Control Frequency Adaptation via Action Persistence in Batch Reinforcement Learning

Alberto Maria Metelli [1] **Flavio Mazzolini** [1] **Lorenzo Bisi** [1 2] **Luca Sabbioni** [1 2] **Marcello Restelli** [1 2]

Abstract

The choice of the control frequency of a system has a relevant impact on the ability of *reinforcement learning algorithms* to learn a highly performing policy. In this paper, we introduce the notion of *action persistence* that consists in the repetition of an action for a fixed number of decision steps, having the effect of modifying the control frequency. We start analyzing how action persistence affects the performance of the optimal policy, and then we present a novel algorithm, *Persistent Fitted Q-Iteration* (PFQI), that extends FQI, with the goal of learning the optimal value function at a given persistence. After having provided a theoretical study of PFQI and a heuristic approach to identify the optimal persistence, we present an experimental campaign on benchmark domains to show the advantages of action persistence and proving the effectiveness of our persistence selection method.

1. Introduction

In recent years, Reinforcement Learning (RL, Sutton & Barto, 2018) has proven to be a successful approach to address complex control tasks: from robotic locomotion (e.g., Peters & Schaal, 2008; Kober & Peters, 2014; Haarnoja et al., 2019; Kilinc et al., 2019) to continuous system control (e.g., Schulman et al., 2015; Lillicrap et al., 2016; Schulman et al., 2017). These classes of problems are usually formalized in the framework of the *discrete–time* Markov Decision Processes (MDP, Puterman, 2014), assuming that the control signal is issued at discrete time instants. However, many relevant real–world problems are more naturally defined in the continuous–time domain (Luenberger, 1979). Even though a branch of literature has studied RL in *continuous–time* MDPs (Bradtke & Duff, 1994; Munos & Bourgine, 1997; Doya, 2000), the majority of the research has focused on the discrete–time formulation, which appears to be a necessary, but effective, approximation.

Intuitively, increasing the *control frequency* of the system offers the agent more control opportunities, possibly leading to improved performance as the agent has access to a larger *policy space*. This might wrongly suggest that we should control the system with the highest frequency possible, within its physical limits. However, in the RL framework, the environment dynamics is unknown, thus, a too fine discretization could result in the opposite effect, making the problem harder to solve. Indeed, any RL algorithm needs samples to figure out (implicitly or explicitly) how the environment evolves as an effect of the agent's actions. When increasing the control frequency, the *advantage* of individual actions becomes infinitesimal, making them almost indistinguishable for standard *value-based* RL approaches (Tallec et al., 2019). As a consequence, the *sample complexity* increases. Instead, low frequencies allow the environment to evolve longer, making the effect of individual actions more easily detectable. Furthermore, in the presence of a system characterized by a "slowly evolving" dynamics, the gain obtained by increasing the control frequency might become negligible. Finally, in robotics, lower frequencies help to overcome some partial observability issues, like action execution delays (Kober & Peters, 2014).

Therefore, we experience a fundamental *trade–off* in the control frequency choice that involves the policy space (larger at high frequency) and the sample complexity (smaller at low frequency). Thus, it seems natural to wonder: "*what is the optimal control frequency?*" An answer to this question can disregard neither the task we are facing nor the learning algorithm we intend to employ. Indeed, the performance loss we experience by reducing the control frequency depends strictly on the properties of the system and, thus, of the task. Similarly, the dependence of the sample complexity on the control frequency is related to how the learning algorithm will employ the collected samples.

In this paper, we analyze and exploit this trade–off in the context of batch RL (Lange et al., 2012), with the goal of enhancing the learning process and achieving higher perfor-

[1]Politecnico di Milano, Milan, Italy. [2]Institute for Scientific Interchange Foundation, Turin, Italy. Correspondence to: Alberto Maria Metelli <albertomaria.metelli@polimi.it>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

mance. We assume to have access to a discrete–time MDP $\mathcal{M}_{\Delta t_0}$, called base MDP, which is obtained from the time discretization of a continuous–time MDP with fixed base control time step Δt_0, or equivalently, a control frequency equal to $f_0 = \frac{1}{\Delta t_0}$. In this setting, we want to select a suitable *control time step* Δt that is an integer multiple of the base time step Δt_0, i.e., $\Delta t = k \Delta t_0$ with $k \in \mathbb{N}_{\geqslant 1}$.[1] Any choice of k generates an MDP $\mathcal{M}_{k\Delta t_0}$ obtained from the base one $\mathcal{M}_{\Delta t_0}$ by altering the transition model so that each action is repeated for k times. For this reason, we refer to k as the action *persistence*, i.e., the number of decision epochs in which an action is kept fixed. It is possible to appreciate the same effect in the base MDP $\mathcal{M}_{\Delta t_0}$ by executing a (non-Markovian and non-stationary) policy that persists every action for k time steps. The idea of repeating actions has been previously employed, although heuristically, with deep RL architectures (Lakshminarayanan et al., 2017).

The contributions of this paper are theoretical, algorithmic, and experimental. We first prove that action persistence (with a fixed k) can be represented by a suitable modification of the Bellman operators, which preserves the contraction property and, consequently, allows deriving the corresponding value functions (Section 3). Since increasing the duration of the control time step $k\Delta t_0$ has the effect of degrading the performance of the optimal policy, we derive an algorithm–independent bound for the difference between the optimal value functions of MDPs $\mathcal{M}_{\Delta t_0}$ and $\mathcal{M}_{k\Delta t_0}$, which holds under Lipschitz conditions. The result confirms the intuition that the performance loss is strictly related to how fast the environment evolves as an effect of the actions (Section 4). Then, we apply the notion of action persistence in the batch RL scenario, proposing and analyzing an extension of Fitted Q-Iteration (FQI, Ernst et al., 2005). The resulting algorithm, *Persistent Fitted Q-Iteration* (PFQI) takes as input a target persistence k and estimates the corresponding optimal value function, assuming to have access to a dataset of samples collected in the base MDP $\mathcal{M}_{\Delta t_0}$ (Section 5). Once we estimate the value function for a set of candidate persistences $\mathcal{K} \subset \mathbb{N}_{\geqslant 1}$, we aim at selecting the one that yields the best performing greedy policy. Thus, we introduce a persistence selection heuristic able to approximate the optimal persistence, without requiring further interactions with the environment (Section 6). After having revised the literature (Section 7), we present an experimental evaluation on benchmark domains, to confirm our theoretical findings and evaluate our persistence selection method (Section 8). We conclude by discussing some open questions related to action persistence (Section 9). The proofs of all the results can be found in Appendix A. The code is available at `github.com/albertometelli/pfqi`.

[1] We are considering the *near–continuous* setting. This is almost w.l.o.g. compared to the continuous time since the discretization time step Δt_0 can be chosen to be arbitrarily small. Typically, a lower bound on Δt_0 is imposed by the physical limitations of the system. Thus, we restrict the search of Δt from the continuous set $\mathbb{R}_{>0}$ to the discrete set $\{k\Delta t_0, k \in \mathbb{N}_{\geqslant 1}\}$. Moreover, considering an already discretized MDP simplifies the mathematical treatment.

2. Preliminaries

In this section, we introduce the notation and the basic notions that we will employ in the remainder of the paper.

Mathematical Background Let $\mathcal{X}$ be a set with a σ-algebra $\sigma_{\mathcal{X}}$, we denote with $\mathscr{P}(\mathcal{X})$ the set of all probability measures and with $\mathscr{B}(\mathcal{X})$ the set of all bounded measurable functions over $(\mathcal{X}, \sigma_{\mathcal{X}})$. If $x \in \mathcal{X}$, we denote with δ_x the Dirac measure defined on x. Given a probability measure $\rho \in \mathscr{P}(\mathcal{X})$ and a measurable function $f \in \mathscr{B}(\mathcal{X})$, we abbreviate $\rho f = \int_{\mathcal{X}} f(x) \rho(\mathrm{d}x)$ (i.e., we use ρ as an operator). Moreover, we define the $L_p(\rho)$-norm of f as $\|f\|_{p,\rho}^p = \int_{\mathcal{X}} |f(x)|^p \rho(\mathrm{d}x)$ for $p \geqslant 1$, whereas the L_∞-norm is defined as $\|f\|_\infty = \sup_{x\in\mathcal{X}} f(x)$. Let $\mathcal{D} = \{x_i\}_{i=1}^n \subseteq \mathcal{X}$ we define the $L_p(\rho)$ empirical norm as $\|f\|_{p,\mathcal{D}}^p = \frac{1}{n}\sum_{i=1}^n |f(x_i)|^p$.

Markov Decision Processes A discrete-time Markov Decision Process (MDP, Puterman, 2014) is a 5-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is a measurable set of states, $\mathcal{A}$ is a measurable set of actions, $P: \mathcal{S}\times\mathcal{A} \to \mathscr{P}(\mathcal{S})$ is the transition kernel that for each state-action pair $(s,a) \in \mathcal{S}\times\mathcal{A}$ provides the probability distribution $P(\cdot|s,a)$ of the next state, $R: \mathcal{S}\times\mathcal{A} \to \mathscr{P}(\mathbb{R})$ is the reward distribution $R(\cdot|s,a)$ for performing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, whose expected value is denoted by $r(s,a) = \int_{\mathbb{R}} x R(\mathrm{d}x|s,a)$ and uniformly bounded by $R_{\max} < +\infty$, and $\gamma \in [0,1)$ is the discount factor.

A policy $\pi = (\pi_t)_{t\in\mathbb{N}}$ is a sequence of functions $\pi_t: \mathcal{H}_t \to \mathscr{P}(\mathcal{A})$ mapping a history $H_t = (S_0, A_0, \dots, S_{t-1}, A_{t-1}, S_t)$ of length $t \in \mathbb{N}$ to a probability distribution over $\mathcal{A}$, where $\mathcal{H}_t = (\mathcal{S}\times\mathcal{A})^t \times \mathcal{S}$. If π_t depends only on the last visited state S_t then it is called Markovian, i.e., $\pi_t: \mathcal{S} \to \mathscr{P}(\mathcal{A})$. Moreover, if π_t does not depend on explicitly t it is stationary, in this case we remove the subscript t. We denote with Π the set of Markovian stationary policies. A policy $\pi \in \Pi$ induces a (state-action) transition kernel $P^\pi: \mathcal{S}\times\mathcal{A} \to \mathscr{P}(\mathcal{S}\times\mathcal{A})$, defined for any measurable set $\mathcal{B} \subseteq \mathcal{S}\times\mathcal{A}$ as (Farahmand, 2011):

$$(P^\pi)(\mathcal{B}|s,a) = \int_{\mathcal{S}} P(\mathrm{d}s'|s,a) \int_{\mathcal{A}} \pi(\mathrm{d}a'|s') \delta_{(s',a')}(\mathcal{B}). \quad (1)$$

The *action-value function*, or Q-function, of a policy $\pi \in \Pi$ is the expected discounted sum of the rewards obtained by performing action a in state s and following policy π thereafter $Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t R_t | S_0 = s, A_0 = a\right]$, where $R_t \sim R(\cdot|S_t, A_t)$, $S_{t+1} \sim P(\cdot|S_t, A_t)$, and $A_{t+1} \sim \pi(\cdot|S_{t+1})$ for all $t \in \mathbb{N}$. The *value function* is the expectation of the Q-function over the actions: $V^\pi(s) = \int_{\mathcal{A}} \pi(\mathrm{d}a|s) Q^\pi(s,a)$. Given a distribution $\rho \in \mathscr{P}(\mathcal{S})$, we define the *expected return* as $J^{\rho,\pi}(s) = \int_{\mathcal{S}} \rho(\mathrm{d}s) V^\pi(s)$. The optimal Q-function is given by: $Q^*(s,a)=\sup_{\pi\in\Pi}Q^\pi(s,a)$ for all $(s,a)\in\mathcal{S}\times\mathcal{A}$. A policy π is *greedy* w.r.t. a function $f\in\mathscr{B}(\mathcal{S}\times\mathcal{A})$ if it plays only greedy actions, i.e., $\pi(\cdot|s)\in\mathscr{P}(\operatorname{argmax}_{a\in\mathcal{A}}f(s,a))$. An *optimal policy* $\pi^*\in\Pi$ is any policy greedy w.r.t. Q^*.

Given a policy $\pi\in\Pi$, the *Bellman Expectation Operator* $T^\pi:\mathscr{B}(\mathcal{S}\times\mathcal{A})\to\mathscr{B}(\mathcal{S}\times\mathcal{A})$ and the *Bellman Optimal Operator* $T^*:\mathscr{B}(\mathcal{S}\times\mathcal{A})\to\mathscr{B}(\mathcal{S}\times\mathcal{A})$ are defined for a bounded measurable function $f\in\mathscr{B}(\mathcal{S}\times\mathcal{A})$ and $(s,a)\in\mathcal{S}\times\mathcal{A}$ as (Bertsekas & Shreve, 2004):

$$(T^\pi f)(s,a)=r(s,a)+(P^\pi f)(s,a),$$
$$(T^* f)(s,a)=r(s,a)+\gamma\int_{\mathcal{S}}P(\mathrm{d}s'|s,a)\max_{a'\in\mathcal{A}}f(s',a').$$

Both T^π and T^* are γ-contractions in L_∞-norm and, consequently, they have a unique fixed point, that are the Q-function of policy π ($T^\pi Q^\pi=Q^\pi$) and the optimal Q-function ($T^*Q^*=Q^*$) respectively.

Lipschitz MDPs Let $(\mathcal{X},d_{\mathcal{X}})$ and $(\mathcal{Y},d_{\mathcal{Y}})$ be two metric spaces, a function $f:\mathcal{X}\to\mathcal{Y}$ is called L_f-Lipschitz continuous (L_f-LC), where $L_f\geqslant 0$, if for all $x,x'\in\mathcal{X}$ we have:

$$d_{\mathcal{Y}}(f(x),f(x'))\leqslant L_f d_{\mathcal{X}}(x,x'). \tag{2}$$

Moreover, we define the Lipschitz semi-norm as $\|f\|_L=\sup_{x,x'\in\mathcal{X}:x\neq x'}\frac{d_{\mathcal{Y}}(f(x),f(x'))}{d_{\mathcal{X}}(x,x')}$. For real functions we employ Euclidean distance $d_{\mathcal{Y}}(y,y')=\|y-y'\|_2$, while for probability distributions we use the Kantorovich (L_1-Wasserstein) metric defined for $\mu,\nu\in\mathscr{P}(\mathcal{Z})$ as (Villani, 2008):

$$d_{\mathcal{Y}}(\mu,\nu)=\mathcal{W}_1(\mu,\nu)=\sup_{f:\|f\|_L\leqslant 1}\left|\int_{\mathcal{Z}}f(z)(\mu-\nu)(\mathrm{d}z)\right|. \tag{3}$$

We now introduce the notions of Lipschitz MDP and Lipschitz policy that we will employ in the following (Rachelson & Lagoudakis, 2010; Pirotta et al., 2015).

Assumption 2.1 (Lipschitz MDP). *Let $\mathcal{M}$ be an MDP. $\mathcal{M}$ is called (L_P,L_r)-LC if for all $(s,a),(\overline{s},\overline{a})\in\mathcal{S}\times\mathcal{A}$:*

$$\mathcal{W}_1(P(\cdot|s,a),P(\cdot|\overline{s},\overline{a}))\leqslant L_P d_{\mathcal{S}\times\mathcal{A}}((s,a),(\overline{s},\overline{a})),$$
$$|r(s,a)-r(\overline{s},\overline{a})|\leqslant L_r d_{\mathcal{S}\times\mathcal{A}}((s,a),(\overline{s},\overline{a})).$$

Assumption 2.2 (Lipschitz Policy). *Let $\pi\in\Pi$ be a Markovian stationary policy. π is called L_π-LC if for all $s,\overline{s}\in\mathcal{S}$:*

$$\mathcal{W}_1(\pi(\cdot|s),\pi(\cdot|\overline{s}))\leqslant L_\pi d_{\mathcal{S}}(s,\overline{s}).$$

3. Persisting Actions in MDPs

By the phrase "executing a policy π at persistence k", with $k\in\mathbb{N}_{\geqslant 1}$, we mean the following type of agent-environment interaction. At decision step $t=0$, the agent selects an action according to its policy $A_0\sim\pi(\cdot|S_0)$. Action A_0 is kept fixed, or *persisted*, for the subsequent $k-1$ decision steps, i.e., actions $A_1,\dots,A_{k-1}$ are all equal to A_0. Then, at decision step $t=k$, the agent queries again the policy $A_k\sim\pi(\cdot|S_k)$ and persists action A_k for the subsequent $k-1$ decision steps and so on. In other words, the agent employs its policy only at decision steps t that are integer multiples of the persistence k ($t \bmod k=0$). Clearly, the usual execution of π corresponds to persistence 1.

3.1. Duality of Action Persistence

Unsurprisingly, the execution of a Markovian stationary policy π at persistence $k>1$ produces a behavior that, in general, cannot be represented by executing any Markovian stationary policy at persistence 1. Indeed, at any decision step t, such a policy needs to remember which action was taken at the previous decision step $t-1$ (thus it is non-Markovian with memory 1) and has to understand whether to select a new action based on t (so it is non-stationary).

Definition 3.1 (k-persistent policy). *Let $\pi\in\Pi$ be a Markovian stationary policy. For any $k\in\mathbb{N}_{\geqslant 1}$, the* k-persistent policy *induced by π is a non–Markovian non–stationary policy, defined for any measurable set $\mathcal{B}\subseteq\mathcal{A}$ and $t\in\mathbb{N}$ as:*

$$\pi_{t,k}(\mathcal{B}|H_t)=\begin{cases}\pi(\mathcal{B}|S_t) & \text{if } t \bmod k=0\\ \delta_{A_{t-1}}(\mathcal{B}) & \text{otherwise}\end{cases}. \tag{4}$$

Moreover, we denote with $\Pi_k=\{(\pi_{t,k})_{t\in\mathbb{N}}:\pi\in\Pi\}$ the set of the k-persistent policies.

Clearly, for $k=1$ we recover policy π as we always satisfy the condition $t \bmod k=0$ i.e., $\pi=\pi_{t,1}$ for all $t\in\mathbb{N}$. We refer to this interpretation of action persistence as *policy view*.

A different perspective towards action persistence consists in looking at the effect of the original policy π in a suitably modified MDP. To this purpose, we introduce the (state-action) persistent transition probability kernel $P^\delta:\mathcal{S}\times\mathcal{A}\to\mathscr{P}(\mathcal{S}\times\mathcal{A})$ defined for any measurable set $\mathcal{B}\subseteq\mathcal{S}\times\mathcal{A}$ as:

$$(P^\delta)(\mathcal{B}|s,a)=\int_{\mathcal{S}}P(\mathrm{d}s'|s,a)\delta_{(s',a)}(\mathcal{B}). \tag{5}$$

The crucial difference between P^π and P^δ is that the former samples the action a' to be executed in the next state s' according to π, whereas the latter replicates in state s' action a. We are now ready to define the k-persistent MDP.

Definition 3.2 (k-persistent MDP). *Let $\mathcal{M}$ be an MDP. For any $k\in\mathbb{N}_{\geqslant 1}$, the* k-persistent MDP *is the following MDP $\mathcal{M}_k=\left(\mathcal{S},\mathcal{A},P_k,R_k,\gamma^k\right)$, where P_k and R_k are the k-persistent transition model and reward distribution respectively, defined for any measurable sets $\mathcal{B}\subseteq\mathcal{S}$, $\mathcal{C}\subseteq\mathbb{R}$ and state-action pair $(s,a)\in\mathcal{S}\times\mathcal{A}$ as:*

$$P_k(\mathcal{B}|s,a)=\left((P^\delta)^{k-1}P\right)(\mathcal{B}|s,a), \tag{6}$$

$$R_k(\mathcal{C}|s,a)=\sum_{i=0}^{k-1}\gamma^i\left((P^\delta)^i R\right)(\mathcal{C}|s,a), \tag{7}$$

and $r_k(s,a)=\int_{\mathbb{R}}xR_k(\mathrm{d}x|s,a)=\sum_{i=0}^{k-1}\gamma^i\left((P^\delta)^i r\right)(s,a)$ is the expected reward, uniformly bounded by $R_{\max}\frac{1-\gamma^k}{1-\gamma}$.

The k-persistent transition model P_k keeps action a fixed for $k-1$ steps while making the state evolve according to P.

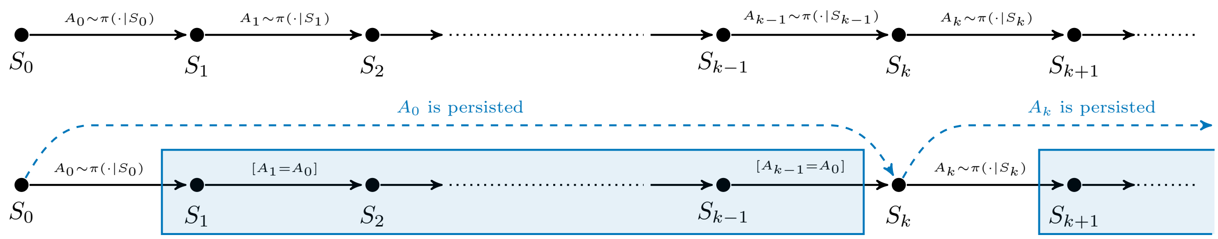

Figure 1. Agent-environment interaction without (top) and with (bottom) action persistence, highlighting duality. The transition generated by the k-persistent MDP $\mathcal{M}_k$ is the cyan dashed arrow, while the actions played by the k-persistent policy are inside the cyan rectangle.

Similarly, the k-persistent reward R_k provides the cumulative discounted reward over k steps in which a is persisted. We define the transition kernel P_k^π, analogously to P^π, as in Equation (1). Clearly, for $k=1$ we recover the base MDP, i.e., $\mathcal{M}=\mathcal{M}_1$.[2] Therefore, executing policy π in $\mathcal{M}_k$ at persistence 1 is equivalent to executing policy π at persistence k in the original MDP $\mathcal{M}$. We refer to this interpretation of persistence as *environment view* (Figure 1). Thus, solving the base MDP $\mathcal{M}$ in the space of k-persistent policies Π_k (Definition 3.1), thanks to this *duality*, is equivalent to solving the k-persistent MDP $\mathcal{M}_k$ (Definition 3.2) in the space of Markovian stationary policies Π.

It is worth noting that the persistence $k \in \mathbb{N}_{\geqslant 1}$ can be seen as an *environmental parameter* (affecting P, R, and γ), which can be externally configured with the goal to improve the learning process for the agent. In this sense, the MDP $\mathcal{M}_k$ can be seen as a Configurable Markov Decision Process with parameter $k \in \mathbb{N}_{\geqslant 1}$ (Metelli et al., 2018; 2019).

Furthermore, a persistence of k induces a k-persistent MDP $\mathcal{M}_k$ with smaller discount factor γ^k. Therefore, the effective horizon in $\mathcal{M}_k$ is $\frac{1}{1-\gamma^k} < \frac{1}{1-\gamma}$. Interestingly, the end effect of persisting actions is similar to reducing the planning horizon, by explicitly reducing the discount factor of the task (Petrik & Scherrer, 2008; Jiang et al., 2016) or setting a maximum trajectory length (Farahmand et al., 2016).

3.2. Persistent Bellman Operators

When executing policy π at persistence k in the base MDP $\mathcal{M}$, we can evaluate its performance starting from any state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$, inducing a Q-function that we denote with Q_k^π and call *k-persistent action-value function* of π. Thanks to duality, Q_k^π is also the action-value function of policy π when executed in the k-persistent MDP $\mathcal{M}_k$. Therefore, Q_k^π is the fixed point of the Bellman Expectation Operator of $\mathcal{M}_k$, i.e., the operator defined for any $f \in \mathscr{B}(\mathcal{S} \times \mathcal{A})$ as $(T_k^\pi f)(s,a) = r_k(s,a) + \gamma^k (P_k^\pi f)(s,a)$, that we call *$k$-persistent Bellman Expectation Operator*. Similarly, again thanks to duality, the optimal Q-function in the space of k-persistent policies Π_k, denoted by Q_k^* and called *k-persistent optimal action-value function*, corresponds to the optimal Q-function of the k-persistent MDP, i.e., $Q_k^*(s,a) = \sup_{\pi \in \Pi} Q_k^\pi(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. As a consequence, Q_k^* is the fixed point of the Bellman Optimal Operator of $\mathcal{M}_k$, defined for $f \in \mathscr{B}(\mathcal{S} \times \mathcal{A})$ as $(T_k^* f)(s,a) = r_k(s,a) + \gamma^k \int_{\mathcal{S}} P_k(\mathrm{d}s'|s,a) \max_{a' \in \mathcal{A}} f(s',a')$, that we call *$k$-persistent Bellman Optimal Operator*. Clearly, both T_k^π and T_k^* are γ^k-contractions in L_∞-norm.

We now prove that the k-persistent Bellman operators are obtained as composition of the base operators T^π and T^*.

Theorem 3.1. *Let $\mathcal{M}$ be an MDP, $k \in \mathbb{N}_{\geqslant 1}$ and $\mathcal{M}_k$ be the k-persistent MDP. Let $\pi \in \Pi$ be a Markovian stationary policy. Then, T_k^π and T_k^* can be expressed as:*

$$T_k^\pi = \left(T^\delta\right)^{k-1} T^\pi \quad \text{and} \quad T_k^* = \left(T^\delta\right)^{k-1} T^*, \tag{8}$$

where $T^\delta : \mathscr{B}(\mathcal{S} \times \mathcal{A}) \to \mathscr{B}(\mathcal{S} \times \mathcal{A})$ is the Bellman Persistent Operator, *defined for $f \in \mathscr{B}(\mathcal{S} \times \mathcal{A})$ and $(s,a) \in \mathcal{S} \times \mathcal{A}$:*

$$\left(T^\delta f\right)(s,a) = r(s,a) + \gamma \left(P^\delta f\right)(s,a). \tag{9}$$

The fixed point equations for the k-persistent Q-functions become: $Q_k^\pi = \left(T^\delta\right)^{k-1} T^\pi Q_k^\pi$ and $Q_k^* = \left(T^\delta\right)^{k-1} T^* Q_k^*$.

4. Bounding the Performance Loss

Learning in the space of k-persistent policies Π_k can only lower the performance of the optimal policy, i.e., $Q^*(s,a) \geqslant Q_k^*(s,a)$ for $k \in \mathbb{N}_{\geqslant 1}$. The goal of this section is to bound $\|Q^* - Q_k^*\|_{p,\rho}$ as a function of the persistence $k \in \mathbb{N}_{\geqslant 1}$. To this purpose, we focus on $\|Q^\pi - Q_k^\pi\|_{p,\rho}$ for a fixed policy $\pi \in \Pi$, since denoting with π^* an optimal policy of $\mathcal{M}$ and with π_k^* an optimal policy of $\mathcal{M}_k$, we have that:

$$Q^* - Q_k^* = Q^{\pi^*} - Q_k^{\pi_k^*} \leqslant Q^{\pi^*} - Q_k^{\pi^*},$$

since $Q_k^{\pi_k^*}(s,a) \geqslant Q_k^{\pi^*}(s,a)$. We start with the following result which makes no assumption about the structure of the MDP and then we particularize it for the Lipschitz MDPs.

Theorem 4.1. *Let $\mathcal{M}$ be an MDP and $\pi \in \Pi$ be a Markovian stationary policy. Let $\mathcal{Q}_k = \{\left(T^\delta\right)^{k-2-l} T^\pi Q_k^\pi : l \in \{0, \dots, k-2\}\}$ and for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ let us define:*

$$d_{\mathcal{Q}_k}^\pi(s,a) = \sup_{f \in \mathcal{Q}_k} \left| \int_{\mathcal{S}} \int_{\mathcal{A}} \left(P^\pi(\mathrm{d}s', \mathrm{d}a'|s,a) - P^\delta(\mathrm{d}s', \mathrm{d}a'|s,a)\right) f(s',a') \right|.$$

[2] If $\mathcal{M}$ is the base MDP $\mathcal{M}_{\Delta t_0}$, the k–persistent MDP $\mathcal{M}_k$ corresponds to $\mathcal{M}_{k\Delta t_0}$. We remove the subscript Δt_0 for brevity.

Then, for any $\rho \in \mathscr{P}(\mathcal{S}\times\mathcal{A})$, $p \geqslant 1$, and $k \in \mathbb{N}_{\geqslant 1}$, it holds that:

$$\|Q^\pi - Q_k^\pi\|_{p,\rho} \leqslant \frac{\gamma(1-\gamma^{k-1})}{(1-\gamma)(1-\gamma^k)} \|d_{\mathcal{Q}_k}^\pi\|_{p,\eta_k^{\rho,\pi}},$$

where $\eta_k^{\rho,\pi} \in \mathscr{P}(\mathcal{S}\times\mathcal{A})$ is a probability measure defined for any measurable set $\mathcal{B}\subseteq\mathcal{S}\times\mathcal{A}$ as:

$$\eta_k^{\rho,\pi}(\mathcal{B}) = \frac{(1-\gamma)(1-\gamma^k)}{\gamma(1-\gamma^{k-1})} \sum_{\substack{i\in\mathbb{N} \\ i \bmod k \neq 0}} \gamma^i \left(\rho (P^\pi)^{i-1}\right)(\mathcal{B}).$$

The bound shows that the Q-function difference depends on the discrepancy $d_{\mathcal{Q}_k}^\pi$ between the transition-kernel P^π and the corresponding persistent version P^δ, which is a form of *integral probability metric* (Müller, 1997), defined in terms of the set $\mathcal{Q}_k$. This term is averaged with the distribution $\eta_k^{\rho,\pi}$, which encodes the (discounted) probability of visiting a state-action pair, ignoring the visitations made at decision steps i that are multiple of the persistence k. Indeed, in those steps, we play policy π regardless of whether persistence is used.[3] The dependence on k is represented in the term $\frac{1-\gamma^{k-1}}{1-\gamma^k}$. When $k\to 1$ this term displays a linear growth in k, being asymptotic to $(k-1)\log\frac{1}{\gamma}$, and, clearly, vanishes for $k=1$. Instead, when $k\to\infty$ this term tends to 1.

If no structure on the MDP/policy is enforced, the dissimilarity term $d_{\mathcal{Q}_k}^\pi$ may become large enough to make the bound vacuous, i.e., larger than $\frac{\gamma R_{\max}}{1-\gamma}$, even for $k=2$ (see Appendix B.1). Intuitively, since the persistence will execute old actions in new states, we need to guarantee that the environment state changes slowly w.r.t. to time and the policy must play similar actions in similar states. This means that if an action is good in a state, it will also be almost good for states encountered in the near future. Although the condition on π is directly enforced by Assumption 2.2, we need a new notion of regularity over time for the MDP.

Assumption 4.1. *Let $\mathcal{M}$ be an MDP. $\mathcal{M}$ is L_T–*Time-Lipschitz Continuous *(L_T–TLC) if for all $(s,a)\in\mathcal{S}\times\mathcal{A}$:*

$$\mathcal{W}_1(P(\cdot|s,a),\delta_s) \leqslant L_T. \tag{10}$$

This assumption requires that the Kantorovich distance between the distribution of the next state s' and the deterministic distribution centered in the current state s is bounded by L_T, i.e., the system does not evolve "too fast" (see Appendix B.3). We can now state the following result.

Theorem 4.2. *Let $\mathcal{M}$ be an MDP and $\pi\in\Pi$ be a Markovian stationary policy. Under Assumptions 2.1, 2.2, and 4.1, if $\gamma\max\{L_P+1, L_P(1+L_\pi)\}<1$ and if $\rho(s,a) = \rho_\mathcal{S}(s)\pi(a|s)$ with $\rho_\mathcal{S}\in\mathscr{P}(\mathcal{S})$, then for any $k\in\mathbb{N}_{\geqslant 1}$:*

$$\|d_{\mathcal{Q}_k}^\pi\|_{p,\eta_k^{\rho,\pi}} \leqslant L_{\mathcal{Q}_k}\left[(L_\pi+1)L_T + \sigma_p\right].$$

where $\sigma_p^p = \sup_{s\in\mathcal{S}} \int_\mathcal{A}\int_\mathcal{A} d_\mathcal{A}(a,a')^p \pi(\mathrm{d}a|s)\pi(\mathrm{d}a'|s)$, and $L_{\mathcal{Q}_k} = \frac{L_T}{1-\gamma\max\{L_P+1, L_P(1+L_\pi)\}}$.

Thus, the dissimilarity $d_{\mathcal{Q}_k}^\pi$ between P^π and P^δ can be bounded with four terms. i) $L_{\mathcal{Q}_k}$ is (an upper-bound of) the Lipschitz constant of the functions in the set $\mathcal{Q}_k$. Indeed, under Assumptions 2.1 and 2.2 we can reduce the dissimilatity term to the Kantorivich distance (Lemma A.5):

$$d_{\mathcal{Q}_k}^\pi(s,a) \leqslant L_{\mathcal{Q}_k}\mathcal{W}_1\left(P^\pi(\cdot|s,a), P^\delta(\cdot|s,a)\right).$$

ii) $(L_\pi+1)$ accounts for the Lipschitz continuity of the policy, i.e., policies that prescribe similar actions in similar states have a small value of this quantity. iii) L_T represents the speed at which the environment state evolves over time. iv) σ_p denotes the average distance (in L_p-norm) between two actions prescribed by the policy in the same state. This term is zero for deterministic policies and can be related to the maximum policy variance (Lemma A.6). A more detailed discussion on the conditions requested in Theorem 4.2 is reported in Appendix B.4.

5. Persistent Fitted Q-Iteration

In this section, we introduce an extension of Fitted Q-Iteration (FQI, Ernst et al., 2005) that employs the notion of persistence.[4] *Persisted Fitted Q-Iteration* (PFQI(k)) takes as input a *target persistence* $k\in\mathbb{N}_{\geqslant 1}$ and its goal is to approximate the k-persistent optimal action-value function Q_k^*. Starting from an initial estimate $Q^{(0)}$, at each iteration we compute the next estimate $Q^{(j+1)}$ by performing an approximate application of k-persistent Bellman optimal operator to the previous estimate $Q^{(j)}$, i.e., $Q^{(j+1)}\approx T_k^* Q^{(j)}$. In practice, we have two sources of approximation in this process: i) the representation of the Q-function; ii) the estimation of the k-persistent Bellman optimal operator. (i) comes from the necessity of using functional space $\mathcal{F}\subset\mathscr{B}(\mathcal{S}\times\mathcal{A})$ to represent $Q^{(j)}$ when dealing with continuous state spaces. (ii) derives from the approximate computation of T_k^* which needs to be estimated from samples.

Clearly, with samples collected in the k-persistent MDP $\mathcal{M}_k$, the process described above reduces to the standard FQI. However, our algorithm needs to be able to estimate Q_k^* for different values of k, using the same dataset of samples collected in the base MDP $\mathcal{M}$ (at persistence 1).[5] For this purpose, we can exploit the decomposition $T_k^* = (T^\delta)^{k-1}T^*$ of Theorem 3.1 to reduce a single application of T_k^* to a sequence of k applications of the 1-persistent operators. Specifically, at each iteration j with $j \bmod k = 0$, given the current estimate $Q^{(j)}$, we need to perform (in this order) a single application of T^* followed by $k-1$ applica-

[3] $\eta_k^{\rho,\pi}$ resambles the γ-discounted state-action distribution (Sutton et al., 1999a), but ignoring the decision steps multiple of k.

[4] From now on, we assume that $|\mathcal{A}|<+\infty$.

[5] In real–world cases, we might be unable to interact with the physical system to collect samples for any persistence k of interest.

Algorithm 1 Persistent Fitted Q-Iteration PFQI(k).

Input: k persistence, J number of iterations ($J \bmod k=0$), $Q^{(0)}$ initial action-value function, $\mathcal{F}$ functional space, $\mathcal{D}=\{(S_i,A_i,S'_i,R_i)\}_{i=1}^n$ batch samples
Output: greedy policy $\pi^{(J)}$
for $j=0,\dots,J-1$ **do**
  if $j \bmod k=0$ **then** (Phase 1)
    $Y_i^{(j)}=\widehat{T}^*Q^{(j)}(S_i,A_i), \quad i=1,\dots,n$
  else
    $Y_i^{(j)}=\widehat{T}^\delta Q^{(j)}(S_i,A_i), \quad i=1,\dots,n$
  end if
  $Q^{(j+1)}\in\operatorname{arginf}_{f\in\mathcal{F}}\|f-Y^{(j)}\|_{2,\mathcal{D}}^2$ (Phase 2)
end for
$\pi^{(J)}(s)\in\operatorname{argmax}_{a\in\mathcal{A}}Q^{(J)}(s,a), \quad \forall s\in\mathcal{S}$ (Phase 3)

tions of T^δ, leading to the sequence of approximations:

$$Q^{(j+1)}\approx\begin{cases}T^*Q^{(j)} & \text{if } j \bmod k=0\\ T^\delta Q^{(j)} & \text{otherwise}\end{cases}. \tag{11}$$

In order to estimate the Bellman operators, we have access to a dataset $\mathcal{D}=\{(S_i,A_i,S'_i,R_i)\}_{i=1}^n$ collected in the base MDP $\mathcal{M}$, where $(S_i,A_i)\sim\nu$, $S'_i\sim P(\cdot|S_i,A_i)$, $R_i\sim R(\cdot|S_i,A_i)$, and $\nu\in\mathscr{P}(\mathcal{S}\times\mathcal{A})$ is a sampling distribution. We employ $\mathcal{D}$ to compute the *empirical Bellman operators* (Farahmand, 2011) defined for $f\in\mathscr{B}(\mathcal{S}\times\mathcal{A})$ as:

$$(\widehat{T}^*f)(S_i,A_i)=R_i+\gamma\max_{a\in\mathcal{A}}f(S'_i,a) \quad i=1,\dots,n$$
$$(\widehat{T}^\delta f)(S_i,A_i)=R_i+\gamma f(S'_i,A_i) \quad i=1,\dots,n.$$

These operators are unbiased conditioned to $\mathcal{D}$ (Farahmand, 2011): $\mathbb{E}[(\widehat{T}^*f)(S_i,A_i)|S_i,A_i]=(T^*f)(S_i,A_i)$ and $\mathbb{E}[(\widehat{T}^\delta f)(S_i,A_i)|S_i,A_i]=(T^\delta f)(S_i,A_i)$.

The pseudocode of PFQI(k) is summarized in Algorithm 1. At each iteration $j=0,\dots J-1$, we first compute the target values $Y^{(j)}$ by applying the empirical Bellman operators, $\widehat{T}^*$ or $\widehat{T}^\delta$, on the current estimate $Q^{(j)}$ (Phase 1). Then, we project the target $Y^{(j)}$ onto the functional space $\mathcal{F}$ by solving the least squares problem (Phase 2):

$$Q^{(j+1)}\in\operatorname*{arginf}_{f\in\mathcal{F}}\left\|f-Y^{(j)}\right\|_{2,\mathcal{D}}^2=\frac{1}{n}\sum_{i=1}^n\left|f(S_i,A_i)-Y_i^{(j)}\right|^2.$$

Finally, we compute the approximation of the optimal policy $\pi^{(J)}$, i.e., the greedy policy w.r.t. $Q^{(J)}$ (Phase 3).

5.1. Theoretical Analysis

In this section, we present the computational complexity analysis and the study of the error propagation in PFQI(k).

Computational Complexity The computational complexity of PFQI(k) decreases monotonically with the persistence k. Whenever applying $\widehat{T}^\delta$, we need a single evaluation of $Q^{(j)}$, while $|\mathcal{A}|$ evaluations are needed for $\widehat{T}^*$ due to the max over $\mathcal{A}$. Thus, the overall complexity of J iterations of PFQI(k) with n samples, disregarding the cost of regression and assuming that a single evaluation of $Q^{(j)}$ takes constant time, is given by $\mathcal{O}(Jn(1+(|\mathcal{A}|-1)/k))$ (Proposition A.1).

Error Propagation We now consider the error propagation in PFQI(k). Given the sequence of Q-functions estimates $(Q^{(j)})_{j=0}^J\subset\mathcal{F}$ produced by PFQI(k), we define the approximation error at each iteration $j=0,\dots,J-1$ as:

$$\epsilon^{(j)}=\begin{cases}T^*Q^{(j)}-Q^{(j+1)} & \text{if } j \bmod k=0\\ T^\delta Q^{(j)}-Q^{(j+1)} & \text{otherwise}\end{cases}. \tag{12}$$

The goal of this analysis is to bound the distance between the k–persistent optimal Q-function Q_k^* and the Q-function $Q_k^{\pi^{(J)}}$ of the greedy policy $\pi^{(J)}$ w.r.t. $Q^{(J)}$, after J iterations of PFQI(k). The following result extends Theorem 3.4 of Farahmand (2011) to account for action persistence.

Theorem 5.1 (Error Propagation for PFQI(k))**.** *Let $p\geqslant1$, $k\in\mathbb{N}_{\geqslant1}$, $J\in\mathbb{N}_{\geqslant1}$ with $J \bmod k=0$ and $\rho\in\mathscr{P}(\mathcal{S}\times\mathcal{A})$. Then for any sequence $(Q^{(j)})_{j=0}^J\subset\mathcal{F}$ uniformly bounded by $Q_{\max}\leqslant\frac{R_{\max}}{1-\gamma}$, the corresponding $(\epsilon^{(j)})_{j=0}^{J-1}$ defined in Equation (12) and for any $r\in[0,1]$ and $q\in[1,+\infty]$ it holds that:*

$$\left\|Q_k^*-Q_k^{\pi^{(J)}}\right\|_{p,\rho}\leqslant\frac{2\gamma^k}{(1-\gamma)(1-\gamma^k)}\left[\frac{2}{1-\gamma}\gamma^{\frac{J}{p}}R_{\max}\right.$$
$$\left.+C_{\mathrm{VI},\rho,\nu}^{\frac{1}{2p}}(J,r,q)\mathcal{E}^{\frac{1}{2p}}(\epsilon^{(0)},\dots,\epsilon^{(J-1)};r,q)\right].$$

The expression of $C_{\mathrm{VI},\rho,\nu}(J;r,q)$ and $\mathcal{E}(\cdot;r,q)$ can be found in Appendix A.3.

We immediately observe that for $k=1$ we recover Theorem 3.4 of Farahmand (2011). The term $C_{\mathrm{VI},\rho,\nu}(J;r,q)$ is defined in terms of suitable *concentrability coefficients* (Definition A.1) and encodes the distribution shift between the sampling distribution ν and the one induced by the greedy policy sequence $(\pi^{(j)})_{j=0}^J$ encountered along the execution of PFQI(k). $\mathcal{E}(\cdot;r,q)$ incorporates the approximation errors $(\epsilon^{(j)})_{j=0}^{J-1}$. In principle, it is hard to compare the values of these terms for different persistences k since both the greedy policies and the regression problems are different. Nevertheless, it is worth noting that the multiplicative term $\frac{\gamma^k}{1-\gamma^k}$ decreases in $k\in\mathbb{N}_{\geqslant1}$. Thus, other things being equal, the bound value decreases when increasing the persistence.

Thus, the trade-off in the choice of control frequency, which motivates action persistence, can now be stated more formally. We aim at finding the persistence $k\in\mathbb{N}_{\geqslant1}$ that, for a fixed J, allows learning a policy $\pi^{(J)}$ whose Q-function $Q_k^{\pi^{(J)}}$ is the closest to Q^*. Consider the decomposition:

$$\left\|Q^*-Q_k^{\pi^{(J)}}\right\|_{p,\rho}\leqslant\|Q^*-Q_k^*\|_{p,\rho}+\left\|Q_k^*-Q_k^{\pi^{(J)}}\right\|_{p,\rho}.$$

The term $\|Q^*-Q_k^*\|_{p,\rho}$ accounts for the performance degradation due to action persistence: it is algorithm–independent, and it increases in k (Theorem 4.1). Instead, the second term $\|Q_k^*-Q_k^{\pi^{(J)}}\|_{p,\rho}$ decreases with k and depends on the algo-

Algorithm 2 Heuristic Persistence Selection.

Input: batch samples $\mathcal{D}=\{(S_0^i,A_0^i,\dots,S_{H_i-1}^i,A_{H_i-1}^i,S_{H_i}^i)\}_{i=1}^m$, set of persistences $\mathcal{K}$, set of Q-function $\{Q_k:k\in\mathcal{K}\}$, regressor `Reg`
Output: approximately optimal persistence $\widetilde{k}$

for $k\in\mathcal{K}$ do
  $\widehat{J}_k^\rho=\frac{1}{m}\sum_{i=1}^m V_k(S_0^i)$
  Use the `Reg` to get an estimate $\widetilde{Q}_k$ of $T_k^* Q_k$
  $\|\widetilde{Q}_k-Q_k\|_{1,\mathcal{D}}=\frac{1}{\sum_{i=1}^m H_i}\sum_{i=1}^m\sum_{t=0}^{H_i-1}|\widetilde{Q}_k(S_t^i,A_t^i)-Q_k(S_t^i,A_t^i)|$
end for
$\widetilde{k}\in\operatorname{argmax}_{k\in\mathcal{K}} B_k=\widehat{J}_k^\rho-\frac{1}{1-\gamma^k}\|\widetilde{Q}_k-Q_k\|_{1,\mathcal{D}}$.

rithm (Theorem 5.1). Unfortunately, optimizing their sum is hard since the individual bounds contain terms that are not known in general (e.g., Lipschitz constants, $\epsilon^{(j)}$). The next section proposes heuristics to overcome this problem.

6. Persistence Selection

In this section, we discuss how to select a persistence k in a set $\mathcal{K}\subset\mathbb{N}_{\geqslant 1}$ of candidate persistences, when we are given a set of estimated Q-functions: $\{Q_k:k\in\mathcal{K}\}$.[6] Each Q_k induces a greedy policy π_k. Our goal is to find the persistence $k\in\mathcal{K}$ such that π_k has the maximum expected return in the corresponding k–persistent MDP $\mathcal{M}_k$:

$$k^*\in\operatorname*{argmax}_{k\in\mathcal{K}} J_k^{\rho,\pi_k},\quad \rho\in\mathscr{P}(\mathcal{S}). \tag{13}$$

In principle, we could execute π_k in $\mathcal{M}_k$ to get an estimate of J_k^{ρ,π_k} and employ it to select the persistence k. However, in the batch setting, further interactions with the environment might be not allowed. On the other hand, directly using the estimated Q-function Q_k is inappropriate, since we need to take into account how well Q_k approximates $Q_k^{\pi_k}$. This trade–off is encoded in the following result, which makes use of the *expected Bellman residual*.

Lemma 6.1. *Let $Q\in\mathscr{B}(\mathcal{S}\times\mathcal{A})$ and π be a greedy policy w.r.t. Q. Let $J^\rho=\int\rho(\mathrm{d}s)V(s)$, with $V(s)=\max_{a\in\mathcal{A}}Q(s,a)$ for all $s\in\mathcal{S}$. Then, for any $k\in\mathbb{N}_{\geqslant 1}$, it holds that:*

$$J_k^{\rho,\pi}\geqslant J^\rho-\frac{1}{1-\gamma^k}\|T_k^*Q-Q\|_{1,\eta^{\rho,\pi}}, \tag{14}$$

where $\eta^{\rho,\pi}=(1-\gamma^k)\rho\pi\left(\mathrm{Id}-\gamma^k P_k^\pi\right)^{-1}$, is the γ-discounted stationary distribution induced by policy π and distribution ρ in MDP $\mathcal{M}_k$.

To get a usable bound, we need to make some simplifications. First, we assume that $\mathcal{D}\sim\nu$ is composed of m *trajectories*, i.e., $\mathcal{D}=\{(S_0^i,A_0^i,\dots,S_{H_i-1}^i,A_{H_i-1}^i,S_{H_i}^i)\}_{i=1}^m$, where H_i is the trajectory length and the initial states are sampled as $S_0^i\sim\rho$. In this way, J^ρ can be estimated from samples as $\widehat{J}^\rho=\frac{1}{m}\sum_{i=1}^m V(S_0^i)$. Second, since we are unable to compute expectations over $\eta^{\rho,\pi}$, we replace it with the sampling distribution ν.[7] Lastly, estimating the expected Bellman residual is problematic since its empirical version is biased (Antos et al., 2008). Thus, we resort to an approach similar to (Farahmand & Szepesvári, 2011), assuming to have a regressor `Reg` able to output an approximation $\widetilde{Q}_k$ of T_k^*Q. In this way, we replace $\|T_k^*Q-Q\|_{1,\nu}$ with $\|\widetilde{Q}_k-Q\|_{1,\mathcal{D}}$ (details in Appendix C). In practice, we set $Q=Q^{(J)}$ and we obtain $\widetilde{Q}_k$ running PFQI(k) for k additional iterations, setting $\widetilde{Q}_k=Q^{(J+k)}$. Thus, the procedure (Algorithm 2) reduces to optimizing the index:

$$\widetilde{k}\in\operatorname*{argmax}_{k\in\mathcal{K}} B_k=\widehat{J}_k^\rho-\frac{1}{1-\gamma^k}\left\|\widetilde{Q}_k-Q_k\right\|_{1,\mathcal{D}}. \tag{15}$$

7. Related Works

In this section, we revise the works connected to persistence, focusing on continuous–time RL and temporal abstractions.

Continuous–time RL Among the first attempts to extend value–based RL to the continuous–time domain there is *advantage updating* (Bradtke & Duff, 1994), in which Q-learning (Watkins, 1989) is modified to account for infinitesimal control timesteps. Instead of storing the Q-function, the *advantage function* $A(s,a)=Q(s,a)-V(s)$ is recorder. The continuous time is addressed in Baird (1994) by means of the semi-Markov decision processes (Howard, 1963) for finite–state problems. The optimal control literature has extensively studied the solution of the Hamilton-Jacobi-Bellman equation, i.e., the continuous–time counterpart of the Bellman equation, when assuming the knowledge of the environment (Bertsekas, 2005; Fleming & Soner, 2006). The model–free case has been tackled by resorting to time (and space) discretizations (Peterson, 1993), with also convergence guarantees (Munos, 1997; Munos & Bourgine, 1997), and coped with function approximation (Dayan & Singh, 1995; Doya, 2000). More recently, the sensitivity of deep RL algorithm to the time discretization has been analyzed in Tallec et al. (2019), proposing an adaptation of advantage updating to deal with small time scales, that can be employed with deep architectures.

Temporal Abstractions The notion of action persistence can be seen as a form of *temporal abstraction* (Sutton et al., 1999b; Precup, 2001). Temporally extended actions have been extensively used in the hierarchical RL literature to model different time resolutions (Singh, 1992a;b), subgoals (Dietterich, 1998), and combined with the actor–critic architectures (Bacon et al., 2017). Persisting an action is a particular instance of a semi-Markov *option*, always lasting k steps. According to the flat option representation (Precup, 2001), we have as initiation set $\mathcal{I}=\mathcal{S}$ the set of all states, as internal policy the policy that plays deter-

[6]For instance, the Q_k can be obtained by executing PFQI(k) with different persistences $k\in\mathcal{K}$.

[7]This introduces a bias that is negligible if $\|\eta^{\rho,\pi}/\nu\|_\infty\approx 1$ (details in Appendix C.1).

Table 1. Results of PFQI in different environments and persistences. For each persistence k, we report the sample mean and the standard deviation of the estimated return of the last policy $\widehat{J}_k^{\rho,\pi_k}$. For each environment, the persistence with highest average performance and the ones not statistically significantly different from that one (Welch's t-test with $p<0.05$) are in bold. The last column reports the mean and the standard deviation of the performance loss δ between the optimal persistence and the one selected by the index B_k (Equation (15)).

Environment	Expected return at persistence k ($\widehat{J}_k^{\rho,\pi_k}$, mean $\pm$ std)							Performance loss (δ mean $\pm$ std)
	$k=1$	$k=2$	$k=4$	$k=8$	$k=16$	$k=32$	$k=64$	
Cartpole	169.9 ± 5.8	176.5 ± 5.0	$\mathbf{239.5\pm4.4}$	10.0 ± 0.0	9.8 ± 0.0	9.8 ± 0.0	9.8 ± 0.0	0.0 ± 0.0
MountainCar	-111.1 ± 1.5	-103.6 ± 1.6	-97.2 ± 2.0	$\mathbf{-93.6\pm2.1}$	$\mathbf{-94.4\pm1.8}$	$\mathbf{-92.4\pm1.5}$	-136.7 ± 0.9	1.88 ± 0.85
LunarLander	-165.8 ± 50.4	-12.8 ± 4.7	$\mathbf{1.2\pm3.6}$	$\mathbf{2.0\pm3.4}$	-44.1 ± 6.9	-122.8 ± 10.5	-121.2 ± 8.6	2.12 ± 4.21
Pendulum	$\mathbf{-116.7\pm16.7}$	$\mathbf{-113.1\pm16.3}$	$\mathbf{-153.8\pm23.0}$	-283.1 ± 18.0	-338.9 ± 16.3	-364.3 ± 22.1	-377.2 ± 21.7	3.52 ± 0.0
Acrobot	-89.2 ± 1.1	$\mathbf{-82.5\pm1.7}$	$\mathbf{-83.4\pm1.3}$	-122.8 ± 1.3	-266.2 ± 1.9	-287.3 ± 0.3	-286.7 ± 0.6	0.80 ± 0.27
Swimmer	21.3 ± 1.1	$\mathbf{25.2\pm0.8}$	$\mathbf{25.0\pm0.5}$	$\mathbf{24.0\pm0.3}$	22.4 ± 0.3	12.8 ± 1.2	14.0 ± 0.2	2.69 ± 1.71
Hopper	58.6 ± 4.8	61.9 ± 4.2	62.2 ± 1.7	59.7 ± 3.1	60.8 ± 1.0	66.7 ± 2.7	$\mathbf{73.4\pm1.2}$	5.33 ± 2.32
Walker 2D	61.6 ± 5.5	37.6 ± 4.0	62.7 ± 18.2	$\mathbf{80.8\pm6.6}$	$\mathbf{102.1\pm19.3}$	$\mathbf{91.5\pm13.0}$	$\mathbf{97.2\pm17.6}$	5.10 ± 3.74

ministically the action taken when the option was initiated, i.e., the k–persistent policy, and as termination condition whether k timesteps have passed after the option started, i.e., $\beta(H_t)=\mathbb{1}_{\{t \bmod k=0\}}$. Interestingly, in Mann et al. (2015) an approximate value iteration procedure for options lasting at least a given number of steps is proposed and analyzed. This approach shares some similarities with action persistence. Nevertheless, we believe that the option framework is more general and usually the time abstractions are related to the semantic of the tasks, rather than based on the modification of the control frequency, like action persistence.

8. Experimental Evaluation

In this section, we provide the empirical evaluation of PFQI, with the threefold goal: i) proving that a persistence $k>1$ can boost learning, leading to more profitable policies, ii) assessing the quality of our persistence selection method, and iii) studying how the batch size influences the performance of PFQI policies for different persistences. Refer to Appendix D for detailed experimental settings.

We train PFQI, using extra-trees (Geurts et al., 2006) as a regression model, for J iterations and different values of k, starting with the same dataset $\mathcal{D}$ collected at persistence 1. To compare the performance of the learned policies π_k at the different persistences, we estimate their expected return J_k^{ρ,π_k} in the corresponding MDP $\mathcal{M}_k$. Table 1 shows the results for different continuous environments and different persistences averaged over 20 runs and highlighting in bold the persistence with the highest average performance and the ones that are not statistically significantly different from that one. Across the different environments we observe some common trends in line with our theory: i) persistence 1 rarely leads to the best performance; ii) excessively increasing persistence prevents the control at all. In Cartpole (Barto et al., 1983), we easily identify a persistence ($k=4$) that outperforms all the others. In the Lunar Lander (Brockman et al., 2016) persistences $k\in\{4,8\}$ are the only ones that lead to positive return (i.e., the lander does not crash) and in the Acrobot domain (Geramifard et al., 2015) we identify $k\in\{2,4\}$ as optimal persistences. A qualitatively different behavior is displayed in Mountain Car (Moore, 1991), Pendulum (Brockman et al., 2016), and Swimmer (Coulom, 2002), where we observe a plateau of three persistences with similar performance. An explanation for this phenomenon is that, in those domains, the optimal policy tends to persist actions on its own, making the difference less evident. Intriguingly, the more complex Mujoco domains, like Hopper and Walker 2D (Erickson et al., 2019), seem to benefit from the higher persistences.

To test the quality of our persistence selection method, we compare the performance of the estimated optimal persistence, i.e., the one with the highest estimated expected return $\widehat{k}\in\operatorname{argmax}\widehat{J}_k^{\rho,\pi_k}$, and the performance of the persistence $\widetilde{k}$ selected by maximizing the index B_k (Equation (15)). For each run $i=1,\dots,20$, we compute the *performance loss* $\delta_i=\widehat{J}_{\widehat{k}}^{\rho,\pi_{\widehat{k}}}-\widehat{J}_{\widetilde{k}_i}^{\rho,\pi_{\widetilde{k}_i}}$ and we report it in the last column of Table 1. In the Cartpole experiment, we observe a zero loss, which means that our heuristic always selects the optimal persistence ($k=4$). Differently, non–zero loss occurs in the other domains, which means that sometimes the index B_k mispredicts the optimal persistence. Nevertheless, in almost all cases the average performance loss is significantly smaller than the magnitude of the return, proving the effectiveness of our heuristics.

In Figure 2, we show the learning curves for the Cartpole experiment, highlighting the components that contribute to the index B_k. The first plot reports the estimated *expected return* $\widehat{J}_k^{\rho,\pi_k}$, obtained by averaging 10 trajectories executing π_k in the environment $\mathcal{M}_k$, which confirms that $k=4$ is the optimal persistence. The second plot shows the *estimated return* $\widehat{J}_k^{\rho}$ obtained by averaging the Q-function Q_k learned with PFQI(k), over the initial states sampled from ρ. We can see that for $k\in\{1,2\}$, PFQI(k) tends to overestimate the return, while for $k=4$ we notice a slight underestimation. The overestimation phenomenon can be explained by the fact that with small persistences we perform a large number

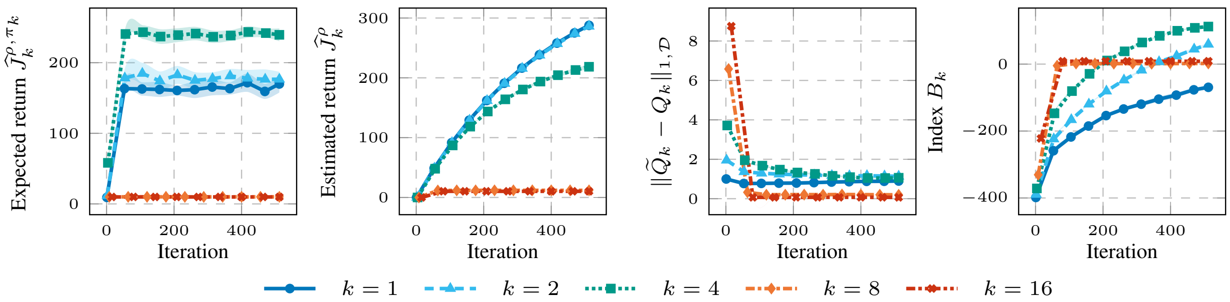

Figure 2. Expected return $\widehat{J}_k^{\rho,\pi_k}$, estimated return $\widehat{J}_k^{\rho}$, estimated expected Bellman residual $\|\widetilde{Q}_k - Q_k\|_{1,\mathcal{D}}$, and persistence selection index B_k in the Cartpole experiment as a function of the number of iterations for different persistences. 20 runs, 95 % c.i.

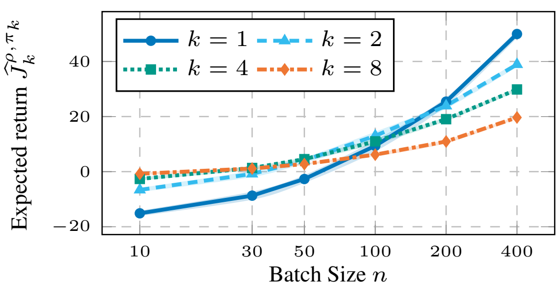

Figure 3. Expected return $\widehat{J}_k^{\rho,\pi_k}$ in the Trading experiment as a function of the batch size. 10 runs, 95 % c.i.

of applications of the operator $\widehat{T}^*$, which involves a maximization over the action space, injecting an overestimation bias. By combining this curve with the expected Bellman residual (third plot), we get the value of our persistence selection index B_k (fourth plot). Finally, we observe that B_k correctly ranks persistences 4 and 8, but overestimates persistences 8 and 16, compared to persistence 1.

To analyze the effect of the batch size, we run PFQI on the Trading environment (see Appendix D.4) varying the number of sampled trajectories. In Figure 3, we notice that the performance improves as the batch size increases, for all persistences. Moreover, we observe that if the batch size is small ($n \in \{10,30,50\}$), higher persistences ($k \in \{2,4,8\}$) result in better performances, while for larger batch sizes, $k = 1$ becomes the best choice. Since data is taken from real market prices, this environment is very noisy, thus, when the amount of samples is limited, PFQI can exploit higher persistences to mitigate the poor estimation.

9. Open Questions

Improving Exploration with Persistence We analyzed the effect of action persistence on FQI with a fixed dataset, collected in the base MDP $\mathcal{M}$. In principle, samples can be collected at arbitrary persistence. We may wonder how well the same sampling policy (e.g., the uniform policy over $\mathcal{A}$), executed at different persistences, explores the environment. For instance, in Mountain Car, high persistences increase the probability of reaching the goal, generating more informative datasets (preliminary results in Appendix E.1).

Learn in $\mathcal{M}_k$ and execute in $\mathcal{M}_{k'}$ Deploying each policy π_k in the corresponding MDP $\mathcal{M}_k$ allows for some guarantees (Lemma 6.1). However, we empirically discovered that using π_k in an MDP $\mathcal{M}_{k'}$ with smaller persistence k' sometimes improves its performance. (preliminary results in Appendix E.2). We wonder what regularity conditions on the environment are needed to explain this phenomenon.

Persistence in On–line RL Our approach focuses on batch off–line RL. However, the on–line framework could open up new opportunities for action persistence. Specifically, we could *dynamically* adapt the persistence (and so the control frequency) to speed up learning. Intuition suggests that we should start with a low frequency, reaching a fairly good policy with few samples, and then increase it to refine the learned policy.

10. Discussion and Conclusions

In this paper, we formalized the notion of action persistence, i.e., the repetition of a single action for a fixed number k of decision epochs, having the effect of altering the control frequency of the system. We have shown that persistence leads to the definition of new Bellman operators and that we are able to bound the induced performance loss, under some regularity conditions on the MDP. Based on these considerations, we presented and analyzed a novel batch RL algorithm, PFQI, able to approximate the value function at a given persistence. The experimental evaluation justifies the introduction of persistence, since reducing the control frequency can lead to an improvement when dealing with a limited number of samples. Furthermore, we introduced a persistence selection heuristic, which is able to identify good persistence in most cases. We believe that our work makes a step towards understanding why repeating actions may be useful for solving complex control tasks. Numerous questions remain unanswered, leading to several appealing future research directions.

Acknowledgements

The research was conducted under a cooperative agreement between ISI Foundation, Banca IMI and Intesa Sanpaolo Innovation Center.

References

Antos, A., Szepesvári, C., and Munos, R. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008. doi: 10.1007/s10994-007-5038-2.

Bacon, P., Harb, J., and Precup, D. The option-critic architecture. In Singh, S. P. and Markovitch, S. (eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 1726–1734. AAAI Press, 2017.

Baird, L. C. Reinforcement learning in continuous time: Advantage updating. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 4, pp. 2448–2453. IEEE, 1994.

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Systems, Man, and Cybernetics*, 13(5):834–846, 1983. doi: 10.1109/TSMC.1983.6313077.

Bertsekas, D. P. *Dynamic programming and optimal control, 3rd Edition*. Athena Scientific, 2005. ISBN 1886529264.

Bertsekas, D. P. and Shreve, S. *Stochastic optimal control: the discrete-time case*. 2004.

Bradtke, S. J. and Duff, M. O. Reinforcement learning methods for continuous-time markov decision problems. In Tesauro, G., Touretzky, D. S., and Leen, T. K. (eds.), *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, pp. 393–400. MIT Press, 1994.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Coulom, R. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control. (Apprentissage par renforcement utilisant des réseaux de neurones, avec des applications au contrôle moteur)*. PhD thesis, Grenoble Institute of Technology, France, 2002.

Dayan, P. and Singh, S. P. Improving policies without measuring merits. In Touretzky, D. S., Mozer, M., and Hasselmo, M. E. (eds.), *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, pp. 1059–1065. MIT Press, 1995.

Dietterich, T. G. The MAXQ method for hierarchical reinforcement learning. In Shavlik, J. W. (ed.), *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pp. 118–126. Morgan Kaufmann, 1998.

Doya, K. Reinforcement learning in continuous time and space. *Neural Computation*, 12(1):219–245, 2000. doi: 10.1162/089976600300015961.

Erickson, Z. M., Gangaram, V., Kapusta, A., Liu, C. K., and Kemp, C. C. Assistive gym: A physics simulation framework for assistive robotics. *CoRR*, abs/1910.04700, 2019.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.*, 6: 503–556, 2005.

Farahmand, A., Nikovski, D. N., Igarashi, Y., and Konaka, H. Truncated approximate dynamic programming with task-dependent terminal value. In Schuurmans, D. and Wellman, M. P. (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 3123–3129. AAAI Press, 2016.

Farahmand, A. M. *Regularization in Reinforcement Learning*. PhD thesis, University of Alberta, 2011.

Farahmand, A. M. and Szepesvári, C. Model selection in reinforcement learning. *Machine Learning*, 85(3):299–332, 2011. doi: 10.1007/s10994-011-5254-7.

Fleming, W. H. and Soner, H. M. *Controlled Markov processes and viscosity solutions*, volume 25. Springer Science & Business Media, 2006.

Geramifard, A., Dann, C., Klein, R. H., Dabney, W., and How, J. P. Rlpy: a value-function-based reinforcement learning framework for education and research. *J. Mach. Learn. Res.*, 16:1573–1578, 2015.

Geurts, P., Ernst, D., and Wehenkel, L. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006. doi: 10.1007/s10994-006-6226-1.

Györfi, L., Kohler, M., Krzyzak, A., and Walk, H. *A Distribution-Free Theory of Nonparametric Regression*. Springer series in statistics. Springer, 2002. ISBN 978-0-387-95441-7. doi: 10.1007/b97848.

Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., and Levine, S. Learning to walk via deep reinforcement learning. In Bicchi, A., Kress-Gazit, H., and Hutchinson, S. (eds.), *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, 2019. doi: 10.15607/RSS.2019.XV.011.

Howard, R. A. Semi-markov decision-processes. *Bulletin of the International Statistical Institute*, 40(2):625–652, 1963.

Jiang, N., Kulesza, A., Singh, S. P., and Lewis, R. L. The dependence of effective planning horizon on model accuracy. In Kambhampati, S. (ed.), *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pp. 4180–4189. IJCAI/AAAI Press, 2016.

Kilinc, O., Hu, Y., and Montana, G. Reinforcement learning for robotic manipulation using simulated locomotion demonstrations. *CoRR*, abs/1910.07294, 2019.

Kober, J. and Peters, J. *Learning Motor Skills - From Algorithms to Robot Experiments*, volume 97 of *Springer Tracts in Advanced Robotics*. Springer, 2014. ISBN 978-3-319-03193-4. doi: 10.1007/978-3-319-03194-1.

Lakshminarayanan, A. S., Sharma, S., and Ravindran, B. Dynamic action repetition for deep reinforcement learning. In Singh, S. P. and Markovitch, S. (eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 2133–2139. AAAI Press, 2017.

Lange, S., Gabel, T., and Riedmiller, M. A. Batch reinforcement learning. In Wiering, M. and van Otterlo, M. (eds.), *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pp. 45–73. Springer, 2012. doi: 10.1007/978-3-642-27645-3_2.

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. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Luenberger, D. G. Introduction to dynamic systems; theory, models, and applications. Technical report, New York: John Wiley & Sons, 1979.

Mann, T. A., Mannor, S., and Precup, D. Approximate value iteration with temporally extended actions. *J. Artif. Intell. Res.*, 53:375–438, 2015. doi: 10.1613/jair.4676.

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.

Metelli, A. M., Ghelfi, E., and Restelli, M. Reinforcement learning in configurable continuous environments. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4546–4555. PMLR, 2019.

Moore, A. W. Efficient memory based learning for robot control. *PhD Thesis, Computer Laboratory, University of Cambridge*, 1991.

Müller, A. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29 (2):429–443, 1997.

Munos, R. A convergent reinforcement learning algorithm in the continuous case based on a finite difference method. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, pp. 826–831. Morgan Kaufmann, 1997.

Munos, R. Performance bounds in l_p-norm for approximate value iteration. *SIAM journal on control and optimization*, 46(2):541–561, 2007.

Munos, R. and Bourgine, P. Reinforcement learning for continuous stochastic control problems. In Jordan, M. I., Kearns, M. J., and Solla, S. A. (eds.), *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997]*, pp. 1029–1035. The MIT Press, 1997.

Munos, R. and Szepesvári, C. Finite-time bounds for fitted value iteration. *J. Mach. Learn. Res.*, 9:815–857, 2008.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

Peters, J. and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4): 682–697, 2008. doi: 10.1016/j.neunet.2008.02.003.

Peterson, J. K. On-line estimation of the optimal value function: Hjb-estimators. In *Advances in Neural Information Processing Systems*, pp. 319–326, 1993.

Petrik, M. and Scherrer, B. Biasing approximate dynamic programming with a lower discount factor. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver,* *British Columbia, Canada, December 8-11, 2008*, pp. 1265–1272. Curran Associates, Inc., 2008.

Pirotta, M., Restelli, M., and Bascetta, L. Policy gradient in lipschitz markov decision processes. *Machine Learning*, 100(2-3):255–283, 2015. doi: 10.1007/s10994-015-5484-1.

Precup, D. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 2001.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Rachelson, E. and Lagoudakis, M. G. On the locality of action domination in sequential decision making. In *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2010, Fort Lauderdale, Florida, USA, January 6-8, 2010*, 2010.

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.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Singh, S. P. Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the National Conference on Artificial Intelligence*, number 10, pp. 202. JOHN WILEY & SONS LTD, 1992a.

Singh, S. P. Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Machine Learning Proceedings 1992*, pp. 406–415. Elsevier, 1992b.

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

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Solla, S. A., Leen, T. K., and Müller, K. (eds.), *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pp. 1057–1063. The MIT Press, 1999a.

Sutton, R. S., Precup, D., and Singh, S. P. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999b. doi: 10.1016/S0004-3702(99)00052-1.

Tallec, C., Blier, L., and Ollivier, Y. Making deep q-learning methods robust to time discretization. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6096–6104. PMLR, 2019.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Watkins, C. J. C. H. *Learning from delayed rewards*. PhD thesis, King's College, University of Cambridge, 1989.