HAL
open science

There Is No Turning Back: A Self-Supervised Approach for Reversibility-Aware Reinforcement Learning

Nathan Grinsztajn, Johan Ferret, Olivier Pietquin, Philippe Preux, Matthieu Geist

To cite this version:

Nathan Grinsztajn, Johan Ferret, Olivier Pietquin, Philippe Preux, Matthieu Geist. There Is No Turning Back: A Self-Supervised Approach for Reversibility-Aware Reinforcement Learning. Neural Information Processing Systems (2021), Dec 2021, Virtual, France. hal-03454640

HAL Id: hal-03454640

https://hal.science/hal-03454640v1

Submitted on 29 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

There Is No Turning Back: A Self-Supervised Approach for Reversibility-Aware Reinforcement Learning

Nathan Grinsztajn*
Inria, Scool Team
CRIStAL, CNRS, Université de Lille
nathan.grinsztajn@inria.fr

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

Olivier Pietquin
Google Research, Brain Team

Philippe Preux
Inria, Scool Team
CRIStAL, CNRS, Université de Lille

Matthieu Geist
Google Research, Brain Team

Abstract

We propose to learn to distinguish reversible from irreversible actions for better informed decision-making in Reinforcement Learning (RL). From theoretical considerations, we show that approximate reversibility can be learned through a simple surrogate task: ranking randomly sampled trajectory events in chronological order. Intuitively, pairs of events that are always observed in the same order are likely to be separated by an irreversible sequence of actions. Conveniently, learning the temporal order of events can be done in a fully self-supervised way, which we use to estimate the reversibility of actions from experience, without any priors. We propose two different strategies that incorporate reversibility in RL agents, one strategy for exploration (RAE) and one strategy for control (RAC). We demonstrate the potential of reversibility-aware agents in several environments, including the challenging Sokoban game. In synthetic tasks, we show that we can learn control policies that never fail and reduce to zero the side-effects of interactions, even without access to the reward function.

1 Introduction

We address the problem of estimating if and how easily actions can be reversed in the Reinforcement Learning (RL) context. Irreversible outcomes are often not to be taken lightly when making decisions. As humans, we spend more time evaluating the outcomes of our actions when we know they are irreversible [29]. As such, irreversibility can be positive (*i.e.* takes risk away for good) or negative (*i.e.* leads to later regret). Also, decision-makers are more likely to anticipate regret for hard-to-reverse decisions [50]. All in all, irreversibility seems to be a good prior to exploit for more principled decision-making. In this work, we explore the option of using irreversibility to guide decision-making and confirm the following assertion: by estimating and factoring reversibility in the action selection process, safer behaviors emerge in environments with intrinsic risk factors. In addition to this, we

*Equal contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

show that exploiting reversibility leads to more efficient exploration in environments with undesirable irreversible behaviors, including the famously difficult Sokoban puzzle game.

However, estimating the reversibility of actions is no easy feat. It seemingly requires a combination of planning and causal reasoning in large dimensional spaces. We instead opt for another, simpler approach (see Fig. 1): we propose to learn in which direction time flows between two observations, directly from the agents' experience, and then consider *irreversible* the transitions that are assigned a temporal direction with high confidence. *In fine*, we reduce reversibility to a simple classification task that consists in predicting the temporal order of events.

Our contributions are the following: 1) we formalize the link between reversibility and precedence estimation, and show that reversibility can be approximated via temporal order, 2) we propose a practical algorithm to learn temporal order in a self-supervised way, through simple binary classification using sampled pairs of observations from trajectories, 3) we propose two novel exploration and control strategies that incorporate reversibility, and study their practical use for directed exploration and safe RL, illustrating their relative merits in synthetic as well as more involved tasks such as Sokoban puzzles.

2 Related Work

To the best of our knowledge, this work is the first to explicitly model the reversibility of transitions and actions in the context of RL, using temporal ordering to learn from trajectories in a self-supervised way, in order to guide exploration and control. Yet, several aspects of the problem we tackle were studied in different contexts, with other motivations; we review these here.

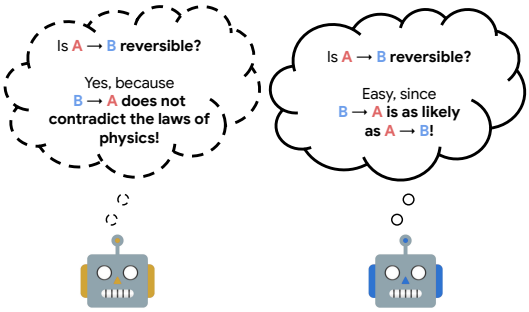


Figure 1: High-level illustration of how reversibility can be estimated. **Left:** from an understanding of physics. **Right:** ours, from experience.

Leveraging reversibility in RL. Kruusmaa et al. [26] estimate the reversibility of state-action couples so that robots avoid performing irreversible actions, since they are more likely to damage the robot itself or its environment. A shortcoming of their approach is that they need to collect explicit state-action pairs and their reversal actions, which makes it hard to scale to large environments. Several works [40, 5, 4] use reachability as a curiosity bonus for exploration: if the current state has a large estimated distance to previous states, it means that it is novel and the agent should be rewarded. Reachability and reversibility are related, in the sense that irreversible actions lead to states from which previous states are unreachable. Nevertheless, their motivations and ours diverge, and we learn reversibility through a less involved task than that of learning reachability. Nair et al. [33] learn to reverse trajectories that start from a goal state so as to generate realistic trajectories that reach similar goals. In contrast, we use reversibility to direct exploration and/or control, not for generating learning data. Closest to our work, Rahaman et al. [37] propose to learn a potential function of the states that increases with time, which can detect irreversibility to some extent. A drawback of the approach is that the potential function is learned using trajectories sampled from a random policy, which is a problem for many tasks where a random agent might fail to cover interesting parts of the state space. In comparison, our method does not use a potential function and learns jointly with the RL agent, which makes it a viable candidate for more complex tasks.

Safe exploration. Safe exploration aims at making sure that the actions of RL agents do not lead to negative or unrecoverable effects that would outweigh the long-term value of exploration [2]. Notably, previous works developed distinct approaches to avoid irreversible behavior: by incremental updates to safe policies [23, 18], which requires knowing such a policy in advance; by restricting policy search to ergodic policies [32] (*i.e.* that can always come back to any state visited), which is costly; by active exploration [28], where the learner can ask for rollouts instead of exploring potentially unsafe areas of the state space itself; and by computing regions of attraction [9] (the part of the state space where a controller can bring the system back to an equilibrium point), which requires prior knowledge of the environment dynamics.

Self-supervision from the arrow of time. Self-supervision has become a central component of modern machine learning algorithms, be it for computer vision, natural language or signal processing. In particular, using temporal consistency as a source of self-supervision is now ubiquitous, be it to learn representations for downstream tasks [19, 38, 12], or to learn to detect temporal inconsistencies [47]. The closest analogies to our work are methods that specifically estimate some aspects of the arrow of time as self-supervision. Most are to be found in the video processing literature, and self-supervised tasks include predicting which way the time flows [35, 47], verifying the temporal order of a subset of frames [30], predicting which video clip has the wrong temporal order among a subset [17] as well as reordering shuffled frames or clips from the video [16, 14, 48]. Bai et al. [6] notably propose to combine several of these pretext tasks along with data augmentation for video classification. Using time as a means of supervision was also explored for image sequencing [8], audio [11] or EEG processing [39]. In RL, self-supervision also gained momentum in recent years [22, 44, 49], with temporal information being featured [1]. Notably, several works [3, 13, 21, 43] leverage temporal consistency to learn useful representations, effectively learning to discriminate between observations that are temporally close and observations that are temporally distant. In comparison to all these works, we estimate the arrow of time through temporal order prediction with the explicit goal of finding irreversible transitions or actions.

3 Reversibility

Degree of Reversibility. We start by introducing formally the notion of reversibility. Intuitively, an action is reversible if it can be undone, meaning that there is a sequence of actions that can bring us back to the original state.

Definition 1. *Given a state s, we call* degree of reversibility within K steps *of an action a*

$$\phi_K(s,a) \coloneqq \sup_{\pi} p_\pi(s \in \tau_{t+1:t+K+1} \mid s_t = s, a_t = a),$$

and the degree of reversibility *of an action is defined as*

$$\phi(s,a) \coloneqq \sup_{\pi} p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s, a_t = a),$$

with $\tau = \{s_i\}_{i=1\dots T} \sim \pi$ corresponding to a trajectory, and $\tau_{t:t'}$ the subset of the trajectory between the timesteps t and t' (excluded). We omit their dependency on π for the sake of conciseness. Given $s \in S$, the action a is reversible *if and only if $\phi(s,a) = 1$, and said* irreversible *if and only if $\phi(s,a) = 0$.*

In deterministic environments, an action is either reversible or irreversible: given a state-action couple (s,a) and the unique resulting state s', $\phi_K(s,a)$ is equal to 1 if there is a sequence of less than K actions which brings the agent from s' to s, and is otherwise equal to zero. In stochastic environments, a given sequence of actions can only reverse a transition up to some probability, hence the need for the notion of degree of reversibility.

Policy-Dependent Reversibility. In practice, it is useful to quantify the degree of reversibility of an action as the agent acts according to a fixed policy π, for which we extend the notions introduced above. We simply write :

$$\phi_{\pi,K}(s,a) \coloneqq p_\pi(s \in \tau_{t+1:t+K+1} \mid s_t = s, a_t = a) \text{ and } \phi_\pi(s,a) \coloneqq p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s, a_t = a).$$

It immediately follows that $\phi_K(s,a) = \sup_\pi \phi_{\pi,K}(s,a)$ and $\phi(s,a) = \sup_\pi \phi_\pi(s,a)$.

4 Reversibility Estimation via Classification

Quantifying the exact degree of reversibility of actions is generally hard. In this section, we show that reversibility can be approximated efficiently using simple binary classification.

4.1 Precedence Estimation

Supposing that a trajectory contains the states s and s', we want to be able to establish *precedence*, that is predicting whether s or s' comes first *on average*. It is a binary classification problem, which

consists in estimating the quantity $\mathbb{E}_{s_t=s,s_{t'}=s'}\left[\mathbb{1}_{t'>t}\right]$. Accordingly, we introduce the precedence estimator which, using a set of trajectories, learns to predict which state of an arbitrary pair is most likely to come first.

Definition 2. *Given a fixed policy π, we define the* finite-horizon precedence estimator *between two states as follows:*

$$\psi_{\pi,T}(s,s') = \mathbb{E}_{\tau\sim\pi}\,\mathbb{E}_{\substack{s_t=s,s_{t'}=s'\\ t,t'<T}}\left[\mathbb{1}_{t'>t}\right].$$

Conceptually, given two states s and s', the precedence estimator gives an approximate probability of s' being visited after s, given that both s and s' are observed in a trajectory. The indices are sampled uniformly within the specified horizon $T \in \mathbb{N}$, so that this quantity is well-defined even for infinite trajectories. Additional properties of ψ, regarding transitivity for instance, can be found in Appx. A.2.

Remark 1. *The quantity $\psi_{\pi,T}(s,s')$ is only defined for pairs of states which can be found in the same trajectory, and is otherwise irrelevant. In what follows, we implicitly impose this condition when considering state pairs.*

Theorem 1. *For every policy π and $s, s' \in S$, $\psi_{\pi,T}(s,s')$ converges when T goes to infinity. We refer to the limit as the* precedence estimator*, written $\psi_\pi(s,s')$.*

The proof of this theorem is developed in Appendix A.3. This result is key to ground theoretically the notion of empirical reversibility ϕ, which we introduce in the next definition. It simply consists in extending the notion of precedence to a state-action pair.

Definition 3. *We finally define the* empirical reversibility *using the precedence estimator:*

$$\bar{\phi}_\pi(s,a) = \mathbb{E}_{s'\sim P(s,a)}\left[\psi_\pi(s',s)\right].$$

In a nutshell, given that we start in s and take the action a, the empirical reversibility $\bar{\phi}_\pi(s,a)$ measures the probability that we go back to s, starting from a state s' that follows (s,a). We now show that our empirical reversibility is linked with the notion of reversibility defined in the previous section, and can behave as a useful proxy.

4.2 Estimating Reversibility from Precedence

We present here our main theoretical result which relates reversibility and empirical reversibility:

Theorem 2. *Given a policy π, a state s and an action a, we have: $\bar{\phi}_\pi(s,a) \geq \frac{\phi_\pi(s,a)}{2}$.*

The full proof of the theorem is given in Appendix A.3.

This result theoretically justifies the name of empirical reversibility. From a practical perspective, it provides a way of using $\bar{\phi}$ to detect actions which are irreversible or hardly reversible: $\bar{\phi}_\pi(s,a) \ll 1$ implies $\phi_\pi(s,a) \ll 1$ and thus provides a sufficient condition to detect actions with low degrees of reversibility. This result gives a way to detect actions that are irreversible given a specific policy followed by the agent. Nevertheless, we are generally interested in knowing if these actions are irreversible for any policy, meaning $\phi(s,a) \ll 1$ with the definition of Section 3. The next proposition makes an explicit connection between $\bar{\phi}_\pi$ and ϕ, under the assumption that the policy π is stochastic.

Proposition 1. *We suppose that we are given a state s, an action a such that a is reversible in K steps, and a policy π. Under the assumption that π is stochastic enough, meaning that there exists $\rho > 0$ such that for every state and action s, a, $\pi(a \mid s) > \rho$, we have: $\bar{\phi}_\pi(s,a) \geq \frac{\rho^K}{2}$. Moreover, we have for all $K \in \mathbb{N}$: $\bar{\phi}_\pi(s,a) \geq \frac{\rho^K}{2}\phi_K(s,a)$.*

The proof is given in Appendix A.4. As before, this proposition gives a practical way of detecting irreversible moves. If for example $\bar{\phi}_\pi(s,a) < \rho^k/2$ for some $k \in \mathbb{N}$, we can be sure that action a is not reversible in k steps. The quantity ρ can be understood as a minimal probability of taking any action in any state. This condition is not very restrictive: ϵ-greedy strategies for example satisfy this hypothesis with $\rho = \frac{\epsilon}{|A|}$.

In practice, it can also be useful to limit the maximum number of time steps between two sampled states. That is why we also define the windowed precedence estimator as follows:

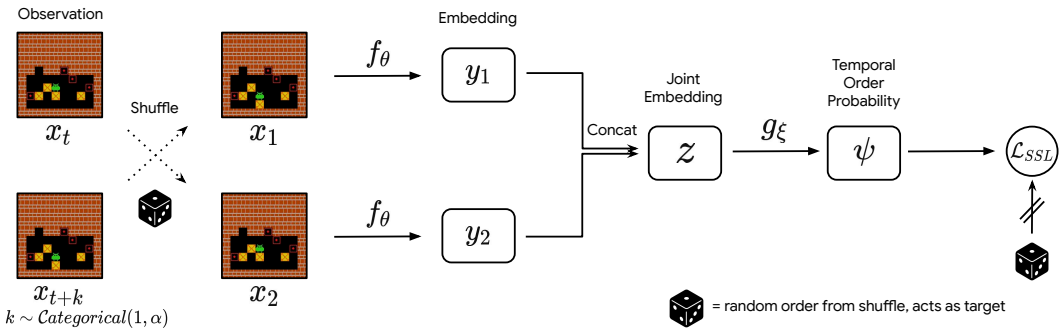


Figure 2: The proposed self-supervised procedure for precedence estimation.

Definition 4. *Given a fixed policy π, we define the* windowed precedence estimator *between two states as follows:*

$$\psi_{\pi,T,w}(s, s') = \mathbb{E}_{\tau\sim\pi}\mathbb{E}_{\substack{s_t=s, s_{t'}=s' \\ t,t'<T \\ |t-t'|\leq w}}\left[\mathbb{1}_{t'>t}\right].$$

Intuitively, compared to previous precedence estimators, $\psi_{\pi,T,w}$ is restricted to short-term dynamics, which is a desirable property in tasks where distinguishing the far future from the present is either trivial or impossible.

5 Reversibility-Aware Reinforcement Learning

Leveraging the theoretically-grounded bridge between precedence and reversibility established in the previous section, we now explain how reversibility can be learned from the agent's experience and used in a practical setting.

Learning to rank events chronologically. Learning which observation comes first in a trajectory is achieved by binary supervised classification, from pairs of observations sampled uniformly in a sliding window on observed trajectories. This can be done fully offline, *i.e.* using a previously collected dataset of trajectories for instance, or fully online, *i.e.* jointly with the learning of the RL agent; but also anywhere on the spectrum by leveraging variable amounts of offline and online data.

This procedure is not without caveats. In particular, we want to avoid overfitting to the particularities of the behavior of the agent, so that we can learn meaningful, generalizable statistics about the order of events in the task at hand. Indeed, if an agent always visits the state s_a before s_b, the classifier will probably assign a close-to-one probability that s_a precedes s_b. This might not be accurate with other agents equipped with different policies, unless transitioning from s_b to s_a is hard due to the dynamics of the environment, which is in fact exactly the cases we want to uncover. We make several assumptions about the agents we apply our method to: 1) agents are learning and thus, have a policy that changes through interactions in the environment, 2) agents have an incentive not to be too deterministic. For this second assumption, we typically use an entropic regularization in the chosen RL loss, which is a common design choice in modern RL methods. These assumptions, when put together, alleviate the risk of overfitting to the idiosyncrasies of a single, non-representative policy.

We illustrate the precedence classification procedure in Fig. 2. A temporally-ordered pair of observations, distant of no more than w timesteps, is sampled from a trajectory and uniformly shuffled. The result of the shuffling operation is memorized and used as a target for the binary classification task. A Siamese network creates separate embeddings for the pair of observations, which are concatenated and fed to a separate feed-forward network, whose output is passed through a sigmoid to obtain a probability of precedence. This probability is updated via negative log-likelihood against the result of the shuffle, so that it matches the actual temporal order.

Then, a transition (and its implicit sequence of actions) represented by a starting observation x and a resulting observation x' is deemed irreversible if the estimated precedence probability $\psi(x, x')$ is superior to a chosen threshold β. Note that we do not have to take into account the temporal proximity of these two observations here, which is a by-product of sampling observations uniformly

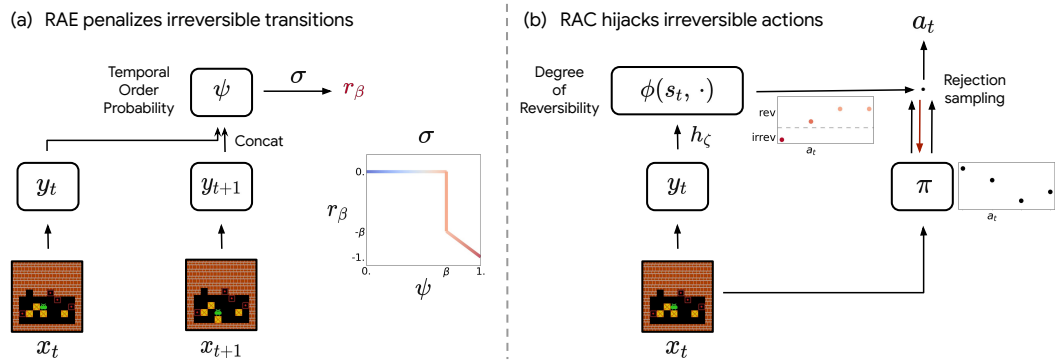


Figure 3: Our proposed methods for reversibility-aware RL. **(a):** RAE encourages reversible behavior via auxiliary rewards. **(b):** RAC avoids irreversible behavior by rejecting actions whose estimated reversibility is inferior to a threshold.

in a window in trajectories. Also, depending on the threshold β, we cover a wide range of scenarios, from pure irreversibility (β close to 1) to soft irreversibility ($\beta > 0.5$, the bigger β, the harder the transition is to reverse). This is useful because different tasks call for different levels of tolerance for irreversible behavior: while a robot getting stuck and leading to an early experiment failure is to be avoided when possible, tasks involving human safety might call for absolute zero tolerance for irreversible decision-making. We elaborate on these aspects in Sec. 6.

Reversibility-Aware Exploration and Control. We propose two different algorithms based on reversibility estimation: Reversibility-Aware Exploration (RAE) and Reversibility-Aware Control (RAC). We give a high-level representation of how the two methods operate in Fig. 3.

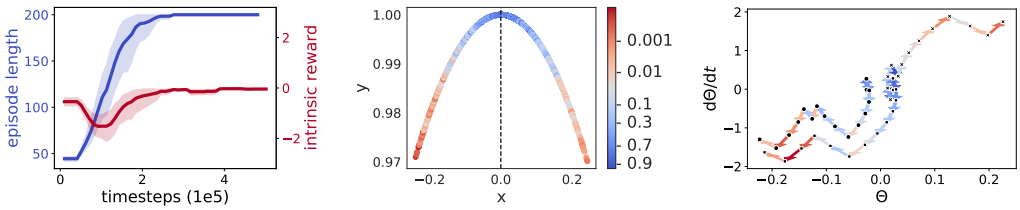
In a nutshell, RAE consists in using the estimated reversibility of a pair of consecutive observations to create an auxiliary reward function. In our experiments, the reward function is a piecewise linear function of the estimated reversibility and a fixed threshold, as in Fig. 3: it grants the agent a negative reward if the transition is deemed too hard to reverse. The agent optimizes the sum of the extrinsic and auxiliary rewards. Note that the specific function we use penalizes irreversible transitions but could encourage such transitions instead, if the task calls for it.

RAC can be seen as the action-conditioned counterpart of RAE. From a single observation, RAC estimates the degree of reversibility of all available actions, and "takes control" if the action sampled from the policy is not reversible enough (*i.e.* has a reversibility inferior to a threshold β). "Taking control" can have many forms. In practice, we opt for rejection sampling: we sample from the policy until an action that is reversible enough is sampled. This strategy has the advantage of avoiding irreversible actions entirely, while trading-off pure reversibility for performance when possible. RAC is more involved than RAE, since the action-conditioned reversibility is learned from the supervision of a standard, also learned precedence estimator. Nevertheless, our experiments show that it is possible to learn both estimators jointly, at the cost of little overhead.

We now discuss the relative merits of the two methods. In terms of applications, we argue that RAE is more suitable for directed exploration, as it only encourages reversible behavior. As a result, irreversible behavior is permitted if the benefits (*i.e.* rewards) outweigh the costs (*i.e.* irreversibility penalties). In contrast, RAC shines in safety-first, real-world scenarios, where irreversible behavior is to be banned entirely. With an optimal precedence estimator and task-dependent threshold, RAC will indeed hijack all irreversible sampled actions. RAC can be especially effective when pre-trained on offline trajectories: it is then possible to generate fully-reversible, safe behavior from the very first online interaction in the environment. We explore these possibilities experimentally in Sec. 6.2.

Both algorithms can be used online or offline with small modifications to their overall logic. The pseudo-code for the online version of RAE and RAC can be found in Appendix B.2.

The self-supervised precedence classification task could have applications beyond estimating the reversibility of actions: it could be used as a means of getting additional learning signal or representational priors for the RL algorithm. Nevertheless, we opt for a clear separation between the



(a) Training curves (b) Relative pole coordinates (c) Random trajectories

Figure 4: **(a):** Training curves of a PPO+RAE agent in reward-free Cartpole. Blue: episode length. Red: intrinsic reward. A 95% confidence interval over 10 random seeds is shown. **(b):** The x and y axes are the coordinates of the end of the pole relatively to the cart position. The color denotes the online reversibility estimation between two consecutive states (logit scale). **(c):** The representation of three random trajectories according to θ (angle of the pole) and $\frac{d\theta}{dt}$. Arrows are colored according to the learned reversibility of the transitions they correspond to.

reversibility and the RL components so that we can precisely attribute improvements to the former, and leave aforementioned studies for future work.

6 Experiments

The following experiments aim at demonstrating that the estimated precedence ψ is a good proxy for reversibility, and at illustrating how beneficial reversibility can be in various practical cases. We benchmark RAE and RAC on a diverse set of environments, with various types of observations (tabular, pixel-based), using neural networks for function approximation. See Appendix C for details.

6.1 Reward-Free Reinforcement Learning

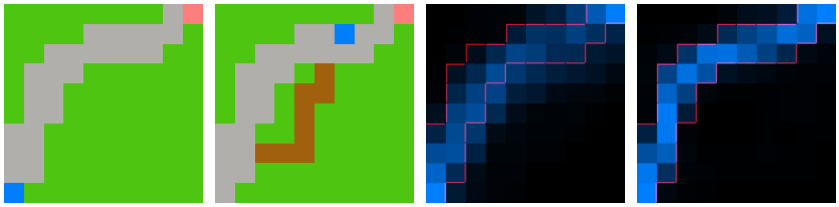
We illustrate the ability of RAE to learn sensible policies without access to rewards. We use the classic pole balancing task Cartpole [7], using the OpenAI Gym [10] implementation. In the usual setting, the agent gets a reward of 1 at every time step, such that the total undiscounted episode reward is equal to the episode length, and incentivizes the agent to learn a policy that stabilizes the pole. Here, instead, we remove this reward signal and give a PPO agent [42] an intrinsic reward based on the estimated reversibility, which is learned online from agent trajectories. The reward function penalizes irreversibility, as shown in Fig. 3. Note that creating insightful rewards is quite difficult: too frequent negative rewards could lead the agent to try and terminate the episode as soon as possible.

We display our results in Fig. 4. Fig. 4a confirms the claim that RAE can be used to learn meaningful rewards. Looking at the intrinsic reward, we discern three phases. Initially, both the policy and the reversibility classifier are untrained (and intrinsic rewards are 0). In the second phase, the classifier is fully trained but the agent still explores randomly (intrinsic rewards become negative). Finally, the agent adapts its behavior to avoid penalties (intrinsic rewards go to 0, and the length of trajectories increases). Our reward-free agent reaches the score of 200, which is the highest possible score.

To further assess the quality of the learned reversibility, we freeze the classifier after 300k timesteps and display its predicted probabilities according to the relative coordinates of the end of the pole (Fig. 4b) and the dynamics of the angle of the pole θ (Fig. 4c). In both cases, the empirical reversibility matches our intuition: the reversibility should decrease as the angle or angular momentum increase, since these coincide with an increasing difficulty to go back to the equilibrium.

6.2 Learning Reversible Policies

In this section, we investigate how RAE can be used to learn reversible policies. When we train an agent to achieve a goal, we usually want it to achieve that goal following implicit safety constraints. Handcrafting such safety constraints would be time-consuming, difficult to scale for complex problems, and might lead to reward hacking; so a reasonable proxy consists in limiting irreversible side-effects in the environment [27].



(a) Initial state (b) A trajectory (c) PPO (500k) (d) PPO+RAE (500k)

Figure 5: **(a):** The Turf environment. The agent can walk on grass, but the grass then turns brown. **(b):** An illustrative trajectory where the agent stepped on grass pixels. **(c):** State visitation heatmap for PPO. **(d):** State visitation heatmap for PPO+RAE. It coincides with the stone path (red).

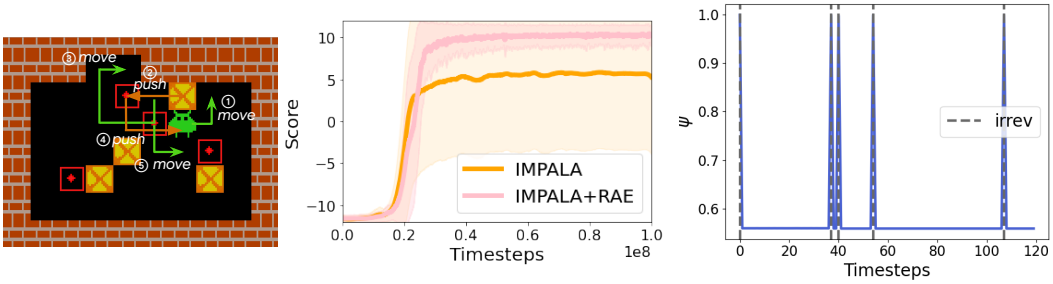


Figure 6: **(a):** Non-trivial reversibility: pushing the box against the wall can be reversed by pushing it to the left, going around, pushing it down and going back to start. A minimum of 17 moves is required to go back to the starting state. **(b):** Performances of IMPALA and IMPALA+RAE on 1k levels of Sokoban (5 seeds average). **(c):** Evolution of the estimated reversibility along one episode.

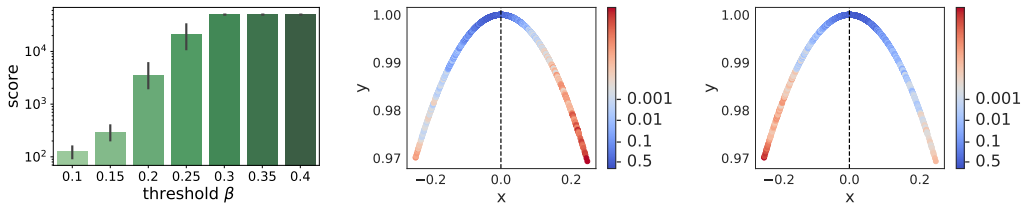
To quantify side-effects, we propose Turf, a new synthetic environment. As depicted in Fig. 5a,5b, the agent (blue) is rewarded when reaching the goal (pink). Stepping on grass (green) will spoil it, causing it to turn brown. Stepping on the stone path (grey) does not induce any side-effect.

In Fig. 5c,5d, we compare the behaviors of a trained PPO agent with and without RAE. The baseline agent is indifferent to the path to the goal, while the agent benefitting from RAE learns to follow the road, avoiding irreversible consequences.

6.3 Sokoban

Sokoban is a popular puzzle game where a warehouse keeper (controlled by the player) must move boxes around and place them in dedicated places. Each level is unique and involves planning, since there are many ways to get stuck. For instance, pushing a box against a wall is often un-undoable, and prevents the completion of the level unless actually required to place the box on a specific target. Sokoban is a challenge to current model-free RL algorithms, as advanced agents require millions of interactions to reliably solve a fraction of levels [46, 20]. One of the reasons for this is tied to exploration: since agents learn from scratch, there is a long preliminary phase where they act randomly in order to explore the different levels. During this phase, the agent will lock itself in unrecoverable states many times, and further exploration is wasted. It is worth recalling that contrary to human players, the agent does not have the option to reset the game when stuck. In these regards, Sokoban is a great testbed for reversibility-aware approaches, as we expect them to make the exploration phase more efficient, by incorporating the prior that irreversible transitions are to be avoided if possible, and by providing tools to identify such transitions.

We benchmark performance on a set of 1k levels. Results are displayed in Fig. 6. Equipping an IMPALA agent [15] with RAE leads to a visible performance increase, and the resulting agent consistently solves all levels from the set. We take a closer look at the reversibility estimates and show that they match the ground truth with high accuracy, despite the high imbalance of the distribution (*i.e.* few irreversible transitions, see Fig. 6c) and complex reversibility estim ation (see Fig. 6a).



(a) Trajectory lengths (b) Coordinates: action 0 (c) Coordinates: action 1

Figure 7: **(a):** Mean score of a random policy augmented with RAC on Cartpole+ for several threshold values, with 95% confidence intervals over 10 random seeds (log scale). **(b) and (c):** The x and y axes are the coordinates of the end of the pole relatively to the cart position. The color indicates the estimated reversibility values.

6.4 Safe Control

In this section, we put an emphasis on RAC, which is particularly suited for safety related tasks.

Cartpole+. We use the standard Cartpole environment, except that we change the maximum number of steps from 200 to 50k to study long-term policy stability. We name this new environment Cartpole+. It is substantially more difficult than the initial setting. We learn reversibility offline, using trajectories collected from a random policy. Fig. 7a shows that a random policy augmented with RAC achieves seemingly infinite scores. For the sake of comparison, we indicate that a DQN [31] and the state-of-the-art M-DQN [45] achieve a maximum score of respectively 1152 and 2801 under a standard training procedure, described in Appendix C.5. This can be surprising, since RAC was only trained on random thus short trajectories (mean length of 20). We illustrate the predictions of our learned estimator in Fig. 7b,7c. When the pole leans to the left ($x < 0$), we can see that moving the cart to the left is perceived as more reversible than moving it to the right. This is key to the good performance of RAC and perfectly on par with our understanding of physics: when the pole is leaning in a direction, agents must move the cart in the same direction to stabilize it.

Turf. We now illustrate how RAC can be used for safe online learning: the implicitly safe constraints provided by RAC prevent policies from deviating from safe trajectories. This ensures for example that agents stay in recoverable zones during exploration.

We learn the reversibility estimator offline, using the trajectories of a random policy. We reject actions whose reversibility is deemed inferior to $\beta = 0.2$, and train a PPO agent with RAC. As displayed in Fig. 8, PPO with RAC learns to reach the goal without causing any irreversible side-effect (*i.e.* stepping on grass) during the whole training process.

The threshold β is a very important parameter of the algorithm. Too low a threshold could lead to overlooking some irreversible actions, while a high threshold could prevent the agent from learning the new task at hand. We discuss this performance/safety trade-off in more details in Appendix. C.7.

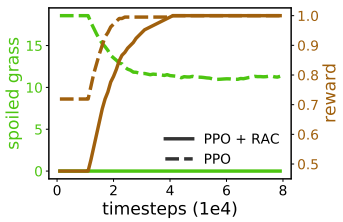


Figure 8: PPO and RAC (solid lines) vs PPO (dashed lines). At the cost of slower learning (brown), our approach prevents the agent from producing a single irreversible side-effect (green) during the learning phase. Curves are averaged over 10 runs.

7 Conclusion

In this work, we formalized the link between the reversibility of transitions and their temporal order, which inspired a self-supervised procedure to learn the reversibility of actions from experience. In combination with two novel reversibility-aware exploration strategies, RAE for directed exploration and RAC for directed control, we showed the empirical benefits of our approach in various scenarios, ranging from safe RL to risk-averse exploration. Notably, we demonstrated increased performance in procedurally-generated Sokoban puzzles, which we tied to more efficient exploration.

Broader impact and ethical considerations. The presented work aims at estimating and controlling potentially irreversible behaviors in RL agents. We think it has interesting applications in safety-first scenarios, where irreversible behavior or side-effects are to be avoided. The societal implication of these effects would be safer interactions with RL-powered components (*e.g.* robots, virtual assistants, recommender systems) which, though rare today, could become the norm. We argue that further research and applications should verify that the induced reversible behavior holds in almost all situations and does not lead to unintended effects. Our method could be deflected from its goal and used to identify and encourage actions with irreversible effects. In this case, a counter measure consists in using our method to flag and replace irreversible actions. Hence, while the method provides information that could be used to deal irreversible harm, we argue that it provides equal capabilities for detection and prevention.

Acknowledgments and Disclosure of Funding

Experiments presented in this paper were partially carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations. NG is a recipient of PhD funding from the AMX program, Ecole Polytechnique. The authors would like to thank Edouard Leurent, Antoine Moulin, Odalric-Ambrym Maillard, Léonard Hussenot, Nino Vieillard, Alexis Jacq, Théophane Weber and Bobak Shahriari for helpful comments and suggestions.

References

[1] A. Amiranashvili, A. Dosovitskiy, V. Koltun, and T. Brox. Motion perception in reinforcement learning with dynamic objects. In *Conference on Robot Learning*, 2018.

[2] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[3] Y. Aytar, T. Pfaff, D. Budden, T. L. Paine, Z. Wang, and N. de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, 2018.

[4] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, 2020.

[5] A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, et al. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*, 2020.

[6] Y. Bai, H. Fan, I. Misra, G. Venkatesh, Y. Lu, Y. Zhou, Q. Yu, V. Chandra, and A. Yuille. Can temporal information help with contrastive self-supervised learning? *arXiv preprint arXiv:2011.13046*, 2020.

[7] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. In *IEEE Transactions on Systems, Man, and Cybernetics*, 1983.

[8] T. Basha, Y. Moses, and S. Avidan. Photo sequencing. In *European Conference on Computer Vision*, 2012.

[9] F. Berkenkamp, R. Moriconi, A. Schoellig, and A. Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. In *Conference on Decision and Control*, 2016.

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016. URL http://arxiv.org/abs/1606.01540.

[11] A. N. Carr, Q. Berthet, M. Blondel, O. Teboul, and N. Zeghidour. Self-supervised learning of audio representations from permutations with differentiable ranking. In *IEEE Signal Processing Letters*, 2021.

[12] R. Dadashi, L. Hussenot, M. Geist, and O. Pietquin. Primal wasserstein imitation learning. In *International Conference on Learning Representations*, 2020.

[13] D. Dwibedi, J. Tompson, C. Lynch, and P. Sermanet. Learning actionable representations from visual observations. In *International Conference on Intelligent Robots and Systems*, 2018.

[14] A. El-Nouby, S. Zhai, G. W. Taylor, and J. M. Susskind. Skip-clip: Self-supervised spatiotemporal representation learning by future clip order ranking. *arXiv preprint arXiv:1910.12770*, 2019.

[15] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, 2018.

[16] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *Conference on Computer Vision and Pattern Recognition*, 2015.

[17] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *Conference on Computer Vision and Pattern Recognition*, 2017.

[18] J. García and F. Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 2012.

[19] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *International Conference on Computer Vision*, 2015.

[20] A. Guez, M. Mirza, K. Gregor, et al. An investigation of model-free planning. In *International Conference on Machine Learning*, 2019.

[21] Z. D. Guo, M. G. Azar, B. Piot, B. A. Pires, and R. Munos. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018.

[22] Z. D. Guo, B. A. Pires, B. Piot, J.-B. Grill, F. Altché, R. Munos, and M. G. Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, 2020.

[23] A. Hans, D. Schneegaß, A. M. Schäfer, and S. Udluft. Safe exploration for reinforcement learning. In *European Symposium on Artificial Neural Networks*, 2008.

[24] M. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[26] M. Kruusmaa, Y. Gavshin, and A. Eppendahl. Don't do things you can't undo: Reversibility models for generating safe behaviours. In *International Conference on Robotics and Automation*, 2007.

[27] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg. AI safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

[28] O.-A. Maillard, T. Mann, R. Ortner, and S. Mannor. Active Rollouts in MDP with Irreversible Dynamics. *Hal preprint hal-02177808*, 2019.

[29] D. W. McAllister, T. R. Mitchell, and L. R. Beach. The contingency model for the selection of decision strategies: An empirical test of the effects of significance, accountability, and reversibility. In *Organizational Behavior and Human Performance*, 1979.

[30] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, 2016.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[32] T. M. Moldovan and P. Abbeel. Safe exploration in markov decision processes. In *International Conference on Machine Learning*, 2012.

[33] S. Nair, M. Babaeizadeh, C. Finn, S. Levine, and V. Kumar. Time reversal as self-supervision. In *International Conference on Robotics and Automation*, 2020.

[34] J. R. Norris. *Markov chains*. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998. ISBN 978-0-521-48181-6.

[35] L. C. Pickup, Z. Pan, D. Wei, Y. C. Shih, C. Zhang, A. Zisserman, B. Scholkopf, and W. T. Freeman. Seeing the arrow of time. In *Conference on Computer Vision and Pattern Recognition*, 2014.

[36] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. Stable baselines3. https://github.com/DLR-RM/stable-baselines3, 2019.

[37] N. Rahaman, S. Wolf, A. Goyal, R. Remme, and Y. Bengio. Learning the arrow of time for problems in reinforcement learning. In *International Conference on Learning Representations*, 2020.

[38] V. Ramanathan, K. Tang, G. Mori, and L. Fei-Fei. Learning temporal embeddings for complex video analysis. In *International Conference on Computer Vision*, 2015.

[39] A. Saeed, D. Grangier, O. Pietquin, and N. Zeghidour. Learning from heterogeneous eeg signals with differentiable channel reordering. In *International Conference on Acoustics, Speech and Signal Processing*, 2020.

[40] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly. Episodic curiosity through reachability. In *International Conference on Learning Representations*, 2019.

[41] M.-P. B. Schrader. gym-sokoban. https://github.com/mpSchrader/gym-sokoban, 2018.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *International Conference on Robotics and Automation*, 2018.

[44] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020.

[45] N. Vieillard, O. Pietquin, and M. Geist. Munchausen reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[46] T. Weber, S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

[47] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[48] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[49] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, 2021.

[50] M. Zeelenberg. Anticipated regret, expected feedback and behavioral decision making. In *Journal of Behavioral Decision Making*, 1999.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Practical caveats and limitations are discussed at the beginning of Sec. 5 and in Sec. 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sec. 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Theoretical assumptions are described along the propositions and theorems in Sec. 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] Complete proofs are given in Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We released the code for every experiments except Sokoban (Sec. 6.3) as some components are proprietary. Additionally, the pseudo-code for the proposed methods is given in Appendix B.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

We organize the supplementary material as follows: in Appendix A, we include the proofs of results from the main text, as well as additional formalism; in Appendix B, we provide additional details about the proposed algorithms, including pseudo-code and figures that did not fit in the main text; and in Appendix C, we detail our experimental procedure, including hyperparameters for all methods/

A Mathematical Elements and Proofs

A.1 Possible Definitions of Reversibility

In this section, we present several intuitive definitions of reversibility in MDPs. We chose the third definition as our reference, which we argue presents several advantages over the others, although they can be interesting in specific contexts. Indeed, Eq. (3) is simpler than Eq. (1), as it does not depend on the discount factor, and more general than Eq. (2), as it does not enforce a fixed number of timesteps for going back to the starting state.

Discounted Reward.

$$\phi_{\pi,K}(s,a) \coloneqq \sum_{k>t}^{K} \gamma^{k-t} p_\pi(s_{t+k} = s \mid s_t = s, a_t = a)\,, \tag{1}$$
$$\phi_{\pi}(s,a) \coloneqq \sum_{k>t}^{\infty} \gamma^{k-t} p_\pi(s_{t+k} = s \mid s_t = s, a_t = a).$$

Fixed Time Step.

$$\phi_{\pi,K}(s,a) \coloneqq \sup_{k\leq K} p_\pi(s_{t+k} = s \mid s_t = s, a_t = a)\,, \tag{2}$$
$$\phi_{\pi}(s,a) \coloneqq \sup_{k\in\mathbb{N}} p_\pi(s_{t+k} = s \mid s_t = s, a_t = a).$$

Undiscounted Reward.

$$\begin{aligned}\phi_{\pi,K}(s,a) &\coloneqq \sum_{k=1}^{K} p_\pi(s_{t+k} = s, s_{t+k-1} \neq s, \dots, s_{t+1} \neq s \mid s_t = s, a_t = a)\,,\\ &= p_\pi(s \in \tau_{t+1:t+K+1} \mid s_t = s, a_t = a)\,. \qquad (3)\\ \phi_{\pi}(s,a) &\coloneqq \sum_{k=1}^{\infty} p_\pi(s_{t+k} = s, s_{t+k-1} \neq s, \dots, s_{t+1} \neq s \mid s_t = s, a_t = a)\,,\\ &= p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s, a_t = a).\end{aligned}$$

A.2 Additional Properties

We write $s \to s'$ if $\psi_\pi(s,s') \geq 0.5$ ("it is more likely to go from s to s' than to go from s' to s") and $s \Rightarrow s'$ if $\psi_\pi(s,s') = 1$ ("it is possible to go from s to s', but it is not possible to come back to s from s'").

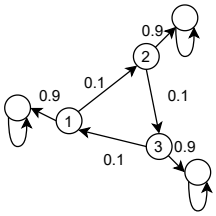


Figure 9: Counter-example for additional property 4. The initial state is sampled uniformly amongst $\{0,1,2\}$.

1. $\psi_\pi(s,s') + \psi_\pi(s',s) = 1$
2. if $s_0 \Rightarrow s_1 \Rightarrow s_2$ then $s_0 \Rightarrow s_2$ (transitivity for $\Rightarrow$)
3. if $s_0 \to s_1 \to \dots \to s_i \Rightarrow s_{i+1} \to \dots \to s_t$ then $s_0 \Rightarrow s_t$
4. in general $s_1 \to s_2$ and $s_2 \to s_3$ doesn't imply $s_1 \to s_3$

Proofs:

(1) $\psi_\pi(s,s') + \psi_\pi(s',s) = \mathbb{E}_{\tau\sim\pi}\mathbb{E}_{t\neq t'|s_t=s,s_{t'}=s'}\big[\mathbb{1}_{t'>t} + \mathbb{1}_{t'<t}\big] = \mathbb{E}_{\tau\sim\pi}\mathbb{E}_{t\neq t'|s_t=s,s_{t'}=s'}\big[1\big] = 1$.

(2) and (3): As (3) is stronger than (2), we only prove (3). If it is possible to have s_0 after s_t in a trajectory, then it is possible to have s_i after s_t. As we have a positive probability of seeing s_t after s_{i+1}, we have a positive probability of seeing s_i after s_{i+1}, which contradicts $s_i \Rightarrow s_{i+1}$.

(4) A counter example can be found in Fig. 9. In this case we clearly have $s_1 \to s_2$, $s_2 \to s_3$ and $s_3 \to s_1$.

A.3 Proofs of Theorem 1 and Theorem 2

In the following, we prove simultaneously Theorem 1 and Theorem 2. We begin by two lemmas.

Lemma 1. *Given a trajectory τ, we denote by $\#_T(s \to s')$ the number of pairs (s, s') in $\tau_{1:T}$ such that s appears before s'. We present a simple formula for $\psi(s', s)$ according to the structure of the state trajectory:*

$$\psi_{\pi,T}(s, s') = \frac{\mathbb{E}_{\tau\sim\pi}\left[\#_T(s \to s')\right]}{\mathbb{E}_{\tau\sim\pi}\left[\#_T(s \to s') + \#_T(s' \to s)\right]} .$$

Proof. In order to simplify the notations, we leave implicit the fact that indices are always sampled within $[0, T]$.

$$\begin{aligned}\psi_{\pi,T}(s, s') &= \mathbb{E}_\pi \mathbb{E}_{t\neq t' | s_t = s, s_{t'} = s'}\left[\mathbb{1}_{t'>t}\right] , \\ &= \frac{\mathbb{E}_\pi \mathbb{E}_{t\neq t'}\left[\mathbb{1}_{t'>t}\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right]}{\mathbb{E}_\pi \mathbb{E}_{t\neq t'}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right]} .\end{aligned}$$

Similarly, we have:

$$\mathbb{E}_\pi \mathbb{E}_{t'>t}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right] = \frac{\mathbb{E}_\pi \mathbb{E}_{t\neq t'}\left[\mathbb{1}_{t'>t}\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right]}{\mathbb{E}_{t\neq t'}\left[\mathbb{1}_{t'>t}\right]} .$$

Combining it with our previous equation:

$$\begin{aligned}\psi_{\pi,T}(s, s') &= \frac{\mathbb{E}_\pi \mathbb{E}_{t'>t}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right]\mathbb{E}_{t\neq t'}\left[\mathbb{1}_{t'>t}\right]}{\mathbb{E}_\pi \mathbb{E}_{t\neq t'},\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right]} , \\ &= \frac{1}{2}\frac{\mathbb{E}_\pi \mathbb{E}_{t'>t}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right]}{\mathbb{E}_\pi \mathbb{E}_{t\neq t'}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right]} .\end{aligned}$$

Looking at the denominator, we can notice:

$$\begin{aligned}\mathbb{E}_\pi \mathbb{E}_{t\neq t'}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right] &= \frac{1}{2}\mathbb{E}_\pi \mathbb{E}_{t<t'}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right] + \frac{1}{2}\mathbb{E}_\pi \mathbb{E}_{t'<t}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right] , \\ &= \frac{1}{2}\mathbb{E}_\pi \mathbb{E}_{t<t'}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'} + \mathbb{1}_{s_t=s'}\mathbb{1}_{s_{t'}=s}\right] ,\end{aligned}$$

which comes from the fact that t and t' play a symmetrical role. Thus,

$$\psi_{\pi,T}(s, s') = \frac{\mathbb{E}_{\tau\sim\pi}\mathbb{E}_t \mathbb{E}_{t'>t}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right]}{\mathbb{E}_{\tau\sim\pi}\mathbb{E}_t \mathbb{E}_{t'>t}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'} + \mathbb{1}_{s_t=s'}\mathbb{1}_{s_{t'}=s}\right]} .$$

Since

$$\begin{aligned}\mathbb{E}_{\tau\sim\pi}\left[\#_T(s \to s')\right] &= \sum_{i<j\leq T} \mathbb{1}_{s_i=s}\mathbb{1}_{s_j=s'} , \\ &= \binom{T}{2}\sum_{i<j\leq T}\frac{1}{\binom{T}{2}}\mathbb{1}_{s_i=s}\mathbb{1}_{s_j=s'} , \\ &= \binom{T}{2}\mathbb{E}_{\tau\sim\pi}\mathbb{E}_t \mathbb{E}_{t'>t}\left[\mathbb{1}_{s_t=s}\mathbb{1}_{s_{t'}=s'}\right] ,\end{aligned}$$

we get:

$$\psi_{\pi,T}(s,s') = \frac{\binom{T}{2}\mathbb{E}_{\tau\sim\pi}\big[\#_T(s\to s')\big]}{\binom{T}{2}\mathbb{E}_{\tau\sim\pi}\big[\#_T(s\to s')+\#_T(s'\to s)\big]}\,,$$
$$\psi_{\pi,T}(s,s') = \frac{\mathbb{E}_{\tau\sim\pi}\big[\#_T(s\to s')\big]}{\mathbb{E}_{\tau\sim\pi}\big[\#_T(s\to s')+\#_T(s'\to s)\big]}\,.$$

□

Lemma 2. *Assume that we are given a fixed trajectory where s appears $k \in \mathbb{N}$ times, in the form of :*

$$s_0 \underbrace{\longrightarrow}_{n_0(s')} s \underbrace{\longrightarrow}_{n_1(s')} s \underbrace{\longrightarrow}_{n_2(s')} s \underbrace{\longrightarrow}_{n_3(s')} \cdots \underbrace{\longrightarrow}_{n_{k-1}(s')} s \underbrace{\longrightarrow}_{n_k(s')},$$

where $n_i(s')$ denotes the number of times s' appears between the ith and the $(i+1)^{th}$ occurrence of s. In this case,

$$\#_T(s\to s') = \sum_{i=0}^{k} i \times n_i(s')\,. \tag{4}$$

If we suppose that $n_1(s') = n_2(s') = \cdots = n_{k-1}(s')$, we also have

$$\#_T(s\to s') - \#_T(s'\to s) = k\left(n_k(s') - n_0(s')\right). \tag{5}$$

Proof. Eq. (4) comes directly from $\#_T(s\to s') = \sum_{i=1}^{k}\sum_{j=i}^{k} n_j(s') = \sum_{i=0}^{k} i\times n_i(s')$. To prove Equ. (5), we first notice that $\#_T(s\to s') + \#_T(s'\to s) = k \times \sum_{i=0}^{k} n_i(s')$. Thus

$$\begin{aligned}\#_T(s\to s') - \#_T(s'\to s) &= 2\times\#_T(s\to s') - \big(\#_T(s\to s') + \#_T(s'\to s)\big)\,,\\ &= 2\left(k\,n_k(s') + n_1(s')\sum_{i=0}^{k-1} i\right) - \big(k\,n_k(s') + k\,n_0(s') + k\,(k-1)\,n_1(s')\big),\\ &= k\,n_k(s') - k\,n_0(s')\,.\end{aligned}$$

□

Theorem 1. *For every policy π and $s, s' \in S$, $\psi_{\pi,T}(s,s')$ converges when T goes to infinity.*

Theorem 2. *Given a policy π, a state s, and an action a, we can link reversibility and empirical reversibility with the inequality: $\bar{\phi}_\pi(s,a) \geq \frac{\phi_\pi(s,a)}{2}$.*

Proof. For a policy π and $s, s' \in S$, we define $\hat{\phi}_\pi(s,s')$ the quantity $p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s')$ such that $\phi_\pi(s,a) = \mathbb{E}_{s'\sim P(s,a)}\left[\hat{\phi}_\pi(s,s')\right]$ In order to prove the theorem, we first prove that $\psi_T(s',s)$ converges to a quantity denoted by $\psi(s',s)$, and that:

$$\forall s,s' \in S, \frac{\hat{\phi}^\pi(s,s')}{2} \leq \psi(s',s)\,. \tag{6}$$

We subdivide our problem into four cases, depending on whether s and s' are recurrent or transient.

Case 1: $p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s) < 1$ and $p_\pi(s' \in \tau_{t+1:\infty} \mid s_t = s') = 1$ (s is transient and s' is recurrent for the Markov chain induced by π). Informally, this means that if a trajectory contains the state s' we tend to see s' an infinite number of times, and we only see s a finite number of times in a given trajectory.

This implies $\hat{\phi}_\pi(s,s') = p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s') = 0$, as recurrent states can only be linked to other recurrent states [34]. It is not possible to find trajectories where s appears after s', thus $\psi_T(s',s) = 0 = \psi(s',s)$. Equ. (6) becomes "$0 \leq 0$".

Case 2: $p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s) = 1$ and $p_\pi(s' \in \tau_{t+1:\infty} \mid s_t = s') < 1$ (s is recurrent and s' is transient for the Markov chain induced by π).

As before, this implies $\hat{\phi}_\pi(s', s) = p_\pi(s' \in \tau_{t+1:\infty} \mid s_t = s) = 0$, and thus it is not possible to see in a trajectory s after s'. It implies $\psi_T(s', s) = 1 = \psi(s', s)$, so Equ. (6) is verified.

Case 3: $p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s) = 1$ and $p_\pi(s' \in \tau_{t+1:\infty} \mid s_t = s') = 1$ (s is recurrent and s' is recurrent for the Markov chain induced by π). We denote by T_k the random variable corresponding to the time of the kth visit to s. A trajectory can be represented as follows:

$$s_0 \underbrace{\longrightarrow}_{n_1(s')} s \underbrace{\longrightarrow}_{n_2(s')} s \underbrace{\longrightarrow}_{n_3(s')} s \underbrace{\longrightarrow}_{n_4(s')} \cdots \underbrace{\longrightarrow}_{n_k(s')} s = s_{T_k} \underbrace{\longrightarrow}_{n_{k+1}(s')} ,$$

where, writing $\sim$ the equality in distribution, $n_2(s') \sim n_3(s') \sim \cdots \sim n_k(s')$ and $\mathbb{E}_\tau n_2(s') = \mathbb{E}_\tau n_3(s') = \cdots = \mathbb{E}_\tau n_k(s')$ using the strong Markov property. From Lemma 1 we get:

$$\begin{aligned}
\psi_{\pi,T}(s, s') &= \frac{\mathbb{E}_{\tau\sim\pi}\big[\#_T(s \to s')\big]}{\mathbb{E}_{\tau\sim\pi}\big[\#_T(s \to s') + \#_T(s' \to s)\big]} , \\
&= \frac{1}{2} \frac{\mathbb{E}_{\tau\sim\pi}\big[\#_T(s \to s') + \#_T(s' \to s) + \#_T(s \to s') - \#_T(s' \to s)\big]}{\mathbb{E}_{\tau\sim\pi}\big[\#_T(s \to s') + \#_T(s' \to s)\big]} , \\
&= \frac{1}{2} + \frac{\mathbb{E}_{\tau\sim\pi}\big[\#_T(s \to s') - \#_T(s' \to s)\big]}{2\,\mathbb{E}_{\tau\sim\pi}\big[\#_T(s \to s') + \#_T(s' \to s)\big]} .
\end{aligned}$$

We can see from Lemma 2 :

$$\mathbb{E}_\tau\big[\#_{T_k}(s \to s') - \#_{T_k}(s' \to s)\big] = -k\,\mathbb{E}_\tau n_1(s') .$$

Thus,

$$\begin{aligned}
\frac{\mathbb{E}_\tau\big[\#_{T_k}(s \to s') - \#_{T_k}(s' \to s)\big]}{\mathbb{E}_{\tau\sim\pi}\big[\#_{T_k}(s \to s') + \#_{T_k}(s' \to s)\big]} &= \frac{-k\,\mathbb{E}_\tau n_1(s')}{k\,\mathbb{E}_\tau n_1(s') + k^2\,\mathbb{E}_\tau n_2(s')} \\
&\xrightarrow[k\to\infty]{} 0.
\end{aligned} \tag{7}$$

Given $t \in \mathbb{N}$ and a trajectory τ, we denote $\#_T(s)$ the random variable corresponding to the number of times when s appear before t, such that a trajectory has the following structure :

$$s_0 \underbrace{\longrightarrow}_{n_1(s')} s \underbrace{\longrightarrow}_{n_2(s')} s \underbrace{\longrightarrow}_{n_3(s')} s \underbrace{\longrightarrow}_{n_4(s')} \cdots \underbrace{\longrightarrow}_{n_k(s')} s = s_{T_{\#_T(s)}} \underbrace{\longrightarrow s_t \longrightarrow}_{n_{k+1}(s')} s = s_{T_{\#_T(s)+1}} .$$

$$\frac{\mathbb{E}_\tau\big[\#_T(s \to s') - \#_T(s' \to s)\big]}{\mathbb{E}_\tau\big[\#_T(s \to s') + \#_T(s' \to s)\big]} \leq \frac{\mathbb{E}_\tau\big[\#_{\#_T(s)}(s \to s') - \#_{\#_T(s)}(s' \to s)\big] + \mathbb{E}_\tau \#_T(s) n_{k+1}(s')}{\mathbb{E}_\tau \#_{\#_T(s)}(s \to s') + \mathbb{E}_\tau \#_{\#_T(s)}(s' \to s)} ,$$

$$\xrightarrow[T\to\infty]{} 0 \text{ as in Equ. (7).}$$

And,

$$\frac{\mathbb{E}_\tau\big[\#_T(s \to s') - \#_T(s' \to s)\big]}{\mathbb{E}_\tau\big[\#_T(s \to s') + \#_T(s' \to s)\big]} \geq \frac{\mathbb{E}_\tau\big[\#_{\#_T(s)}(s \to s') - \#_{\#_T(s)}(s' \to s)\big] - \mathbb{E}_\tau \sum_{i=1}^{\#_T(s)+1} n_i(s')}{\mathbb{E}_\tau\big[\#_{\#_T(s)}(s \to s') + \#_{\#_T(s)}(s' \to s)\big]} ,$$

$$\xrightarrow[T\to\infty]{} 0$$

Therefore,

$$\frac{\mathbb{E}_\tau\big[\#_T(s \to s') - \#_T(s' \to s)\big]}{\mathbb{E}_\tau\big[\#_T(s \to s') + \#_T(s' \to s)\big]} \xrightarrow[T\to\infty]{} 0 \text{ , and finally,}$$

$$\psi_{\pi,T}(s,s') = \frac{1}{2} + \frac{\mathbb{E}_\tau\big[\#_T(s \to s') - \#_T(s' \to s)\big]}{2\mathbb{E}_{\tau\sim\pi}\big[\#_T(s \to s') + \#_T(s' \to s)\big]} \xrightarrow[T\to\infty]{} \frac{1}{2}\,.$$

As $\hat{\phi}_\pi(s,s') = 1$ here, we immediately have $\frac{\hat{\phi}_\pi(s,s')}{2} = \psi(s',s)$. We can notice that the inequality is tight in this case.

Case 4: $p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s) < 1$ and $p_\pi(s' \in \tau_{t+1:\infty} \mid s_t = s') < 1$ (s is transient and s' is transient for the Markov chain induced by π). To simplify the following formulas, we will write $\alpha = p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s')$. Here, we denote by $\#(s)$ the random variable corresponding to the total number of visits of the state s, and $\#(s \to s')$ the number of pairs such that s appears before s'. $\#(s)$ follows the geometric distribution $G\left(1 - p_\pi(s \in \tau_{t+1:\infty} \mid s_t = s)\right)$.

$\#_T(s \to s')$ converges almost surely to $\#(s \to s')$, and we have $\#_T(s \to s') \leq \#(s \to s')$. Therefore, using the dominated convergence theorem, $\mathbb{E}_\tau\big[\#_T(s \to s')\big] \xrightarrow[T\to\infty]{} \mathbb{E}_\tau\big[\#(s \to s')\,\big]$, and thus:

$$\psi_{\pi,T}(s',s) = \frac{\mathbb{E}_\tau\big[\#_T(s' \to s)\big]}{\mathbb{E}_\tau\big[\#_T(s \to s') + \#_T(s' \to s)\big]} \xrightarrow[T\to\infty]{} \frac{\mathbb{E}_\tau\#(s' \to s)}{\mathbb{E}_\tau\big[\#(s \to s') + \#(s' \to s)\big]} = \psi^\pi(s',s)\,.$$

This time, we consider a trajectory τ where s appears k times after s', such that it is of the form:

$$\underbrace{s'\ldots s'}_{n_0(s')\geq 0} \longrightarrow \underbrace{s\ldots s}_{n_1(s)>0} \longrightarrow \underbrace{s'\ldots s'}_{n_1(s')>0} \longrightarrow \underbrace{s\ldots s}_{n_2(s)>0} \longrightarrow \cdots \longrightarrow \underbrace{s'\ldots s'}_{n_{k-1}(s')>0} \longrightarrow \underbrace{s\ldots s}_{n_k(s)>0} \longrightarrow \underbrace{s'\ldots s'}_{n_k(s')\geq 0} \longrightarrow$$

Here, $n_0(s')$ is the number of times when s' appears in the trajectory before the first appearance of s', $n_i(s)$ is the number of times when s appears between two occurrences of s', and $n_k(s')$ the number of times when s' appears after the last appearance of s. From the strong Markov property, $n_1(s') \sim n_2(s') \sim \cdots \sim n_{k-1}(s')$ and $n_1(s) \sim n_2(s) \sim \cdots \sim n_k(s)$. Note also that these variables are all independent. Here k is a random variable following the geometric distribution $G(\alpha)$ where $\alpha = p(s \in \tau_{t:\infty} \mid s_t = s')$. Notice that when $n_k(s') > 0$, we have $n_k(s) \sim n_1(s)$ and $n_k(s') \sim n_1(s')$.

Using these two simplifications, one can write:

$$\begin{aligned}
\mathbb{E}_\tau\Big[\#(s' \to s) - \#(s \to s')\Big|k\Big] &\geq \mathbb{E}_\tau\Big[\#(s' \to s) - \#(s \to s')\Big|k, n_k(s') > 0\Big]\,,\\
&\geq \mathbb{E}_\tau\Big[n_0(s')\big[n_1(s) + (k-1)n_1(s) + n_k(s)\big] - n_1(s)\big[kn_1(s') + n_k(s')\big] +\\
&\quad n_k(s)\big[kn_1(s') - n_k(s')\big] - n_k(s')(k-1)n_1(s)\Big|k, n_k(s') > 0\Big]\,,\\
&\geq -k\mathbb{E}_\tau\Big[n_1(s)\Big|k\Big]\mathbb{E}_\tau\Big[n_k(s')\Big|k, n_k(s') > 0\Big] \text{ as in Lemma 2}\,,\\
&\geq -k\mathbb{E}_\tau(n_1(s))\mathbb{E}_\tau(n_1(s'))\,.
\end{aligned}$$

Likewise,

$$\begin{aligned}
\mathbb{E}_\tau\big[\#(s' \to s) + \#(s \to s')\,\big|\,k\big] &= \mathbb{E}_\tau\Big[k\,n_1(s)n_k(s') + k\,n_0(s')n_1(s) + k\,(k-1)n_1(s)n_1(s')\Big|k\Big]\,,\\
&= k\Big[\mathbb{E}_\tau\big[n_1(s)\big]\mathbb{E}_\tau\big[n_1(s')\big] + \mathbb{E}_\tau\big[n_1(s)\big]\mathbb{E}_\tau\big[n_0(s')\big]\Big] + k(k-1)\mathbb{E}_\tau\big[n_1(s)\big]\mathbb{E}_\tau\big[n_1(s')\big]\,.
\end{aligned}$$

Thus,

$$
\begin{aligned}
&\frac{\mathbb{E}_\tau\big[\#(s' \to s) - \#(s \to s')\big]}{\mathbb{E}_\tau\big[\#(s \to s') + \#(s' \to s)\big]} = \frac{\sum_{i=1}^{\infty} p(k=i)\mathbb{E}_\tau\left[\#(s' \to s) - \#(s \to s') \,|\, k=i\right]}{\sum_{i=1}^{\infty} p(k=i)\mathbb{E}_\tau\left[\#(s \to s') + \#(s' \to s) \,|\, k=i\right]}\,, \\
&\geq -\frac{\sum_{i=1}^{\infty} \alpha^{i-1}(1-\alpha)\, i\, \mathbb{E}_\tau\big[n_1(s)\big]\, \mathbb{E}_\tau\big[n_1(s')\big]}{\sum_{i=1}^{\infty} \alpha^{i-1}(1-\alpha)\Big[i\left(\mathbb{E}_\tau\big[n_1(s)\big]\mathbb{E}_\tau\big[n_1(s')\big] + \mathbb{E}_\tau\big[n_1(s)\big]\mathbb{E}_\tau\big[n_0(s')\big]\right) + i\,(i-1)\mathbb{E}_\tau\big[n_1(s)\big]\mathbb{E}_\tau\big[n_1(s')\big]\Big]}\,, \\
&\geq -\frac{\sum_{i=1}^{\infty} \alpha^{i-1}(1-\alpha) i \mathbb{E}_\tau\big[n_1(s)\big]\mathbb{E}_\tau\big[n_1(s')\big]}{\sum_{i=1}^{\infty} \alpha^{i-1}(1-\alpha)\Big[i\,\mathbb{E}_\tau\big[n_1(s)\big]\mathbb{E}_\tau\big[n_1(s')\big] + i\,(i-1)\mathbb{E}_\tau\big[n_1(s)\big]\mathbb{E}_\tau\big[n_1(s')\big]\Big]}\,, \\
&\geq -\frac{\sum_{i=1}^{\infty} \alpha^{i-1}(1-\alpha)\, i}{\sum_{i=1}^{\infty} \alpha^{i-1}(1-\alpha)\big[i + i\,(i-1)\big]}\,, \\
&\geq -\frac{\sum_{i=1}^{\infty} \alpha^{i-1}(1-\alpha)\, i}{\sum_{i=1}^{\infty} \alpha^{i-1}(1-\alpha)\, i^2}\,, \\
&\geq -\frac{\frac{1}{1-\alpha}}{\frac{1+\alpha}{(1-\alpha)^2}}\,, \\
&\geq -\frac{1-\alpha}{1+\alpha}\,.
\end{aligned}
$$

From Lemma 1,

$$
\begin{aligned}
\psi^\pi(s', s) &= \frac{1}{2}\left(1 + \frac{\mathbb{E}_\tau\big[\#(s' \to s) - \#(s \to s')\big]}{\mathbb{E}_{\tau\sim\pi}\big[\#(s \to s') + \#(s' \to s)\big]}\right)\,, \\
&\geq \frac{1}{2}\left(1 - \frac{1-\alpha}{1+\alpha}\right)\,, \\
&\geq \frac{\alpha}{1+\alpha}\,, \\
&\geq \frac{\alpha}{2} = \frac{\hat{\phi}_\pi(s, s')}{2}\,.
\end{aligned}
$$

As a quick summary, we divided our problem in 4 cases, and proved that in each case, for every pair of states s, s', we have $\psi^\pi(s', s) \geq \frac{\hat{\phi}_\pi(s,s')}{2}$.

To end the proof, we simply take the expectation over the distribution of the next states:

$$
\begin{aligned}
\mathbb{E}_{s'\sim P(s,a)}\psi_\pi(s', s) &\geq \frac{1}{2}\mathbb{E}_{s'\sim P(s,a)}\hat{\phi}_\pi(s, s')\,, \\
\bar{\phi}_\pi(s, a) &\geq \frac{\phi_\pi(s, a)}{2}\,.
\end{aligned}
$$

□

A.4 Proof of proposition 1

Proposition 1. *We suppose that we are given a state s, an action a such that a is reversible in K steps, a policy π and $\rho > 0$. Then, $\bar{\phi}_\pi(s, a) \geq \frac{\rho^K}{2}$, where A denotes the number of actions. Moreover, we have for all $K \in \mathbb{N}$: $\bar{\phi}_\pi(s, a) \geq \frac{\rho^K}{2}\phi_K(s, a)$.*

Proof. We first prove the second part of the proposition, which is more general. From Definition 1, and as the set of policies is closed, there is a policy π^* such that $\phi_K(s, a) = p_{\pi^*}(s \in \tau_{t+1:t+K+1} \mid$

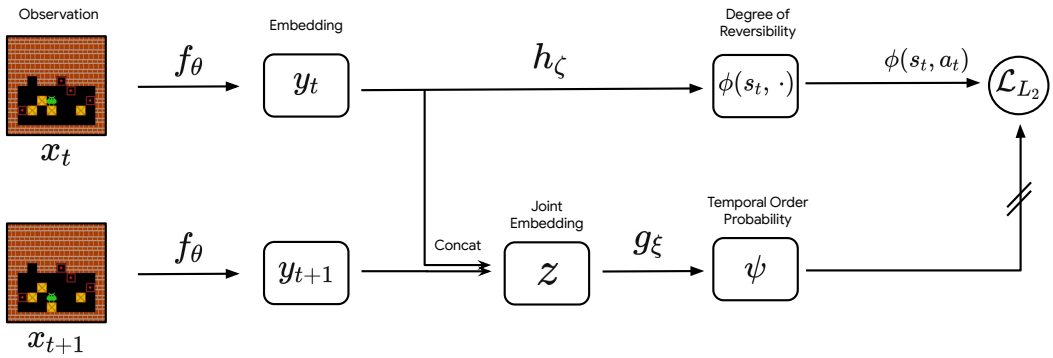


Figure 10: The training procedure for the reversibility estimator used in RAC.

$s_t = s, a_t = a)$. We begin by noticing that π has a probability at least equal to ρ to copy the policy π^* in every state.

It can be stated more formally:

$$\forall s \in S, \mathbb{E}_{a\sim\pi(s), a^*\sim\pi^*(s)}(\mathbb{1}_{a=a'}) = \sum_{a\in A} p_\pi(a \mid s) p_{\pi^*}(a \mid s) \geq \rho\Big(\sum_{a\in A} p_{\pi^*}(a \mid s)\Big) = \rho\,.$$

Then, we have:

$$\begin{aligned}
&\phi_{\pi,K}(s,a) = p_\pi(s \in \tau_{t+1:t+K+1} \mid s_t = s, a_t = a)\,,\\
&= \mathbb{E}_\pi\left[\mathbb{1}_{s\in\tau_{t+1:t+K+1}} \mid s_t = s, a_t = a\right],\\
&= \mathbb{E}_{s_{t+2},\ldots,s_{t+K+1}\sim\pi}\mathbb{E}_{s_{t+1}\sim p(s_t,a_t)}\left[\mathbb{1}_{s\in\tau_{t+1:t+K+1}} \mid s_t = s, a_t = a\right],\\
&= \mathbb{E}_{s_{t+3},\ldots,s_{t+K+1}\sim\pi}\mathbb{E}_{\substack{a_{t+1}\sim\pi(s_{t+1})\\ s_{t+2}\sim p(s_{t+1},a_{t+1})}}\mathbb{E}_{s_{t+1}\sim p(s_t,a_t)}\left[\mathbb{1}_{s\in\tau_{t+1:t+K+1}} \mid s_t = s, a_t = a\right],\\
&= \mathbb{E}_{s_{t+3},\ldots,s_{t+K+1}\sim\pi}\mathbb{E}_{\substack{a_{t+1}\sim\pi(s_{t+1}), a'_{t+1}\sim\pi^*(s_{t+1})\\ s_{t+2}\sim p(s_{t+1},a_{t+1})}}\mathbb{E}_{s_{t+1}\sim p(s_t,a_t)}\left[\mathbb{1}_{s\in\tau_{t+1:t+K+1}} \mid s_t = s, a_t = a\right],\\
&\geq \mathbb{E}_{s_{t+3},\ldots,s_{t+K+1}\sim\pi}\mathbb{E}_{\substack{a_{t+1}\sim\pi(s_{t+1}), a^*_{t+1}\sim\pi^*(s_{t+1})\\ s_{t+2}\sim p(s_{t+1},a_{t+1})\\ s_{t+1}\sim p(s_t,a_t)}}\left[\mathbb{1}_{s\in\tau_{t+1:t+K+1}} \mid s_t = s, a_t = a, a_{t+1} = a^*_{t+1}\right]\mathbb{1}_{a_{t+1}=a^*_{t+1}}\,,\\
&\geq \rho\,\mathbb{E}_{s_{t+3},\ldots,s_{t+K+1}\sim\pi}\mathbb{E}_{s_{t+1},s_{t+2}\sim\pi^*}\left[\mathbb{1}_{s\in\tau_{t+1:t+K+1}} \mid s_t = s, a_t = a\right], \text{and iterating the same process, },\\
&\geq \rho^K\mathbb{E}_{s_{t+1},s_{t+2},\ldots,s_{t+K+1}\sim\pi^*}\left[\mathbb{1}_{s\in\tau_{t+1:t+K+1}} \mid s_t = s, a_t = a\right],\\
&\geq \rho^K\phi_K(s,a)\,.
\end{aligned}$$

We can conclude using Theorem 2: $\bar{\phi}_\pi(s,a) \geq \frac{\phi_\pi(s,a)}{2} \geq \frac{\phi_{\pi,K}(s,a)}{2} \geq \frac{\rho^K}{2}\phi_K(s,a)$.

□

B Additional Details About Reversibility-Aware RL

B.1 Learning a reversibility estimator

We illustrate how the reversibility estimator is trained in Fig. 10. We remind the reader that it is a component that is specific to RAC. See Algorithm 2 for the detailed procedure of how to train it jointly with the standard precedence estimator and the RL agent.

Algorithm 1: RAE: Reversibility-Aware Exploration (online)

Initialize the agent weights Θ and number of RL updates per trajectory k;
Initialize the precedence classifier weights θ, ξ, window size w, threshold β and learning rate η;
Initialize the replay buffer $\mathcal{B}$;
for *each iteration* **do**
  `/* Collect interaction data and train the agent. */`
  Sample a trajectory $\tau = \{x_i, a_i, r_i\}_{i=1\ldots T}$ with the current policy;
  Incorporate irreversibility penalties $\tau' = \left\{x_i, a_i, r_i + r_\beta\big(\psi_{\theta,\xi}(x_i, x_{i+1})\big)\right\}_{i=1\ldots T}$;
  Store the trajectory in the replay buffer $\mathcal{B} \leftarrow \mathcal{B} \cup \tau$;
  Do k RL steps and update Θ;
  `/* Update the precedence classifier. */`
  for *each training step* **do**
    Sample a minibatch $\mathcal{D}_{batch}$ from $\mathcal{B}$;
    `/* Self-supervised precedence classification, loss in Eq.(9). */`
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{SSL}(\mathcal{D}_{batch})$;
    $\xi \leftarrow \xi - \eta \nabla_\xi \mathcal{L}_{SSL}(\mathcal{D}_{batch})$;
  end
end

Algorithm 2: RAC: Reversibility-Aware Control (online)

Initialize the agent weights Θ and number of RL updates per trajectory k;
Initialize the precedence classifier weights θ, ξ, window size w, threshold β and learning rate η;
Initialize the reversibility estimator weights ζ;
Initialize the replay buffer $\mathcal{B}$;
for *each iteration* **do**
  `/* Collect interaction data with the modified control policy and train the agent. */`
  Sample a trajectory τ under the rejection sampling policy $\bar{\pi}$ from eq.(8) ;
  Store the trajectory in the replay buffer $\mathcal{B} \leftarrow \mathcal{B} \cup \tau$;
  Do k RL steps and update Θ;
  `/* Update the precedence classifier. */`
  for *each training step* **do**
    Sample a minibatch $\mathcal{D}_{batch}$ from $\mathcal{B}$;
    `/* Self-supervised precedence classification, loss in Eq.(9). */`
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{SSL}(\mathcal{D}_{batch})$;
    $\xi \leftarrow \xi - \eta \nabla_\xi \mathcal{L}_{SSL}(\mathcal{D}_{batch})$;
  end
  `/* Update the reversibility estimator, loss in Eq.(10). */`
  for *each training step* **do**
    Sample a minibatch $\mathcal{D}_{batch}$ from $\mathcal{B}$;
    `/* Regression of the precedence classifier probabilities. */`
    $\zeta \leftarrow \zeta - \eta \nabla_\zeta \mathcal{L}_{L2}(\mathcal{D}_{batch}, \psi_{\theta,\xi})$;
  end
end

B.2 Pseudo-code for RAE and RAC

We give the pseudo-code for the online versions of RAE (Algorithm 1) and RAC (Algorithm 2). The rejection sampling policy $\bar{\pi}$ under approximate reversibility ϕ and threshold β is expressed as follows:

$$\bar{\pi}(a|x) = \begin{cases} 0 & \text{if } \phi(x,a) < \beta \\ \pi(a|x)/Z & \text{otherwise, with } Z = \sum_{a' \in \mathcal{A}} \mathbb{1}\{\phi(x,a') \geq \beta\}\pi(a'|x) \end{cases}. \tag{8}$$

This is equivalent, on average, to sampling from the policy π until an action that is reversible enough is found.

The loss we use to train the precedence estimator has the expression:

$$\mathcal{L}_{SSL}(\mathcal{D}_{batch}) = \frac{1}{|\mathcal{D}_{batch}|} \sum_{(x,x',y) \in \mathcal{D}_{batch}} -y \log\left(\psi_{\theta,\xi}(x,x')\right) + (1-y)\log\left(1 - \psi_{\theta,\xi}(x,x')\right), \tag{9}$$

where y is the binary result of the shuffle, with value 1 if observations were not shuffled (thus in the correct temporal order), and 0 otherwise. Pairs of observations (x, x') can be separated by up to w timesteps.

The loss we use to train the reversibility estimator (in RAC only) has the expression:

$$\mathcal{L}_{L2}(\mathcal{D}_{batch}, \psi_{\theta,\xi}) = \frac{1}{2\,|\mathcal{D}_{batch}|} \sum_{(x,a,x') \in \mathcal{D}_{batch}} \left(\psi_{\theta,\xi}(x,x') - \phi_{\zeta}(x,a)\right)^2, \tag{10}$$

where (x, a, x') are triples of state, action and next state sampled from the collected trajectories.

The offline versions of both RAE and RAC can be derived by separating each online algorithm into two parts: 1) training the precedence classifier (and the reversibility estimator for RAC), which is achieved by removing the data collection and RL steps and by using a fixed replay buffer; and 2) training the RL agent, which is the standard RL procedure augmented with modified rewards for RAE, and modified control for RAC, using the classifiers learned in the first part without fine-tuning.

C Experimental Details

C.1 Reward-Free Reinforcement Learning

Cartpole. The observation space is a tabular 4-dimensional vector: (cart position x, cart velocity $\dot{x}$, pole angle θ, pole velocity $\dot{\theta}$). The discrete action space consists of applying a force left or right. The episode terminates if the pole angle is more than ±12° ($|\theta| \leq 0.209$ radians), if the cart position is more than ±2.4, or after 200 time-steps. It is considered solved when the average return is greater than or equal to 195.0 over 100 consecutive trials.

Architecture and hyperparameters. The reversibility network inputs a pair of observations and produces an embedding by passing each one into 2 fully connected layers of size 64 followed by ReLU. The two embeddings are concatenated, and projected into a scalar followed by a sigmoid activation. We trained this network doing 1 gradient step every 500 time steps, using the Adam optimizer [25] and a learning rate of 0.01. We used batches of 128 samples, that we collected from a replay buffer of size 1 million. The penalization threshold β was fine-tuned over the set $[0.5, 0.6, 0.7, 0.8, 0.9]$ and eventually set to 0.7. We notice informally that it was an important parameter. A low threshold could lead to over penalizing the agent leading the agent to terminate the episode as soon as possible, whereas a high threshold could slow down the learning.

Regarding PPO, both the policy network and the value network are composed of two hidden layers of size 64. Training was done using Adam and a learning rate of 0.01. Other PPO hyperparameters were defaults in Raffin et al. [36], except that we add an entropy cost of 0.05.

C.2 Learning Reversible Policies

Environment. The environment consists of a 10 × 10 pixel grid. It contains an agent, represented by a single blue pixel, which can move in four directions: up, down, left, right. The pink pixel

represents the goal, green pixels grass and grey pixels a stone path. Stepping on grass spoils it and the corresponding pixel turns brown, as shown in Fig. 5b. A level terminates after getting to the goal, or after 120 timesteps. Upon reaching the goal, the agent receives a reward of +1, every other action being associated with 0 reward.

Architecture and hyperparameters. The reversibility network takes a pair of observations as input and produces an embedding by passing each observation through 3 identical convolutional layers of kernel size 3, with respectively 32, 64 and 64 channels. The convolutional outputs are flattened, linearly projected onto 64 dimensional vectors and concatenated. The resulting vector is projected into a scalar, which goes through a final sigmoid activation.

As done for Cartpole, we trained this network doing 1 gradient step every 500 time steps, using the Adam optimizer with a learning rate of 0.01. We used minibatches of 128 samples, that we collected from a replay buffer of size 1M. The penalization threshold β was set to 0.8, and the intrinsic reward was weighted by 0.1, such that the intrinsic reward was equal to $-0.1\,\mathbb{1}_{\psi(s_t,s_{t+1})>0.8}\,\psi(s_t,s_{t+1})$.

For PPO, both the policy network and the value network are composed of 3 convolutional layers of size 32, 64, 64. The output is flattened and passed through a hidden layer of size 512. Each layers are followed by a ReLU activation. Policy logits (size 4) and baseline function (size 1) were produced by a linear projection. Other PPO hyperparameters were defaults in Raffin et al. [36], except that we add an entropy cost of 0.05.

C.3 Sokoban

We use the Sokoban implementation from Schrader [41]. The environment is a 10x10 grid with a unique layout for each level. The agent receives a -0.1 reward at each timestep, a +1 reward when placing a box on a target, a -1 reward when removing a box from a target, and a +10 reward when completing a level. Observations are of size $(10, 10, 3)$. Episodes have a maximal length of 120, and terminate upon placing the last box on the remaining target. At the beginning of each episode, a level is sampled uniformly from a set of 1000 levels, which prevents agents from memorizing puzzle solutions. The set is obtained by applying random permutations to the positions of the boxes and the position of the agent, and is pre-computed for efficiency. All levels feature four boxes and targets.

We use the distributed IMPALA implementation from the Acme framework [24] as our baseline agent in these experiments. The architecture and hyperparameters were obtained by optimizing for sample-efficiency on a single held-out level. Specifically, the agent network consists of three 3x3 convolutional layers with 8, 16 and 16 filters and strides 2, 1, and 1 respectively; each followed by a ReLU nonlinearity except the last one. The outputs are flattened and fed to a 2-layer feed-forward network with 64 hidden units and ReLU nonlinearities. The policy and the value network share all previous layers, and each have a separate final one-layer feed-forward network with 64 hidden units and ReLU nonlinearities as well. Regarding agent hyperparameters, we use 64 actors running in parallel, a batch size of 256, an unroll length of 20, and a maximum gradient norm of 40. The coefficient of the loss on the value is 0.5, and that of the entropic regularization 0.01. We use the Adam optimizer with a learning rate of 0.0005, a momentum decay of 0 and a variance decay of 0.99.

The precedence estimator network is quite similar: it consists of two 3x3 convolutional layers with 8 filters each and strides 2 and 1 respectively; each followed by a ReLU nonlinearity except the last one. The outputs are flattened and fed to a 3-layer feed-forward network with 64 hidden units and ReLU nonlinearities, and a final layer with a single neuron. We use dropout in the feed-forward network, with a probability of 0.1. Precedence probabilities are obtained by applying the sigmoid function to the outputs of the last feed-forward layer. The precedence estimator is trained offline on 100k trajectories collected from a random agent. It is trained on a total of 20M pairs of observations sampled with a window of size 15, although we observed identical performance with larger sizes (up to 120, which is the maximal window size). We use the Adam optimizer with a learning rate of 0.0005, a momentum decay of 0.9, a variance decay of 0.999. We also use weight decay, with a coefficient of 0.0001. We use a threshold β of 0.9. We selected hyperparameters based on performance on validation data.

C.4 Reversibility-Aware Control in Cartpole+

Learning ψ. The model architecture is the same as described in Appendix C.1. The training is done offline using a buffer of 100k trajectories collected using a random policy. State pairs are fed to the classifier in batches of size 128, for a total of 3M pairs. We use the Adam optimizer with a learning rate of 0.01. We use a window w equal to 200, which is the maximum number of timesteps in our environment.

Learning ϕ. We use a shallow feed-forward network with a single hidden layer of size 64 followed by a ReLU activation. From the same buffer of trajectories used for ψ, we sample 100k transitions and feed them to ϕ in batches of size 128. As before, training is done using Adam and a learning rate of 0.01.

C.5 DQN and M-DQN in Cartpole+

We use the same architecture for DQN and M-DQN. The network is a feed-forward network composed of two hidden layers of size 512 followed by ReLU activation. In both cases, we update the online network every 4 timesteps, and the target network every 400 timesteps. We use a replay buffer of size 50k, and sample batches of size 128. We use the Adam optimizer with a learning rate of 0.001.

We train both algorithms for 2M timesteps. We run an evaluation episode every 1000 timesteps, and report the maximum performance encountered during the training process. We perform a grid search for the discount factor $\gamma \in [0.99, 0.999, 0.9997]$, and for M-DQN parameters $\alpha \in [0.7, 0.9, 0.99]$ and $\tau \in [0.008, 0.03, 0.1]$. The best performances were obtained for $\alpha = 0.9, \tau = 0.03$, and $\gamma = 0.99$.

C.6 Reversibility-Aware Control in Turf

Learning ψ. We use the same model architecture as in RAE (Appendix C.2), and the same offline training procedure that was used for Cartpole+ (Appendix C.4). The window w was set to 120, which is the maximum number of steps in Turf.

Learning ϕ. The architecture is similar to ψ, except for the last linear layers: the output of the convolutional layers is flattened and fed to a feed-forward network with one hidden layer of size 64 followed by a ReLU. Again, we used the exact same training procedure as in the case of Cartpole+ (Appendix C.4).

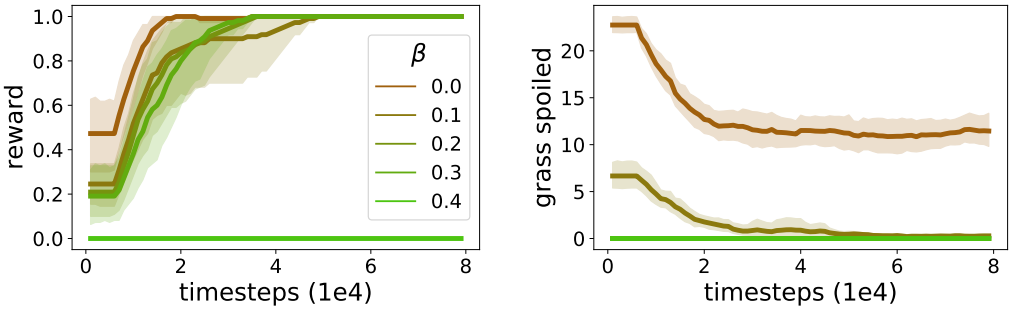
C.7 Safety and Performance Trade-off in Turf

We investigate the performance-to-safety trade-off induced by reversibility-awareness in Turf. In Fig. 11a, we see that the agent is not able to reach the goal when the threshold is greater than 0.4: with too high a threshold, every action leading to the goal could be rejected. We also see that it solves the task under lower threshold values, and that lowering β results in faster learning. On the other hand, Fig. 11b shows that achieving zero irreversible side-effects during the learning is only possible when β is greater than 0.2. In this setting, the optimal thresholds are thus between 0.2 and 0.3, allowing the agent to learn the new task while eradicating every side-effect.

This experiment gives some insights on how to tune β in new environments. It should be initialized at 0.5 and decreased progressively, until the desired agent behaviour is reached. This would ensure that the chosen threshold is the maximal threshold such that the environment can be solved, while having the greatest safety guarantees.

D Performance in Stochastic MDPs

To study how reversibility-awareness helps in stochastic MDPs, we use a 2D cliff-walking gridworld where stochasticity comes from the wind: additionally to its move, the agent is pushed towards the cliff with a fixed probability. The agent gets a +1 reward for each timestep it stays alive, with a maximum of 250 timesteps. A reversibility-aware agent with a well calibrated threshold should avoid most moves that push it towards the cliff. We use a 6x8 grid, with a maximum of 250 timesteps per episode, and report results averaged over 5000 runs. We provide two tables: Table 1 with the average



(a) (b)

Figure 11: **(a):** Reward learning curve for PPO+RAC and several thresholds β (average over 10 random seeds). A threshold of 0 means actions are never rejected, and corresponds to the standard PPO. **(b):** Number of irreversible side-effects (grass pixels stepped on). For β between 0.2 and 0.4, 0 side-effects are induced during the whole learning.

scores of a random policy and Table 2 with the average scores of a random policy equipped with RAC. Rows correspond to varying stochasticity and columns to varying threshold values.

Table 1: Scores for a random policy in the 2D cliff-walking gridworld, where p is the probability of being pushed by the wind. Higher is better.

p \ Threshold	0.	0.1	0.2	0.3	0.4
0.	57.5	57.7	61.2	58.2	57.7
0.1	29.8	28.8	29.5	30.2	29.6
0.2	18.6	18.5	19.3	18.9	18.8
0.3	13.4	13.3	13.9	13.6	13.4
0.4	10.5	10.7	10.4	10.2	10.2

Table 2: Scores for a random policy with RAC in the 2D cliff-walking gridworld, where p is the probability of being pushed by the wind. Higher is better.

p \ Threshold	0.	0.1	0.2	0.3	0.4
0.	59.1	250.0	250.0	250.0	250.0
0.1	29.2	56.0	56.3	80.2	248.5
0.2	18.7	26.7	29.2	85.8	238.6
0.3	13.2	16.8	19.6	77.6	250.0
0.4	10.4	12.5	24.9	152.2	250.0

We can notice that:

- RAC significantly improves performance (reaching maximum or near-maximum score),
- a well-tuned threshold value is crucial to get decent performance with RAC,
- the optimal threshold increases with the stochasticity of the environment (but seems to quickly converge).