Lazy-MDPs: Towards Interpretable RL by Learning When to Act

Alexis Jacq*
Google Research, Brain Team
Paris, France
alexisjacq@google.com

Johan Ferret*
Google Research, Brain Team
Inria, Scool Team
CRIStAL, CNRS, Université de Lille

Olivier Pietquin
Google Research, Brain Team
Paris, France
pietquin@google.com

Matthieu Geist
Google Research, Brain Team
Paris, France
mfgeist@google.com

ABSTRACT

Traditionally, Reinforcement Learning (RL) aims at deciding *how to act* optimally for an artificial agent. We argue that deciding *when to act* is equally important. As humans, we drift from default, instinctive or memorized behaviors to focused, thought-out behaviors when required by the situation. To enhance RL agents with this aptitude, we propose to augment the standard Markov Decision Process and make a new mode of action available: being *lazy*, which defers decision-making to a default policy. In addition, we penalize non-lazy actions in order to enforce minimal effort and have agents focus on critical decisions only. We name the resulting formalism *lazy-MDPs*. We study the theoretical properties of lazy-MDPs, expressing value functions and characterizing greediness and optimal solutions. Then we empirically demonstrate that policies learned in lazy-MDPs generally come with a form of interpretability: by construction, they show us the states where the agent takes control over the default policy. We deem those states and corresponding actions important since they explain the difference in performance between the default and the new, lazy policy. With suboptimal policies (even uniform random) as default, we observe that agents are still able to get close to and sometimes outperform DQN on Atari games while only taking control in a limited subset of states.

KEYWORDS

Reinforcement Learning; Markov Decision Processes; Explainability

ACM Reference Format:
Alexis Jacq*, Johan Ferret*, Olivier Pietquin, and Matthieu Geist. 2022. Lazy-MDPs: Towards Interpretable RL by Learning When to Act. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Decision-making is about providing answers to a standard question: *"how to act?"*. While Markov Decision Processes (MDPs) [34] provide the canonical formalism to ask this question, Reinforcement Learning (RL) provides algorithms to answer it. In this work, we study a different question: ***"when and how to act?"***. There are several motivations for this particular question. First, in many tasks there is only a handful of states that are critical and require complex decision-making, while in other states the action has less impact than the own dynamic of the MDP (for example, the orientation of a falling piece in Tetris has no importance until it reaches the floor). Another motivation is that of learning on top of an existing policy: one might be able to learn a better policy by learning when to take control over this default policy (this is the case in many human-robot interactions [26], for example self-driving cars that would let the human drive except in critical situations, robots assisting surgery, etc). Default policies can take arbitrary forms: controllers, handcrafted policies, programs, and many others.

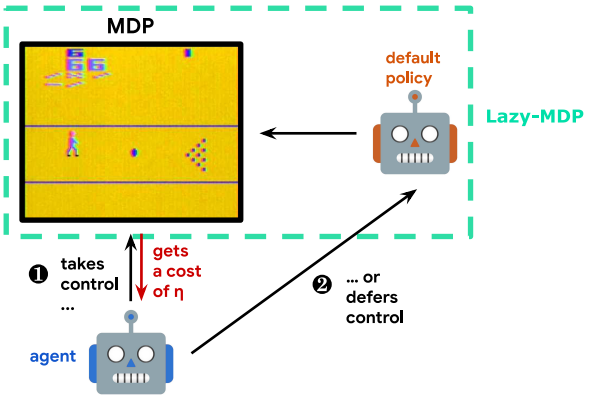


Figure 1: High-level overview of a lazy-MDP.

To study this alternative question, we need an alternative formalism. Instead of starting from scratch, we propose to augment the existing MDP framework (see Fig. 1): we extend the action space with a novel action, the *lazy* action; and we modify the reward function to penalize agents when they take control (*i.e.* pick an action from the original action space). Choosing the *lazy* action defers the decision-making process to a default policy. Augmenting the MDP framework means that we can take any decision-making problem that can be expressed as an MDP and turn it into a lazy-MDP. Since defaulting is a discrete action, we chose to focus on discrete actions setting for simplicity, but everything could be adapted to continuous control (for instance using two actors, a default one and a learned one; and one critic for each).

Lazy-MDPs have interesting properties for interpretability: states where the policy diverges from the default hold information. In more details, we leverage the statewise differences between the default policy and the new, lazy-policy to make sense of what is needed to get performance improvement with respect to the default

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

policy (under arbitrary default policies) or to make sense of the overall task (under specific default policies). An important point we want to highlight is that the type of interpretability we consider here is different from explanations [29]. Explanations, in the context of RL, would bring answers to "why" questions (either about agent behavior or the importance of actions), which is not what we tackle here.

Our contributions are the following: 1) we propose a novel formalism called *lazy-MDP* that provides modified decision-making problems where agents have to learn when and how to act, 2) we study lazy-MDPs from a theoretical point of view and prove that we can characterize optimality, which depends on the third-party policy and the value of the penalty for taking control, and 3) we study lazy-MDPs empirically, showing that they lead to learning an interpretable partition of the states. We also study how making control less frequent (by increasing the penalty) impacts the score of agents. In hard exploration tasks, reducing the frequency of controls can even lead to improved performance.

2 RELATED WORK

While the idea of constraining policies to adopt a default behavior as often as possible in MDPs is to the best of our knowledge novel, it lives at the crossroads of several subfields of RL reviewed here.

Residual RL. Residual approaches [21, 39] consist in learning a residual policy, whose action is added to that of a base policy to get the resulting action. By its nature, residual RL is restricted to continuous control problems, where the sum of two actions is still a valid action. In contrast, the lazy-MDP abstraction is applicable to discrete control problems.

Exploration-conscious RL. Here we discuss related augmented MDPs. Shani et al. [37] show that exploration-conscious RL [44] can be solved via a surrogate MDP, where the dynamics are obtained by a linear interpolation of the dynamics induced by the current policy and those induced by a fixed policy; same for rewards. This amounts to learning a policy that is optimal given that the actual behavior is a fixed mixture of that policy and a base one. In comparison, the policies learned in lazy-MDPs are more controllable in the sense that they can switch between the base policy and a learned policy on the basis of states, and serve the subtly different purpose of learning when and how to act.

Interpretable RL. There are several types of interpretability studied in the literature. Many works try to quantify the influence of parts of the inputs on the decisions of the agents. This can be done via post-hoc gradient methods, using actual policy gradients [48, 50] or finite-difference estimates [17, 33], coupled with saliency maps as visualizations. Another way to do so is by training ad-hoc interpretable models [12, 25] or programs [45, 46] to mimic the non-interpretable models used as policy networks. Others try to make sense of the representations learned by agents, using dimensionality reduction techniques [50] or state aggregation methods [42]. Another focus is to select trajectories that are representative of the overall behavior of the RL agent [1, 2]. Finally, some works try to recover the approximate preferences of the agent, under the form of a reward function, or coefficients for a known reward decomposition [9, 22]. While interesting on their own, none of the mentioned works explicitly tackle the question of identifying states that are crucial to the decision-making process. The action-gap [8] and importance advising [43] are quantities that hint at this aspect. As we show in Sec. 7.2, both suffer from several shortcomings compared to the proposed method.

Credit assignment in RL. Temporal credit assignment consists in associating specific actions to specific results (*i.e.* task success or high returns). Existing approaches complement or modify RL algorithms by either decomposing observed returns as the sum of redistributed rewards along observed trajectories [3, 13, 20, 35] or incorporating hindsight information into the RL process [14, 18, 27]. Our approach is related but differs in several points: it is tied to (and aims at making sense of) performance improvements instead of outcomes, and it comes from an abstraction over MDPs (which is still parametric but with dramatically less parameters).

Temporal abstractions in RL. Options are common temporal abstractions in RL [4, 5, 32, 41]. They consist in triples $(\mathcal{I}, \pi, \beta)$ where $\mathcal{I}$ is a set of states the option can be initiated into, π is the policy that selects actions when the option is active, and β is a random variable that gives the per-state probability of terminating the option. In general, learning options from scratch is hard, prone to collapse to single-action options, and less efficient than standard RL. Huang et al. [19] introduce Markov Jump Processes (MJPs), in which the agent both takes action and controls the frequency at which observations are received. Higher frequencies induce an increasing auxiliary cost to model scenarios where observations are limited. In essence, they propose to learn when to observe, while we propose to learn when to take control. In a related way, Biedenkapp et al. [10] propose skip-MDPs, which decompose policies in the combination of a behavior policy (*i.e.* which selects the action) and skip policy (*i.e.* which selects the number of timesteps the action will be repeated for). Skip-MDPs does not exactly learn when to act as it permanently plays a decided action. This work rather shows that in most of situations an action need to be repeated in consecutive states, but as there are no states where the agent is deferring the control to an independent policy, they do not highlight the states where the agent decisions has a strong impact on the resulting behaviour and reward. Note that dynamic action repetition [23, 38] is conceptually similar, but is not formalised as an abstraction over MDPs.

Regularized RL. Regularization in RL [15] is a well-studied topic. In particular, entropic regularization [30] encourages learned policies to be as random as possible in all states. In contrast, when the default policy is uniform random, policies learned in lazy-MDPs are encouraged to be entirely random in a subset of all states only. Also, Kullback-Leibler regularization [47] encourages policies to stay close to their previous iterate during learning. In contrast, policies learned in lazy-MDPs act identically to the default policy in a subset of all states, and can act in arbitrary ways in the others. Another way to ensure that the behavior of an agent does not diverge from a baseline behavior is to apply regularization on the state visitation distribution, instead of the action distribution induced by the policy [16, 24].

3 FRAMEWORK

We use the Markov Decision Process (MDP) formalism. An MDP is a tuple $M = (\mathcal{S}, \mathcal{A}, \gamma, r, \mathcal{P}, \delta_0)$ where $\mathcal{S}$ is a state space, $\mathcal{A}$ is a discrete action space, $\gamma \in [0, 1]$ is a discount factor, $r \in [r_{min}, r_{max}]^{\mathcal{S}\times\mathcal{A}}$

is a reward function, $\mathcal{P} \in \Delta_{\mathcal{S}}^{\mathcal{S}\times\mathcal{A}}$ is a transition kernel (here Δ_Y^X is the set of functions that map an element of X to a probability distribution over Y), and δ_0 is the distribution of the initial state. We note the subspace of absorbing states $\mathcal{S}_{abs} \subseteq \mathcal{S}$. Absorbing states deterministically transition to themselves with zero rewards. In the following, we assume that we are in the infinite-horizon setting, and that $\gamma < 1$, but the proposed formalism is applicable to the finite-horizon setting as well. Given an MDP, a policy $\pi \in \Delta_{\mathcal{S}}^{\mathcal{A}}$ maps states to probability distributions over actions is used to dict a behavior. The value function $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s]$ measures the expectation of the delayed rewards by following a policy π starting at s. Similarly, the action value function $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a]$ measures the expectation of the delayed rewards by following a policy π starting with action a at state s. Another value function that we will use in section 5 is $Z^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}\{s_t \notin \mathcal{S}_{\text{abs}}\}|s_0 = s, a_0 = a]$, which is the expected discounted sum of steps before the agent meets a terminating state.

We now define the **Lazy Markov Decision Process** (*lazy-MDP*). A lazy-MDP is a tuple $M_+ = (M, \bar{a}, \bar{\pi}, \eta)$, where $M = (\mathcal{S}, \mathcal{A}, \gamma, r, \mathcal{P}, \delta_0)$ is the *base* MDP, $\bar{a}$ the lazy action that defers decision making to the default policy $\bar{\pi} \in \Delta_{\mathcal{A}}^{\mathcal{S}}$ and $\eta \in \mathbb{R}$ is a penalty. The reward function is that of the base MDP, except that all actions but the lazy one incur an additional reward of $-\eta$. Hence, a lazy-MDP is also an MDP. It can be written as $M_+ = (\mathcal{S}, \mathcal{A}_+, \gamma, r_+, \mathcal{P}_+, \delta_0)$. While $\mathcal{S}, \gamma, \delta_0$ are conserved from the base MDP, $\mathcal{A}_+, r_+$ and $\mathcal{P}_+$ depend on their equivalents in the base MDP, and on $\bar{\pi}$ and η:

$$\mathcal{A}_+ = \mathcal{A} \cup \{\bar{a}\},$$

$$r_+(s, a) = \begin{cases} r(s, a) - \eta, & \text{if } a \in \mathcal{A}, \\ \sum_{a\in\mathcal{A}} \bar{\pi}(a|s) r(s, a), & \text{if } a = \bar{a}, \end{cases}$$

$$\mathcal{P}_+(s'|s, a) = \begin{cases} \mathcal{P}(s'|s, a), & \text{if } a \in \mathcal{A}, \\ \sum_{a\in\mathcal{A}} \bar{\pi}(a|s)\mathcal{P}(s'|s, a), & \text{if } a = \bar{a}. \end{cases}$$

In what follows, we will use the notation X_+ to distinguish functions or distributions over the augmented action space $\mathcal{A}_+$.

4 SOLVING LAZY-MDPS

In this section, we provide a characterization of the optimality in lazy-MDPs, similarly to what is done for regular MDPs. The derived results have two main implications. First (in Sec. 4.2 and 4.3), we identify what we call the *lazy-gap*, which quantifies the importance of taking control or not in a given state. Then (in Sec. 4.4), we show that taking control or not depending on the sole value of this lazy-gap leads to optimal behavior in the lazy-MDP. All statements are proven in the Appendix.

4.1 Value functions

Let $\pi_+(a \in \mathcal{A}_+|s)$ be a policy in the lazy-MDP. If the agent chooses the lazy action $\bar{a}$, the performed action $a \in \mathcal{A}$ is sampled according to the default policy $\bar{\pi}$. We formalize the resulting *lazy policy* (in the base MDP) as follows:

$$\pi(a \in \mathcal{A}|s) = P\Big[(a \sim \pi_+) \cup (\bar{a} \sim \pi_+ \cap a \sim \bar{\pi})\Big],$$
$$= \pi_+(a|s) + \pi_+(\bar{a}|s)\bar{\pi}(a|s),$$

satisfying $\sum_{a\in\mathcal{A}} \pi(a|s) = 1$. A crucial point is that π has the same dynamics in the base MDP as π_+ in the corresponding lazy-MDP. We are interested in the value function $V_+^{\pi_+}(s)$, which is the value of π_+ in the lazy-MDP, and takes the penalties into account. We would like to decompose $V_+^{\pi_+}(s)$ as a function of $V^\pi(s)$ (*i.e.* the value function associated with π in the base MDP) and a cost function $C^{\pi_+}(s)$:

$$V_+^{\pi_+}(s) = V^\pi(s) + C^{\pi_+}(s).$$

Theorem 1. *$C^{\pi_+}(s)$ satisfies the following Bellman equation:*

$$C^{\pi_+}(s) = -\eta(1 - \pi_+(\bar{a}|s)) + \gamma\mathbb{E}_{a\sim\pi, s'\sim\mathcal{P}(\cdot|s,a)} C^{\pi_+}(s').$$

While V^π is the expected discounted sum of rewards obtained by following π in the base MDP, the cost C^{π_+} can be interpreted as the expected discounted sum of the incurred penalties. For instance, a policy that never picks the lazy action (*i.e.* $\forall s,\ \pi_+(\bar{a}|s) = 0$) gets a maximal cost:

$$V_+^{\pi_+} = V^\pi - \frac{\eta}{1 - \gamma}.$$

4.2 Q-functions

Let $Q_{\setminus\bar{a}}^{\pi_+}(s, a \in \mathcal{A})$ be the value (in the lazy-MDP) of taking another action than the default one:

$$\begin{aligned} Q_{\setminus\bar{a}}^{\pi_+}(s, a) &= r(s, a) - \eta + \gamma\mathbb{E}_{s'}\Big[V_+^{\pi_+}(s')\Big] \\ &= r(s, a) - \eta + \gamma\mathbb{E}_{s'}\Big[V^\pi(s') + C^{\pi_+}(s')\Big] \\ &= Q^\pi(s, a) - \eta + \gamma\mathbb{E}_{s'}\Big[C^{\pi_+}(s')\Big], \end{aligned}$$

and $\pi_{\setminus\bar{a}}(a \in \mathcal{A}|s)$ be the policy obtained by excluding the lazy action from π_+, *i.e.* assuming $\pi_+(\bar{a}|s) < 1$:

$$\forall a \in \mathcal{A},\quad \pi_{\setminus\bar{a}}(a|s) = \frac{\pi_+(a|s)}{\sum\limits_{a'\neq\bar{a}} \pi_+(a'|s)} = \frac{\pi_+(a|s)}{1 - \pi_+(\bar{a}|s)}.$$

We can express the value function $V_+^{\pi_+}$ as a function of $Q_{\setminus\bar{a}}^{\pi_+}$:

Property 1.

$$\begin{aligned} V_+^{\pi_+}(s) =& (1 - \pi_+(\bar{a}|s))\mathbb{E}_{a\sim\pi_{\setminus\bar{a}}}\Big[Q_{\setminus\bar{a}}^{\pi_+}(s, a)\Big] \\ &+ \pi_+(\bar{a}|s)\Big(\mathbb{E}_{a\sim\bar{\pi}}\Big[Q_{\setminus\bar{a}}^{\pi_+}(s, a)\Big] + \eta\Big). \end{aligned}$$

We then have the expression for $Q_+^{\pi_+}(s, a \in \mathcal{A}_+)$, the Q-function of π_+ in the lazy-MDP:

Property 2.

$$Q_+^{\pi_+}(s, a) = \begin{cases} Q_{\setminus\bar{a}}^{\pi_+}(s, a) & \text{if } a \neq \bar{a}, \\ \mathbb{E}_{a\sim\bar{\pi}}\Big[Q_{\setminus\bar{a}}^{\pi_+}(s, a)\Big] + \eta & \text{if } a = \bar{a}. \end{cases}$$

4.3 Greediness

A policy π_+ is greedy wrt Q_+ (noted $\pi_+ \in \mathcal{G}(Q_+)$) if and only if:

$$\forall s \in \mathcal{S},\quad \pi_+(\cdot|s) \in \arg\max_{\pi_+(\cdot|s)} \mathbb{E}_{a\sim\pi_+}\Big[Q_+(s, a)\Big].$$

Given a Q-function Q, a useful quantity to construct a greedy policy is what we call the *lazy-gap*, noted $G_Q(s)$:

$$G_Q(s) = \max_{\mathcal{A}} Q(s,\cdot) - \mathbb{E}_{\bar{\pi}}\Big[Q(s,a)\Big],$$

which is the gap between the value of the best action (in $\mathcal{A}$) and the expected action-value when the immediate next action is picked by the default policy (as defined by Q) *in the lazy-MDP* (*i.e.*, with costs taken into account). In order to be greedy with respect to a policy Q_+, one needs to choose the lazy action if $G_{Q_{+\backslash\bar{a}}}(s) \leq \eta$, and to take the argmax of $Q_{+\backslash\bar{a}}$ otherwise (over $\mathcal{A}$).

Property 3. *The following policy π_+ is greedy with respect to Q_+:*

$$\pi_+(\cdot|s) = \begin{cases} \frac{\mathbb{I}\left\{a \in \arg\max_{\mathcal{A}} Q_{+\backslash\bar{a}}(s,a)\right\}}{|\arg\max_{\mathcal{A}} Q_{+\backslash\bar{a}}(s,a)|} & \text{if } G_{Q_{+\backslash\bar{a}}}(s) > \eta, \\ \mathbb{I}\left\{a = \bar{a}\right\} & \text{otherwise.} \end{cases}$$

Since the greediness as constructed above does not depend on the value of the lazy action $Q_+(s,\bar{a})$, we can define a greedy policy in the lazy-MDP with respect to a Q-function of the base MDP:

$$\mathcal{G}(Q)(\cdot|s) := \begin{cases} \frac{\mathbb{I}\left\{a \in \arg\max_{\mathcal{A}} Q(s,a)\right\}}{|\arg\max_{\mathcal{A}} Q(s,a)|} & \text{if } G_Q(s) > \eta, \\ \mathbb{I}\left\{a = \bar{a}\right\} & \text{otherwise.} \end{cases}$$

4.4 Optimality

We define the greedy operator $\mathcal{T}$, that maps a Q-function to the immediate reward plus the average value of the next state according to the greedy policy:

$$\mathcal{T}Q(s, a \in \mathcal{A}) := r(s,a) - \eta + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a), a' \sim \mathcal{G}(Q)(\cdot|s')}\Big[Q(s',a')\Big].$$

Theorem 2. *$\mathcal{T}$ is a γ-contraction, and converges to $Q^* := Q^{\pi_+^*}_{\backslash\bar{a}}$ where π_+^* is the optimal policy in the lazy-MDP.*

This allows us to identify the optimal policy to take decisions in the augmented action space $\mathcal{A}_+$.

Corollary 1. *π_+^* is a deterministic policy that verifies $\pi_+^*(\bar{a}|s) > 0$ if and only if $G^*(s) > \eta$, with $G^* = G_{Q^*}$ is the lazy-gap under the optimal action-value in the lazy-MDP.*

5 SETTING THE COST

Given a known base MDP, we may want to find the minimal cost $\eta_{\min}$ such that the lazy action is taken in at least one state, as well as the maximal cost $\eta_{\max}$ such that the lazy action is not taken in all states. From Prop. 1:

$$\eta_{\min} = \inf\Big\{\eta > 0 \text{ s.t. } \exists s,\ G^*(s) < \eta\Big\},$$
$$\eta_{\max} = \sup\Big\{\eta > 0 \text{ s.t. } \exists s,\ G^*(s) > \eta\Big\},$$

where G^* is the lazy-gap associated with the optimal value function $Q^{\pi_+^*}_{\backslash\bar{a}}$. Taking η between these two values allows to train agents that decide when to act in a non-trivial way. One can then equate states where the agent takes control as important states, under the right η.

5.1 $\eta_{\max}$

When η is equal or larger than the lazy-gap in all states, the optimal policy consists in deferring all actions to the default policy, which induces no cost. Thus, $\eta_{\max}$ is simply equal to the maximal lazy-gap under the default policy.

Theorem 3. *Let $Q^{\bar{\pi}}(s, a \in \mathcal{A})$ be the Q-function of the default policy in the base MDP. Then: $\eta_{max} = \max_s G_{Q^{\bar{\pi}}}(s)$.*

5.2 $\eta_{\min}$

One cannot apply a similar treatment to $\eta_{\min}$, due to most actions being non-default and corresponding costs having to be taken into account. However, if η is smaller or equal to the lazy-gap in all states, an optimal agent will follow the optimal policy of the base MDP π^*, and the incurred cost will be equal to $-\eta$ multiplied by the fictitious Q-value associated with π^* for a reward function that is 0 in absorbing states and 1 otherwise:

$$Z^{\pi^*}(s,a) = \mathbb{I}\{s \notin \mathcal{S}_{abs}\} + \gamma \mathbb{E}_{s'}\Big[\mathbb{E}_{\pi^*}\Big[Z^{\pi^*}(s',\cdot)\Big]\Big]$$

By construction $C^{\pi^*}(s) = -\eta \mathbb{E}_{\pi^*}[Z^{\pi^*}(s,\cdot)]$, where $C^{\pi^*}(s)$ is the cost for always following π^* from s. If no absorbing state is ever reached, we have $Z^{\pi^*}(s,a) = \frac{1}{1-\gamma}$ for all state s and action a. However, in practice the MDPs we consider have terminal states and eventually end. Z^{π^*} can thus have different values under different state-action couples, impacting the value of $\eta_{\min}$. Its value is given by the following theorem:

Theorem 4. *Let π^* be the optimal policy in the base MDP, and $Q^*(s, a \in \mathcal{A})$ the associated Q-function. Then:*

$$\eta_{min} = \min_s \max_a \frac{Q^*(s,a) - \mathbb{E}_{\bar{\pi}}\Big[Q^*(s,\cdot)\Big]}{1 + \left(\mathbb{E}_{\pi^*}\Big[Z^{\pi^*}(s,\cdot)\Big] - \mathbb{E}_{\bar{\pi}}\Big[Z^{\pi^*}(s,\cdot)\Big]\right)},$$

with $\eta_{min} \geq 0$.

We empirically validate these boundaries over different lazy-MDPs and report the results in Appendix **??**. As expected, when $\eta < \eta_{\min}$ no lazy actions are ever selected, and when $\eta > \eta_{\max}$, the agent always chooses the lazy action. We discuss how to approximate $\eta_{\min}$ and $\eta_{\max}$ in the next section.

6 LEARNING WHEN AND HOW TO ACT

Since lazy-MDPs can be described as augmented MDPs, standard RL algorithms can still be used to provide policies that maximize the cumulative sum of rewards (which include a cost when taking control). As a result, by converting an MDP into a lazy-MDP, RL agents learn when and how to act without any change to their workings. We now discuss design choices for the two parameters of lazy-MDPs: the cost η and the third-party policy $\bar{\pi}$.

Value of cost. Regarding η, the explicit expressions for the bounds we provided guarantee meaningful behavior from optimal

policies (*i.e.* not always defaulting and not always taking control). Estimating $\eta_{\max}$ is feasible since the default policy $\bar{\pi}$ is supposed available. It requires to estimate its action-values (for instance, using SARSA [36]) so that the maximal lazy-gap can be approximated (either by taking the maximum gap across the known set of states, or using rollouts to get approximate coverage). To estimate $\eta_{\min}$, usually one does not have access to the optimal Q-function Q^* of the base MDP nor to Z^{π^*}. In that case, a solution is to use value iteration or Q-learning [49] to get approximations Q_θ and Z_θ, where Z_θ is obtained by replacing all the rewards by ones in the loss used to learn Q_θ.

Choice of the default policy. Regarding the choice of $\bar{\pi}$, we argue that taking a random policy (with, say, uniform action probabilities) is the simplest option available: it does not require any knowledge about the task, and is on par with the idea that the agent should take control only when actions actually matter (*i.e.*, a specific action is noticeably better than uniform sampling). Doing so results in a type of regularization that is conceptually close to entropic regularization, except that the agent has incentive to be as random as possible *in a subset of the states only*. In some cases, including complex scenarios, alternative options might be preferable: having to take control too often could lead to a high cumulative cost and discourage exploration, unless η is properly tuned. Similarly to residual learning, an interesting substitute is a known, suboptimal policy. In that case, the agent has incentive to take control in states where the base policy is noticeably suboptimal.

Interpretability of lazy policies. Due to the cost of taking control, the lazy policies that are learned should only take control in a handful of states. We hope that this leads to increased interpretability for several reasons: the subset of states (and the corresponding controls) can be assessed against expert knowledge, and the performance of the learned policy can be compared to that of the base policy to ensure that the gains are sufficient and justify the overhead. Specifically, we argue that there are two special cases where lazy actions indeed give information about the *overall* importance (and unimportance) of states:

(1) with a uniform random policy as default (and the right penalty), an optimal agent should only pick its actions when selecting among optimal actions brings a substantial advantage over picking the action at random. Therefore, we argue that states where the agent defers its actions are likely to be unimportant in the sense that the agent is content with acting randomly.
(2) with a mixture between optimal and uniform random policy as default (and the right penalty), an optimal agent should only pick its actions when selecting among optimal actions brings a substantial advantage over *often* selecting among optimal actions. Therefore, we argue that states where the agent picks its actions are likely to be important in the sense that the agent is *not* content with acting *almost* optimally.

More broadly speaking, lazy actions give information about *the specific importance of states so as to improve on the performance of the default policy.*

We study the interpretability of solutions empirically in the next section.

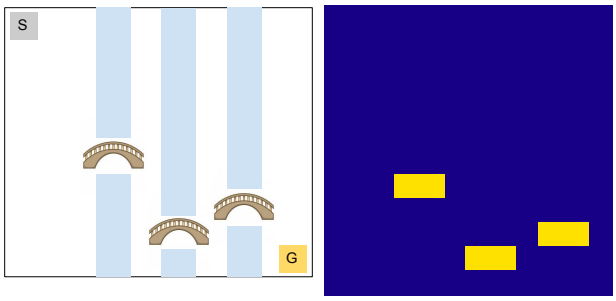


Figure 2: Left: Rivers and Bridges environment. The agent starts up left (S) and has to cross the rivers through the bridges to reach the goal point (G). Falling in the water is punished by $\mathcal{R} = -100$ and reaching the goal is rewarded by $\mathcal{R} = 1$. The default policy is the optimal π^* everywhere but on the bridges where it is uniformly random. Right: Heatmap of the resulting lazy-gap using $\eta = \eta_{\min} = 0$. As expected the lazy-gap is zero everywhere but on the bridges where optimal agents learn to take control. This result is valid for any value of $\eta < \eta_{\max}$.

7 EXPERIMENTS

In this section, we empirically address the following questions: 1) Do policies from lazy-MDPs learn to take control when it matters? 2) Are the partitions of states where the agent decides or not to act interpretable? 3) How does reducing the frequency of agent controls (by increasing the cost η) affect its returns? Details about implementations and the choice of hyperparameters can be found in Appendix **??**.

7.1 Taking control when it matters

We first study the behavior of lazy policies on small discrete problems, where the exact solutions can be approached as well as values for $\eta_{\min}$ and $\eta_{\max}$.

Rivers and Bridges. A simple environment that illustrates how lazy-MDPs work is a gridworld involving some dangerous pathways – where the agent needs to provide precise controls, while other states are safe – the agent can rely on the default policy. We implemented a basic scenario in which the agent has to cross three rivers by taking slippery bridges. We call this environment Rivers and Bridges (R&B), which is illustrated in Fig 2. Falling in the water is penalized by a strong negative reward ($\mathcal{R} = -100$), while reaching the goal point beyond the rivers results in a small positive reward ($\mathcal{R} = 1$). To simulate the slipperiness of the bridges, we take as default policy the policy that is optimal everywhere but on the bridges where it is uniformly random. That way, the agent should trust the default policy everywhere but on the bridges where it should take the control despite the cost. As there is at least one state where the policy is optimal, applying Theorem 4, we get $\eta_{\min} = 0$. As shown in Fig. 2 right, we verify that for any η such that $0 < \eta \leq \eta_{\max}$, the lazy-gap $G^*(s)$ is only positive on the bridges, which means an optimal agent only takes control in those states, and justifies the application of lazy-MDPs to evaluate where and when to trust a default behavior.

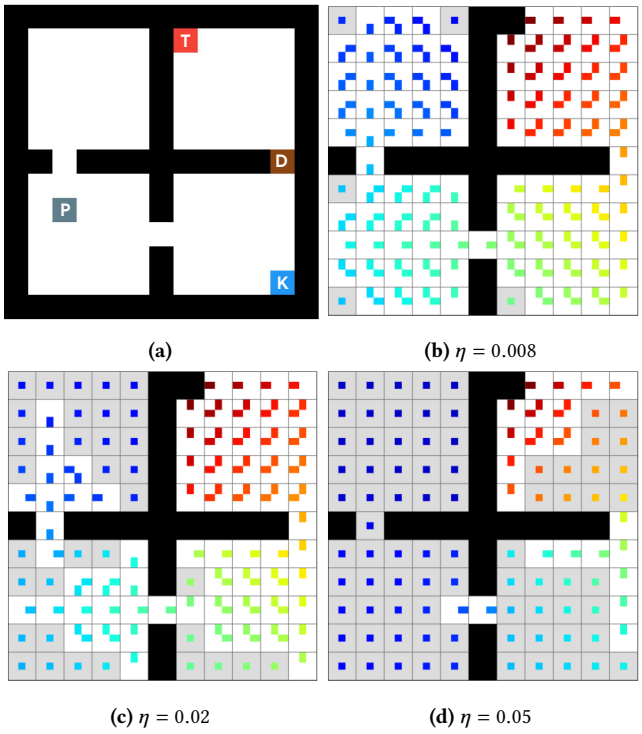


Figure 3: (a): Key-Door-Treasure environment. (b-d): Policies learned in the lazy-MDP by value iteration, with a uniform random default policy, and increasing η. Centered points in gray cells identify lazy actions, color indicates learned action-values (blue=lowest, red=highest). With η increasing, the policy learns to reduce its controls to states that allow it to progress from one room to another as well as those in the vicinity of the goal state.

Key-Door-Treasure [31] (KDT) is a variant of the classic Four Rooms task [40] with a harder exploration problem: the agent needs to grab a key, open the door and get to the location of the treasure (in that order) to solve the task. The agent is only rewarded when reaching the treasure. We study the nature of the solutions learned in the lazy-MDP version of KDT under several values of η, with a uniform random default policy. The results are shown in Fig. 3. They match our intuition: the higher the value of η, the fewer the states in which the agent takes control; until the agent only acts in the most crucial states (*i.e.* to pass from one room to another or to get to the treasure).

7.2 Interpretability

To study interpretability, we use more complex environments where the behaviour of an RL agents is not trivially interpretable. We make the hypothesis that lazy-MDPs can help at explaining which states and what actions are important in order to get high returns. In this study, we op for a qualitative measure of importance and show (via corresponding frames) the states of importance as the ones where the agent decides to take control over the uniform random, default policy. We look at lazy policies learned in the lazy-MDP version of Atari 2600 games from the Arcade Learning Environment [7]. We focus on games where timing plays a central role, such as Pong, Breakout and Ms Pacman. We use a standard DQN agent [28], whose implementation we take from the Dopamine framework [11]. We display a representative portion of a lazy agent trajectory in Breakout in Fig. 4, which is well aligned with our intuition of the timing of this task: critical controls happen when the ball gets back to the paddle, while the remaining controls have limited impact on subsequent success. We also display key moments of a lazy agent trajectory in Ms Pacman in Fig. 5. The agent alternates between defaulting (most of the time) and taking control (sparsely, either for a single frame or a sequence of frames) in order to escape ghosts, to obtain power-ups and defeat ghosts, or to collect multiple bonuses in a row. Finally, we display a representative portion of episode in Bowling in Fig. 6. The agent takes control when aiming with the ball, and defers control to the default policy when the ball is moving towards the pins, during which actions have no effects on the outcome of the throw. All in all, the timing of the agent controls matches our intuitions.

We also compared the importance as quantified by the lazy-gap with usual measures, such as the action-gap [8], which measures the difference between the best and the second best action values at a given state ($\max_{\mathcal{A}} Q(s,\cdot) - \max_{\mathcal{A}\setminus a^*} Q(s,\cdot)$), and importance advice [43], which measures the difference between the best and the worst action values at a given state ($\max_{\mathcal{A}} Q(s,\cdot) - \min_{\mathcal{A}} Q(s,\cdot)$). Fig. ?? in Appendix ?? displays the state importance according to these measures on the KDT environment. As visible, the lazy-gap only attributes importance to states with key actions (picking up the key, passing through doors, reaching the treasure). On the other hand, the action-gap uniformly emphasizes all states along the trajectory of the optimal policy, while the importance advice is dominated by the proximity to the reward and does not discriminate key actions.

7.3 Doing less for better exploration

A side effect of lazy-MDPs with a uniform random default policy is that they push agents to maintain randomness in states where acting randomly is affordable (*i.e.* does not impact future performance too much). This is helpful in hard exploration tasks, where local minima make the exploration more difficult. Actually, encouraging randomness in the policy via lazy-MDPs has two benefits: it regularizes behaviors and avoids determinism, and it rewards smart exploration where the random actions are only taken when it is safe to explore. To study the role of lazy-MDPs for exploration, we add a distractor state (*i.e.* an absorbing state with a small reward) in the upper-left room in KDT so as to introduce a local minimum. Q-learning agents, even when explicitly increasing exploration (e.g. with linearly decayed epsilon-greedy action selection), mostly fail and always go for the distractor. In that situation, augmenting the MDP as a lazy-MDP with a uniform random default policy encourages random behaviors over consecutive steps, which helps exploring and going past the local minimum. We illustrate this effect on Fig. 7. For that experiment, we used tabular Q-learning with learning rate $\alpha = 0.5$, epsilon-greedy exploration starting at $\epsilon_0 = 0.1$ and linearly decayed until $\epsilon_\infty = 0$, $\gamma = 0.99$, and a reward $r = 0.1$ for the apple. Episodes that did not end in an absorbing

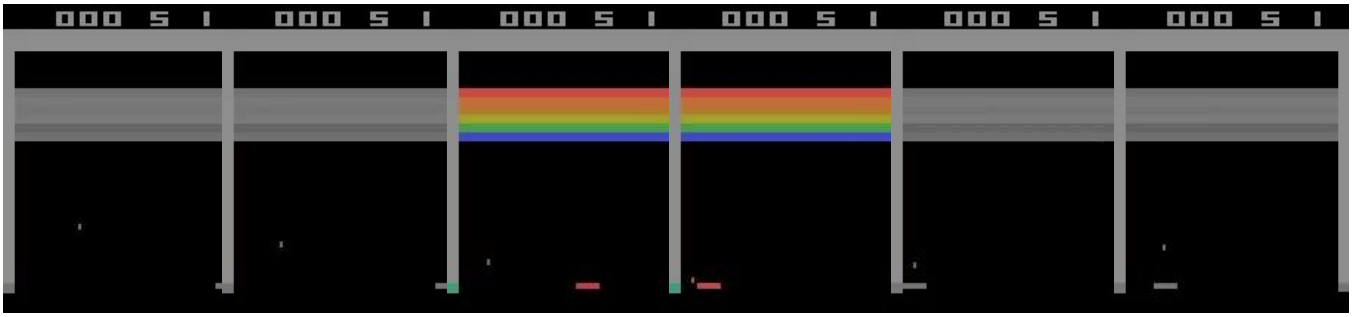


Figure 4: Illustration of the policy learned in the lazy-MDP version of Breakout ($\eta = 0.1$, with a uniform random default policy): the agent learns to take control (colored frames) only moments before the ball has to be hit in order not to lose.

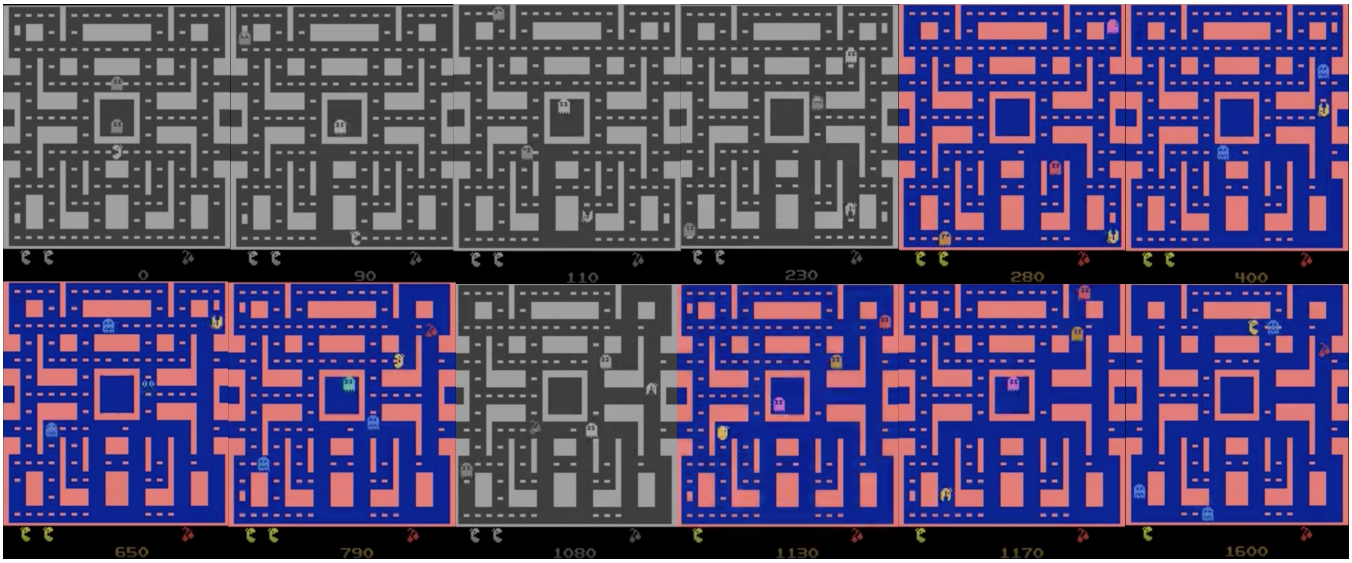


Figure 5: Illustration of the policy learned in the lazy-MDP version of Ms Pacman ($\eta = 0.5$, with a uniform random default policy). Frames are ordered from left to right, top to bottom, and correspond to non-consecutive frames from a single episode of interaction. At the beginning of the episode, the agent is immobile for a few frames, during which it defaults its actions to avoid penalties (frame 1). Once free to move, it keeps defaulting its actions and collecting nearby bonuses (frames 2-4) until ghosts become close enough. The agent then sparsely takes control (colored frames), be it to obtain power-ups (frames 5 & 7), and later on eat the ghosts (frames 6 & 12), or collect several bonuses in a row (frames 10-11).

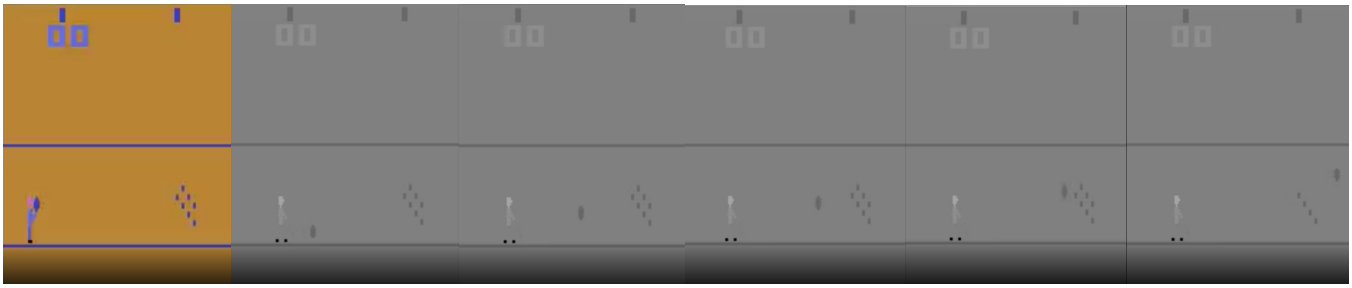
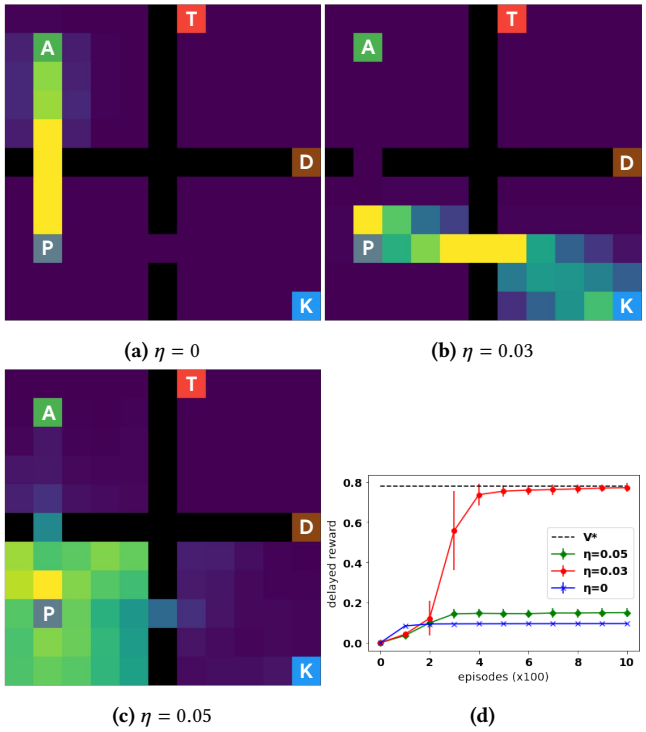


Figure 6: Illustration of the policy learned in the lazy-MDP version of Bowling ($\eta = 0.04$, with a uniform random default policy). The agent learns to only take control (colored frames) when preparing to throw the ball. This is the only time it can control the orientation of the throw, which is key to knock pins down. In subsequent timesteps, its actions have no effects on the ball, and accordingly the lazy action is picked instead.



(a) $\eta = 0$ (b) $\eta = 0.03$ (c) $\eta = 0.05$ (d)

Figure 7: We add an apple which grants a small reward to KDT. It acts as a distractor for exploration. (a-c): Asymptotic state occupancy measure of q-learning for different values of the cost η, *before the agent has the key*. We can see that q-learning in the original MDP ($\eta = 0$) is attracted to the local optimum, while q-learning in the lazy-MDP avoids the local optimum and eventually learns the optimal behavior (when $\eta = 0.03$). (d): Scores (excluding the cost) of q-learning with different values of η. Only the agent solving the lazy-MDP with $\eta = 0.03$ converges to the optimal behavior. Both state occupancy measures and performance curves are averaged over 100 seeds.

state (*i.e.* treasure or apple) were ended after 1000 steps. Under the right cost ($\eta = 0.03$) lazy policies keep exploring after finding the small reward, and eventually find the key and the treasure, leading to a reward that justifies the cost for taking control. With a cost too high ($\eta = 0.05$), lazy policies never take control.

This motivates investigating if such an effect of taking less control while achieving higher returns can be observed in more complex games, requiring function approximation. Hence we converted hard exploration tasks in Atari to lazy-MDPs, including dense reward tasks (Bank Heist, Frostbite, Ms Pacman, Zaxxon) and sparse reward tasks (Gravitar, Private Eye), as classified in [6]. As previously, we used uniform random default policies and several values for the cost η (0.005, 0.01, 0.02, 0.05, 0.1, 0.2) and reported the percentage of the score (with respect to a standard DQN agent [28]) as a function of the resulting frequency of control taking (when the lazy policy does not choose the lazy action) in figure 8. Reported values are averaged over 3 seeds. We observe that in most cases,

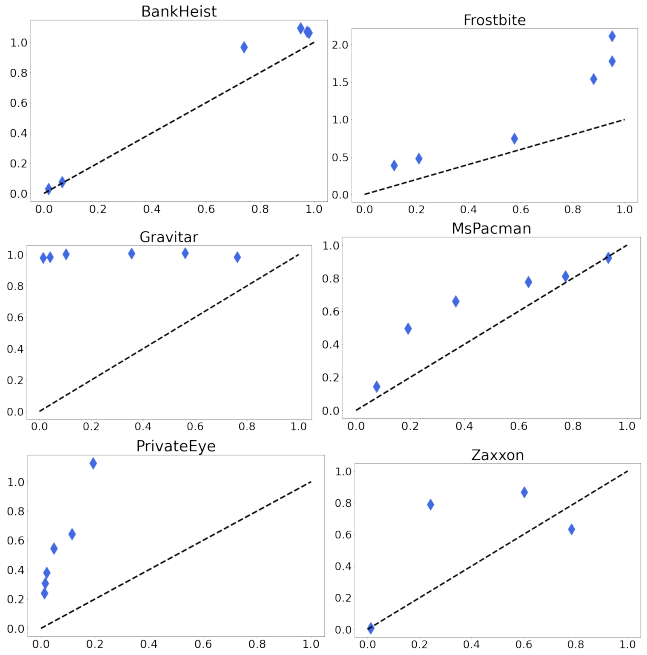


Figure 8: Percentage of the score of a standard DQN agent achieved by lazy-policies learned with different values of η and uniformly random default policy on hard exploration tasks of Atari. The x-axis represents the fraction of controls taken and the y-axis the percentage of the DQN baseline score reached. Each value is averaged over 3 seeds. Score 0% correspond to getting no reward and score 100% correspond to getting as much reward as a standard DQN.

reducing the frequency of control taking does not decrease the score too much (up to almost 100% of the score in Gravitar with less than 10% of control, 80% of the score with less than 30% of control in Zaxxon and more than 100% of the score with 20% of control on Private Eye). Moreover, we even observed in Frostbite that the lazy policy learned with low penalties achieved 200% of the score, confirming that lazy-MDP can also have applications for improved exploration.

8 CONCLUSION

In this work, we studied a novel paradigm for decision-making: learning when and how to act. We proposed lazy-MDPs, natural abstractions over MDPs that are well-suited for this learning problem, and showed that RL could still be used to provide solutions in that setting. We studied the theoretical properties of lazy-MDPs, including value functions and optimality. In experiments, we showed that lazy-MDPs present an interesting edge: when converted back to the original MDPs, the policies learned in lazy-MDPs tend to be more interpretable, as they highlight states where taking control is crucial to achieve increased returns. With uniform random default policies, we show that policies learned in lazy-MDPs via DQN perform close to policies learned in standard MDPs, while only taking control in a fraction of the states; and can even reach higher scores in hard exploration tasks.

REFERENCES

[1] D. Amir and O. Amir. Highlights: Summarizing agent behavior to people. In *International Conference on Autonomous Agents and Multiagent Systems*, 2018.

[2] Y. Amitai and O. Amir. "I don't think so": Disagreement-based policy summaries for comparing agents. *arXiv preprint arXiv:2102.03064*, 2021.

[3] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. Rudder: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, 2019.

[4] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *AAAI Conference on Artificial Intelligence*, 2017.

[5] A. Barreto, D. Borsa, S. Hou, G. Comanici, E. Aygün, P. Hamel, D. K. Toyama, J. J. Hunt, S. Mourad, D. Silver, et al. The option keyboard: Combining skills in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

[6] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, 2016.

[7] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2013.

[8] M. G. Bellemare, G. Ostrovski, A. Guez, P. Thomas, and R. Munos. Increasing the action gap: New operators for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[9] I. Bica, D. Jarrett, A. Hüyük, and M. van der Schaar. Learning "what-if" explanations for sequential decision-making. In *International Conference on Learning Representations*, 2021.

[10] A. Biedenkapp, R. Rajan, F. Hutter, and M. Lindauer. Towards TempoRL: learning when to act. *International Conference on Machine Learning, BIG workshop*, 2021.

[11] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, and M. G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. *arXiv preprint arXiv:1812.06110*, 2018. URL http://arxiv.org/abs/1812.06110.

[12] Y. Coppens, K. Efthymiadis, T. Lenaerts, A. Nowé, T. Miller, R. Weber, and D. Magazzeni. Distilling deep reinforcement learning policies in soft decision trees. In *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*, 2019.

[13] J. Ferret, R. Marinier, M. Geist, and O. Pietquin. Self-attentional credit assignment for transfer in reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2019.

[14] J. Ferret, O. Pietquin, and M. Geist. Self-imitation advantage learning. In *International Conference on Autonomous Agents and Multiagent Systems*, 2021.

[15] M. Geist, B. Scherrer, and O. Pietquin. A theory of regularized markov decision processes. In *International Conference on Machine Learning*, 2019.

[16] M. Geist, J. Pérolat, M. Laurière, R. Elie, S. Perrin, O. Bachem, R. Munos, and O. Pietquin. Concave utility reinforcement learning: the mean-field game viewpoint. *arXiv preprint arXiv:2106.03787*, 2021.

[17] S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, 2018.

[18] A. Harutyunyan, W. Dabney, T. Mesnard, M. Gheshlaghi Azar, B. Piot, N. Heess, H. P. van Hasselt, G. Wayne, S. Singh, D. Precup, et al. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*, 2019.

[19] Y. Huang, V. Kavitha, and Q. Zhu. Continuous-time markov decision processes with controlled observations. In *Allerton Conference on Communication, Control, and Computing*, 2019.

[20] C.-C. Hung, T. Lillicrap, J. Abramson, Y. Wu, M. Mirza, F. Carnevale, A. Ahuja, and G. Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 2019.

[21] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *International Conference on Robotics and Automation*, 2019.

[22] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*, 2019.

[23] A. Lakshminarayanan, S. Sharma, and B. Ravindran. Dynamic action repetition for deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2017.

[24] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[25] G. Liu, O. Schulte, W. Zhu, and Q. Li. Toward interpretable deep reinforcement learning with linear model u-trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2018.

[26] V. B. Meresht, A. De, A. Singla, and M. Gomez-Rodriguez. Learning to switch between machines and humans. *arXiv preprint arXiv:2002.04258*, 2020.

[27] T. Mesnard, T. Weber, F. Viola, S. Thakoor, A. Saade, A. Harutyunyan, W. Dabney, T. Stepleton, N. Heess, A. Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. In *International Conference on Machine Learning*, 2021.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[29] C. Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[30] G. Neu, A. Jonsson, and V. Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

[31] J. Oh, Y. Guo, S. Singh, and H. Lee. Self-imitation learning. In *International Conference on Machine Learning*, 2018.

[32] D. Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

[33] N. Puri, S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *International Conference on Learning Representations*, 2019.

[34] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[35] D. Raposo, S. Ritter, A. Santoro, G. Wayne, T. Weber, M. Botvinick, H. van Hasselt, and F. Song. Synthetic returns for long-term credit assignment. *arXiv preprint arXiv:2102.12425*, 2021.

[36] G. A. Rummery and M. Niranjan. *Online Q-learning using connectionist systems*, volume 37. Citeseer, 1994.

[37] L. Shani, Y. Efroni, and S. Mannor. Exploration conscious reinforcement learning revisited. In *International Conference on Machine Learning*, 2019.

[38] S. Sharma, A. S. Lakshminarayanan, and B. Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. In *International Conference on Learning Representations*, 2017.

[39] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[40] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[41] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

[42] N. Topin and M. Veloso. Generation of policy-level explanations for reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2019.

[43] L. Torrey and M. Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013.

[44] H. Van Seijen, H. Van Hasselt, S. Whiteson, and M. Wiering. A theoretical and empirical analysis of expected sarsa. In *IEEE symposium on adaptive dynamic programming and reinforcement learning*, pages 177–184, 2009.

[45] A. Verma, H. M. Le, Y. Yue, and S. Chaudhuri. Imitation-projected programmatic reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

[46] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, 2019.

[47] N. Vieillard, T. Kozuno, B. Scherrer, O. Pietquin, R. Munos, and M. Geist. Leverage the average: an analysis of kl regularization in rl. In *Advances in Neural Information Processing Systems*, 2020.

[48] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, 2015.

[49] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 1992.

[50] T. Zahavy, N. Ben-Zrihem, and S. Mannor. Graying the black box: Understanding dqns. In *International Conference on Machine Learning*, 2016.