Todorov, E. Linearly-solvable markov decision problems. In *Advances in Neural Information Processing Systems*, pp. 1369–1376, 2007.

Todorov, E. Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28): 11478–11483, 2009.

Van Erven, T. and Harremos, P. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

Widrow, B. and Smith, F. W. Pattern-recognizing control systems. *Computer and information sciences*, 288, 1964.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., and Sumner, A. Torcs, the open racing car simulator. 4:6, 2000.