Quantum Machine Intelligence (2020) 2:13
https://doi.org/10.1007/s42484-020-00023-9

RESEARCH ARTICLE

On the convergence of projective-simulation–based reinforcement learning in Markov decision processes

W. L. Boyajian[1] · J. Clausen[1] · L. M. Trenkwalder[1] · V. Dunjko[1,2] · H. J. Briegel[1,3]

Received: 7 November 2019 / Accepted: 2 July 2020 / Published online: 5 November 2020

Abstract

In recent years, the interest in leveraging quantum effects for enhancing machine learning tasks has significantly increased. Many algorithms speeding up supervised and unsupervised learning were established. The first framework in which ways to exploit quantum resources specifically for the broader context of reinforcement learning were found is projective simulation. Projective simulation presents an agent-based reinforcement learning approach designed in a manner which may support quantum walk-based speedups. Although classical variants of projective simulation have been benchmarked against common reinforcement learning algorithms, very few formal theoretical analyses have been provided for its performance in standard learning scenarios. In this paper, we provide a detailed formal discussion of the properties of this model. Specifically, we prove that one version of the projective simulation model, understood as a reinforcement learning approach, converges to optimal behavior in a large class of Markov decision processes. This proof shows that a physically inspired approach to reinforcement learning can guarantee to converge.

Keywords Reinforcement learning · Projective simulation · Convergence proof · Markov decision process · Physics-inspired artificial intelligence

1 Introduction

In the past decade, quantum information science established itself as a fruitful research field that leverages quantum effects to enhance communication and information processing tasks (Nielsen and Chuang 2000; Bennett and DiVincenzo 1995). The results and insights gained inspired further investigations which more recently contributed to the emergence of the field quantum machine learning (Schuld et al. 2014; Biamonte et al. 2016; Dunjko and Briegel 2018). The aim of this new field is twofold. On the one hand, machine learning methods are developed to further our understanding and control of physical systems and, on the other hand, quantum information processing is employed to enhance certain aspects of machine learning. A learning framework that features in both aspects of quantum machine learning is projective simulation (PS). In particular, PS can be seen as a platform for the design of autonomous (quantum) learning agents (Briegel and las Cuevas 2012).

The development of projective simulation is not motivated by the aim of designing ever-faster computer algorithms. Projective simulation is a tool for understanding various aspects of learning, where agents are viewed from the perspective of realizable entities such as robots or biological systems interacting with an unknown environment. In this embodied approach, the agent's perception is influenced by its sensors, its actions are limited by its physical capabilities, and its memory is altered by its interaction with an environment. The deliberation process of the agent can be described by a random walk process on the memory structure and it is their quantum counterpart, quantum random walks, that offers a direct route to the quantization of the deliberation and learning process. Thereby, PS not only allows us to study learning in the quantum domain, it also offers speedups in a variety of learning settings (Paparo et al. 2014; Sriarunothai et al. 2017).

Projective simulation can be used to solve reinforcement learning (RL) problems as well. Taken as a classical RL

✉ L. M. Trenkwalder
lea.trenkwalder@uibk.ac.at

[1] Institute for Theoretical Physics, University of Innsbruck, 6020 Innsbruck, Austria

[2] LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

[3] Department of Philosophy, University of Konstanz, 78457 Konstanz, Germany

approach, the PS has proven to be a successful tool for learning how to design quantum experiments (Melnikov et al. 2018). In Melnikov et al. (2018), PS was used to design experiments that generate high-dimensional multipartite entangled photonic states. The ability of PS to learn and adapt to an unknown environment was further used for optimizing and adapting quantum error correction codes (Nautrup et al. 2019). In a quite different context, PS is used to model complex skill acquisition in robotics (Hangl et al. 2016, 2020).

Although PS has been shown suitable for a number of applications, it is a fair question of just how well it does, compared with other models, or compared with theoretical optima. However, the empirical evaluation of a model through simulations and analytically proving the properties of the same model are fundamentally distinct matters. For example, in many applications, empirical convergence can be reached even if the conditions for theoretical convergence are not met. In any real-world application, such as learning to play the game of Go, convergence to optimal performance, even though it is theoretically feasible, is not reached due to the size of the state space, which for the game of Go consists of 10^{170} states. This, however, is not worrying in practice where the goal is to create a well-performing and fast algorithm without the goal of full convergence or theoretical guarantees. In numerical investigations of various textbook problems, it was shown that PS demonstrates a competitive performance with respect to standard RL methods (Melnikov et al. 2017, 2018; Mautner et al. 2015; Makmal et al. 2016). In this work, we complement those results by comparing PS with other RL approaches from a theoretical perspective. Specifically, we analyze if PS converges to an optimal solution. Other methods, like Q-learning and SARSA, have already been proven to converge in environments which are describable by Markov Decision Processes (MDPs) (Dayan and Sejnowski 1994; Singh et al. 2000; Jaakkola et al. 1994; Watkins and Dayan 1992). One should notice, however, that Q-learning and SARSA are methods equipped with update rules explicitly designed for such problems. PS, in contrast, was designed with a broader set of time-varying and partially observable learning environments in mind. For this reason, it is capable of solving tasks that a direct (naive) implementation of Q-learning and SARSA fails to learn as they are designed to obtain a time-independent optimal policy (Watkins and Dayan 1992; Sutton and Barto 2018); examples can be found in Mautner et al. (2015). Thus, it would be unlikely for a PS agent to exactly realize the same optimality with respect to the discounted infinite horizon reward figures of merit (for which Q-learning was designed) without any further adjustment to the model. Nonetheless, in this work, we analyze the properties of PS taken as a pure MDP solving RL algorithm. We show that a slightly modified PS variant recovers the notion of state-action values as a function of its internal parameters, while preserving the main characteristics that make PS stand out from other RL algorithms, such as the locality of the update rules. As we show, this new variant is suitable for episodic MDPs, and we can prove convergence to the optimal strategy for a range of solutions. In the process, we connect the modified PS model with the basic PS model, which allows us to partially explain and understand the empirical performance and successes of PS reported in previous experimental works.

This paper is organized as follows: We quickly recap the main concepts of RL theory[1] in Section 2 concerning MDPs that will be used by us during the rest of this paper before we present the PS model in Section 3. In Section 4, we begin by introducing the adaption to PS needed for the convergence proof, which will be followed by the convergence proof that is based on a well-known theorem in stochastic approximation theory. In the Appendix of the paper, we provide a detailed exposition of RL methods which introduces the necessary concepts for the analysis, with a broader perspective on RL theory in mind. Additionally, after discussing multiple variants of the PS update rules and their implications, we present an extensive investigation of the similarities and difference of PS to standard RL methods.

2 Markov decision processes

2.1 Policy and discounted return

In the RL framework, an *RL problem* is a general concept that encompasses the learning of an agent through the interaction with an environment with the goal of maximizing some precisely defined figure of merit such as a reward function. In a discrete-time framework, the agent–environment interaction can be modeled as follows. At every time step t, the agent perceives the environmental *state* S_t. Then, the agent chooses an *action* A_t to execute upon the environment. The environment completes the cycle by signaling to the agent a new state S_{t+1} and a reward R_{t+1}. The variables R_t, S_t, and A_t are, in general, random variables, where R_t can take values $r_t \in \mathbb{R}$, while S_t and A_t take values sampled from sets $\mathcal{S} = \{s_1, s_2, \dots\}$ and $\mathcal{A} = \{a_1, a_2, \dots\}$ respectively. For simplicity, we assume in the following that these two sets are finite and r_t is bounded for all time steps t.

A particularly important set of RL problems are those where the environment satisfies the Markovian property. These problems can be modeled by Markov Decision

[1] We will follow the notation introduced in Sutton and Barto (2018) closely.

Processes (MDPs). In an MDP, the probabilities of transitions and rewards are given by the set of probabilities:

$$p(s', r \mid s, a) \coloneqq \Pr\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\}. \tag{1}$$

At every time step, the agent chooses an action as the result of some internal function that takes as input the current state of the environment. Thus, formally, an agent maps states into actions, which is captured by the so-called *policy* of the agent. Mathematically, the policy (at a certain time step t) can be defined as the set of probabilities:

$$\pi(a \mid s) \coloneqq \Pr\{A_t = a \mid S_t = s\}. \tag{2}$$

The successive modification of these probabilities, $\pi = \pi_t$, through the experience with the environment constitutes the *learning* that the agent undergoes in order to achieve a goal. In an MDP, the notion of goal can be formalized by introducing a new random variable:

$$G_t(\gamma_{\text{dis}}) \coloneqq \sum_{k=0}^{\infty} \gamma_{\text{dis}}^k R_{t+k+1}, \tag{3}$$

called the *discounted return*, where $\gamma_{\text{dis}} \in [0, 1]$ is the *discount parameter*. The case with $\gamma_{\text{dis}} = 1$ is reserved for episodic tasks, where the agent–environment interaction naturally terminates at some finite time. The discounted return at some time step t consists of the sum of all rewards received after t, discounted by how far in the future they are received. The *solution* to the MDP is the policy that maximizes the expected return starting from any state s, called the *optimal policy*.

A particular set of RL problems we will consider in this work are the so-called *episodic* tasks. In these, the agent-environment interactions naturally break into episodes, e.g., an agent playing some card game, or trying to escape from a maze. Note that while in some episodic problems the objective could be to finish the episode with the fewest possible actions (e.g., escaping a maze), in general, the optimal solution is not necessarily related to ending the episode. A notion of episodic MDP can be easily incorporated into the theoretical formalism recalled above, by including a set $\mathcal{S}_T \subset \mathcal{S}$, of so-called terminal or absorbing states. These states are characterized by the fact that transitions from a terminal state lead back to the same state with unit probability and zero reward. In episodic MDPs, the goal for the agent is to maximize the expected discounted return per episode.

It should be noted that the concept of absorbing states is a theoretical construct introduced to include the concept of episodic and non-episodic MDPs into a single formalism. In a practical implementation, however, after reaching a terminal state, an agent would be reset to some *initial state*, which could be a predefined state or chosen at random for instance. While such a choice could have an impact on learning rates, it is irrelevant regarding the optimal policy. For this reason, in the following, we do not make any assumption about the choice of the initial states. We will assume, however, that the environment signals the finalization of the episode to the agent.

2.2 Value functions and optimal policy

The concept of an optimal policy is closely intertwined with that of *value functions*. The value $v_\pi s$ of a state $s \in \mathcal{S}$ under a certain policy π is defined as the expected return after state s is visited; i.e., it is the value:

$$v_\pi(s) \coloneqq \mathrm{E}_\pi \{G_t \mid S_t = s\}. \tag{4}$$

It has been proven for finite MDPs that there exists at least one policy, called the *optimal policy* π^*, which maximizes over the space of policies $v_\pi(s)\ \forall s$ simultaneously, i.e.:

$$v_*(s) = \max_\pi \{v_\pi(s)\}, \forall s \in \mathcal{S}, \tag{5}$$

where v_* denotes the value functions associated to the optimal policy.

Value functions can also be defined for state-action pairs. The so-called Q-value of a pair (s, a), for a certain policy π, is defined as the expected return received by the agent following the execution of action a while in state s, and sticking to the policy π afterwards. The Q-values of the optimal policy, or *optimal Q-values*, can be written in terms of the optimal state value functions as:

$$q_*(s, a) = r(s, a) + \gamma_{\text{dis}} \mathrm{E}\{v_*(S_{t+1}) | S_t = s, A_t = a\}, \tag{6}$$

where

$$r(s, a) = \mathrm{E}\{R_{t+1} \mid S_t = s, A_t = a\}. \tag{7}$$

The relevance of Q-values is evidenced by noting that given the set of all q_* values, an optimal policy can be derived straightforwardly as:

$$\pi^*(s) = \arg\max_{a'} \left\{q_*(s, a')\right\}. \tag{8}$$

(Note the notational difference in the arguments to distinguish between the stochastic policy (2), which returns a probability, and the deterministic policy (8), which returns an action.) For this reason, standard RL methods achieve an optimal policy in an indirect way, as a function of the internal parameters of the model, which are those which are updated through the learning of the model, and which in the limit converge to the q_* values. A similar procedure will be used by us in Section 4, where we discuss the convergence of PS to the optimal policy of MDPs.

2.3 Q-Learning and SARSA

Q-Learning and SARSA are two prominent algorithms that capture an essential idea of RL: online learning in an

unknown environment. They are particularly designed to solve Markovian environments and their prominence can in part be ascribed to the theoretical results that prove their convergence in MDPs. In both algorithms, learning is achieved by estimating the *action value function* $q_\pi(s,a)$ for every state action pair for a given policy π. This estimate is described as the Q-value which is assigned to each state-action pair. The update of the Q-value is given by:

$$Q_{t+1}(s_t,a_t) = (1-\alpha)Q_t(s_t,a_t) + \alpha(R_{t+1} + \gamma_{\text{dis}} f(Q_t(s_{t+1},a_{t+1})). \tag{9}$$

The learning rate α describes how fast a new estimate of the Q-value overwrites the previous estimate. In SARSA, the function f is the identity, so that the Q-value is not only updated by the reward R_{t+1} but also with the Q-value of the next state-action pair along the policy π. Thus, SARSA is an on-policy algorithm, as described in Appendix A. In Q-learning, on the other hand, the function $f = \max_{a_{t+1}}$ takes the maximal Q-value of the next state. This algorithm is an off-policy algorithm due to sampling of the next action independently from the update of the Q-values.

3 Projective simulation

Projective simulation (PS) is a physically inspired framework for artificial intelligence introduced in Briegel and las Cuevas (2012). The core of the model is a particular kind of memory called *episodic and compositional memory* (ECM) composed of a stochastic network of interconnected units, called clips (cf. Fig. 2 in Briegel and las Cuevas (2012)). Clips represent either percepts or actions experienced in the past, or in more general versions of the model, combinations of those. The architecture of ECM, representing deliberation as a random walk in a network of clips, together with the possibility of combining clips and thus creating structures within the network, allows for modeling incipient forms of *creativity* (Briegel 2012; Hangl et al. 2020). Additionally, the deliberation process leading from percepts to actions has a physical interpretation in PS. Visiting any environmental state activates a corresponding percept clip in the ECM. This activation can be interpreted as an excitation, which then propagates stochastically through the network in the form of a random walk. The underlying dynamics have the potential to be implementable by real physical processes, thus relating the model to embodied agents including systems which exploit quantum effects, as has been explored in Dunjko et al. (2016) and Clausen and Briegel (2018).

PS can be used as an RL approach, where the action, the percept, and the reward are used to update the ECM structure. In general, the PS framework enables to leverage complex graph structures to enhance learning. For example, generalization can be implemented through manipulation of the ECM topology so that the RL agent is capable of learning in scenarios it would otherwise fail to learn (Melnikov et al. 2017). However, this generalization mechanism is not necessary for solving MDP environments.

Before we discuss the ECM for solving MDPs in detail, we need to emphasize the difference between the state of the environment and the percept the agent receives. In an algorithm specifically designed to solve MDPs, the state contains sufficient information of the environment such that the corresponding transition function fulfills the Markov property. We will refer to this type of state as Markov state. This assumption on the state space can generally not be made in most realistic learning scenarios but it can be generalized to partially observable MDPs where the Markovian dynamics are hidden. In a partially observable environment, the input of the learning algorithm is an observation that is linked to a Markov state via a, from the perspective of the algorithm, unknown probability distribution.

A percept, as introduced in the PS model, further generalizes the concept of such an observation. Here, the percept does not necessarily have to be connected to an underlying Markov state contrary to the observation in partially observable MDPs. This distinction might not seem necessary for learning in a classical environment but plays a significant role when one considers quantum systems that cannot be described with hidden variable models. In this work, since we focus on MDPs, we will equate the percepts an agent receives and the state of the MDP. In the following, both are denoted by s. Furthermore, we will not emphasize the difference between the percept and its corresponding percept clip, assuming there is a one-to-one correspondence between percept and percept clip. The same holds for the actions and their corresponding action clips.

The ECM structure used to solve MDPs consists of one layer of percept clips that is fully connected with a layer of action clips. Each edge represents a state action pair (s,a) which is assigned a real-valued weight (or hopping value) $h = h(s,a)$ and a non-negative glow value $g = g(s,a)$. While the weight h determines the probability of transition between a percept clip and an action clip, the glow variable g measures how 'susceptible' this weight h is to future rewards from the environment. In Eq. 10, h^{eq} is an (arbitrarily given) equilibrium value, and λ_{t+1} is the reward received immediately after action a_t, in accordance with the time-indexing conventions in Sutton and Barto (2018) as shown in Fig. 1.

A variety of different update rules are discussed in Appendix D and compared with other RL methods in Appendix E. In the following, we will focus on the standard update used in Briegel and las Cuevas (2012), Mautner et al.

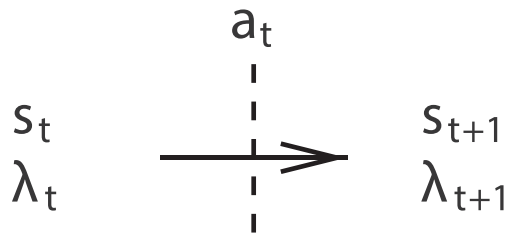

Fig. 1 Transition from time step t to $t+1$, ($t=0, 1, 2, \ldots$), via the agent's decision a_t, where s and λ denote environment state and reward ($\lambda_0=0$), respectively (adapted from Sutton and Barto (2018))

(2015), and Melnikov et al. (2018). The update rules for the h-value and the glow value are given by:

$$h_{t+1}(s,a) = h_t(s,a) - \gamma(h_t(s,a) - h^{\text{eq}}) + g_t(s,a)\lambda_{t+1} \tag{10}$$

$$g_t(s,a) = (1-\delta_{(s,a),(s_t,a_t)})(1-\eta)g_{t-1}(s,a) + \delta_{(s,a),(s_t,a_t)} \tag{11}$$

The update of the h-value consists, in the language used in the context of Master equations, of a gain and a loss term. The parameter for the loss term is called damping parameter and is denoted by $\gamma \in [0, 1]$. The parameter for the gain term is called glow parameter and is denoted by $\eta \in [0, 1]$. In particular, $\eta = 1$ recovers the original PS as introduced in Briegel and las Cuevas (2012). Finally, $\delta_t := \delta_{s,s_t}\delta_{a,a_t} = \delta_{(s,a),(s_t,a_t)}$ denotes the Kronecker delta symbol, which becomes 1 if the respective (s, a)-pair is visited at cycle t, and is otherwise 0. The agent's policy is defined as the set of all conditional probabilities (i.e., transition probabilities in the ECM clip network):

$$p_{ij} = p(a_j|s_i) = \frac{\Pi(h_{ij})}{\kappa_i}, \quad \kappa_i = \sum_j \Pi(h_{ij}), \tag{12}$$

of selecting action a_j when in state s_i and is here described in terms of some given function Π. Examples of Π which have been used or discussed in the context of PS are an identity function (Briegel and las Cuevas 2012; Mautner et al. 2015):

$$\Pi(x) = x, \tag{13}$$

if x is non-negative, and an exponential function leading to the well-known softmax policy (Melnikov et al. 2018) if a normalization factor is added:

$$\Pi(x) = \mathrm{e}^{\beta x}, \tag{14}$$

where $\beta \geq 0$ is a real-valued parameter.

4 Convergence of PS in episodic MDPs

In previous works, it has been shown numerically that the basic version of a PS agent is capable of learning the optimal strategy in a variety of textbook RL problems. The PS model with standard update rules, however, does not necessarily converge in all MDP settings. This version of the PS is thoroughly analyzed in Appendix D and Appendix E. As recalled in Section 2, in MDPs, optimality can be defined in terms of the optimal policy. In this section, we present a modified version of the PS that has been designed exclusively to tackle this kind of problem. We consider arbitrary episodic MDPs, and derive an analytical proof of convergence. In this version, the policy function depends on the normalized $\tilde{h}$ values, which, as we show later, behave similarly as state-action values, and in fact, in episodic MDPs, they converge to the optimal q_* values for a range of discount parameters.

4.1 Projective simulation for solving MDPs

In the following, we introduce a new variant of PS aimed at solving episodic MDPs. In those problems, there is a well-defined notion of optimality, given by the optimal policy. As described above, the basic PS constitutes a direct policy method (see also Appendix A). Finding the optimal policy of an MDP by policy exploration seems a rather difficult task. However, as other methods have proven, finding the optimal q_* values can be done with relatively simple algorithms, and the optimal policy can be derived from the q_* values in a straightforward manner. Motivated by this, we add a new local variable to the ECM network in order to recover a notion of state-action values while maintaining the locality of the model.

For this version, we consider "first-visit" glow, defined as follows.[2] The glow of any given edge is set to 1 whenever that edge is visited for the first time during an episode and in any other circumstance it is damped by a factor $(1-\eta)$, even if the same edge is visited again during the same episode. In addition, the entire glow matrix is reset to zero at the end of an episode. We thus write the updates as:

$$h_{t+1}(s,a) = h_t(s,a) + \lambda_{t+1}g_t(s,a) \tag{15}$$

$$g_t(s,a) = (1-\eta)g_{t-1}(s,a) + \delta_{(s,a),(s,a)_{\text{first-visit}}} \tag{16}$$

$$N_{t+1}(s,a) = N_t(s,a) + \delta_{(s,a),(s,a)_{\text{first-visit}}} \tag{17}$$

Here, the update for h is the same as in Eq. 10, but given that the MDPs are time-independent environments, γ has been set to 0. We add a matrix N to the standard PS, which counts the number of episodes during which each entry of h has been updated. The idea behind these updates is that the ratios:

$$\tilde{h}_t(s,a) := \frac{h_t(s,a)}{N+1} \tag{18}$$

[2] We can assume without loss of generality that the environment signals the finalization of the episode. Thus, the first visits to an edge can be determined locally. Moreover, the same signal can be used to reset the glow values locally.

resemble state-action values. To gain some intuition about this, note that h-values associated to visited edges will accumulate during a single episode a sum of rewards of the form:

$$\lambda_t + (1-\eta)\lambda_{t+1} + (1-\eta)^2\lambda_{t+2} + \ldots, \tag{19}$$

which gets truncated at the time step the episode ends. Hence, the normalized $\tilde{h}$ values become averages of sampled discounted rewards (see Appendix E.1). Later, we show that paired with the right policy and glow coefficient the $\tilde{h}$ values converge to the optimal q_* values.

Instead of considering a policy function of the h-values as in Eq. 12, here we will consider a policy function given by

$$p_{i,j} = \frac{\Pi(\tilde{h}_{i,j})}{c_i}, \quad c_i = \sum_j \Pi(\tilde{h}_{i,j}), \tag{20}$$

for a certain function $\Pi(\cdot)$. Given that the $\tilde{h}$-values are, in general, not diverging with time (in fact they are bounded in the case of bounded rewards) a linear function, as in Eq. 13, would fail to converge to a deterministic policy. A solution for that is to use a softmax function as in Eq. 14, where the free coefficient β is made time dependent. By letting β diverge with time, the policy can become deterministic in the limit.

Similarly to Monte Carlo methods, which may be equipped with a variety of update rules, giving rise to first-visit or many-visit Monte Carlo methods, the choice of the glow update rule is to some extent arbitrary but may depend on the physical implementation of PS and the ECM. For example, instead of Eq. 15, one could use the accumulating glow update, given in Eq. 53. In that case, one simply needs to change the update rule of N, given in Eq. 17 in such a way that every visit of the edge is counted, instead of only first visits. Intuitively, both pairs of update rules have similar effects, in the sense that in both cases $\tilde{h}(s,a)$ equals an average of sampled discounted returns starting from the time a state-action pair (s,a) was visited. However, while for first-visit glow, we were able to prove convergence, that is not the case for accumulating glow. Therefore, when referring to this version of PS in the following, we assume update rules given by Eqs. 15–17.

4.2 Convergence to the optimal policy

The convergence of $\tilde{h}$ values to q_* values can be proven by a standard approach used in the literature to prove, for example, the convergence of RL methods like Q-learning and SARSA, or prediction methods like TD(λ). In the remainder of the paper, we will use interchangeably the boldface notation $\boldsymbol{e}$ to denote a state-action pair as well as the explicit notation (s,a) whenever convenience dictates. Denoting by $\tilde{h}_m(\boldsymbol{e})$ the $\tilde{h}$-value of edge $\boldsymbol{e}$ at the end of episode m, we define the family of random variables $\Delta_m(\boldsymbol{e}) \coloneqq \tilde{h}_m(\boldsymbol{e}) - q_*(\boldsymbol{e})$. We want to show that in the limit of large m, $\Delta_m(\boldsymbol{e})$ converges to zero for all $\boldsymbol{e}$. Moreover, it is desirable that such convergence occurs in a strong sense, i.e., with probability 1. We show that by following the standard approach of constructing an update rule for $\Delta_m(\boldsymbol{e})$ which satisfies the conditions of the following theorem [3]

Theorem 1 *A random iterative process* $\Delta_{m+1}(\boldsymbol{x}) = [1-\alpha_m(\boldsymbol{x})]\,\Delta_m(\boldsymbol{x}) + \alpha_m(\boldsymbol{x})F_m(\boldsymbol{x})$, $\boldsymbol{x} \in X$ *converges to zero with probability one (w.p.1) if the following properties hold:*

1. *the set of possible states X is finite.*
2. $0 \le \alpha_m(\boldsymbol{x}) \le 1$, $\sum_m \alpha_m(\boldsymbol{x}) = \infty$, $\sum_m \alpha_m^2(\boldsymbol{x}) < \infty$ *w.p.1, where the probability is over the learning rates $\alpha_m(\boldsymbol{x})$.*
3. $\|\mathrm{E}\{F_m(\cdot)|P_m\}\|_{\mathrm{W}} \le \kappa\|\Delta_m(\cdot)\|_{\mathrm{W}} + c_m$, *where* $\kappa \in [0,1)$ *and c_m converges to zero w.p.1*
4. $\mathrm{Var}\{F_m(\boldsymbol{x})|P_m\} \le K(1+\|\Delta_m(\cdot)\|_{\mathrm{W}})^2$, *where K is some constant.*

Here P_m is the past of the process at step m, and the notation $\|\cdot\|$ denotes some fixed weighted maximum norm.

In addition to $\Delta_m(\boldsymbol{e})$ meeting the conditions of the theorem, the policy function must also satisfy two specific requirements. First of all, it must be *greedy* with respect to the $\tilde{h}$-values (at least in the limit of m to infinity). In that way, provided that the $\tilde{h}$-values converge to the optimal q_* values, the policy becomes automatically an optimal policy. Additionally, to guarantee that all Δ_m keep being periodically updated, the policy must guarantee infinite exploration. A policy that satisfies these two properties is called GLIE (Singh et al. 2000), standing for Greedy in the Limit and Infinite Exploration. Adapting the results from Singh et al. (2000) for PS and episodic environments, we can show (see Appendix G) that a softmax policy function defined by:

$$\pi_m(a|s,\tilde{\boldsymbol{h}}_m) = \frac{\exp\left[\beta_m \tilde{h}_m(s,a)\right]}{\sum_{a'\in\mathcal{A}} \exp\left[\beta_m \tilde{h}_m(s,a')\right]} \tag{21}$$

is GLIE, provided that $\beta_m \to_{m\to\infty} \infty$ and $\beta_m \le C\ln(m)$, where C is a constant depending on η and $|\mathcal{S}|$. While the first condition on β_m guarantees that the policy is greedy in the limit, the second one guarantees that the agent will keep exploring all state-action pairs infinitely

[3] This theorem is a known result in the field of stochastic approximation. While the first version of the theorem was presented in Dvoretzky et al. (1956), it can be found in many forms in the literature. The version presented here, where the contraction property has been relaxed by allowing a noise that tends to zero, has been presented in Singh et al. (2000).

often. In this particular example, we have considered β_m to depend exclusively on the episode index. By doing so, the policy remains local, because β_m can be updated using exclusively the signal of the environment indicating the finalization of the episode. Note however that the choice of the policy function, as far as it is GLIE, has no impact on the convergence proof. We are now in a position to state our main result about the convergence of PS-agents in the form of the following theorem.

Theorem 2 *For any finite episodic MDP with a discount factor of γ_{dis}, the policy resulting from the new updates converges with probability one to the optimal policy, provided that:*

1. *Rewards are bounded,*
2. $0 \leq \gamma_{\text{dis}} \leq 1/3$ *, where* $\gamma_{\text{dis}} = 1 - \eta$*,*
3. *The policy is a GLIE function of the $\tilde{h}$-values.*

Note that we have restricted the range of values γ_{dis} can take. The reason for that is related to the way the h-values are updated in PS. In Q-learning and SARSA, where the γ_{dis} parameter of the MDP is directly included in the algorithm, every time an action is taken its corresponding Q-value is updated by a sum of a single reward and a discounted bootstrapping term. Given that the PS updates do not use bootstrapping, that term is "replaced" by a discounted sum of rewards. Due to this difference, the contraction property (Condition 3 in Theorem 1) is not so straightforward to prove forcing us to consider smaller values of γ_{dis}. However, this condition on the γ_{dis} parameter is not a fundamental restriction of the PS model, but merely a result of how convergence is proven in this work.

4.3 Environments without terminal states

In Theorem 2, we have considered exclusively episodic MDPs. However, it is still possible for these environments to have an optimal policy which does not drive the agent to any terminal states. This observation suggests that the scope of problems solvable by PS can be further extended to a subset of non-episodic MDPs.

Given any non-episodic MDP, one can construct an episodic MDP from it by adding one single terminal state s_T and one single transition leading to it with non-zero probability, i.e., by defining $p_T = \Pr(s_T|s,a) \neq 0$ for some arbitrary pair (s,a). Thus, while the original non-episodic MDP falls outside the scope of Theorem 2, PS could be used to tackle the non-episodic MDP. Anyway, in general, these two problems might have different solutions, i.e., different optimal policies. However, given that both the pair (s,a) for which $p_T \neq 0$ and the value of p_T are arbitrary, by properly choosing them, the difference between the two optimal policies could become negligible or non-existent. That could be done easily having some useful knowledge about the solution of the original MDP. Consider for instance a grid world, where multiple rewards are placed randomly around some central area of grid cells. Even without knowing the exact optimal policy, one can correctly guess that there will be an optimal cyclic path about the center of the world yielding the maximum expected discounted return. Hence, adding a terminal state in some remote corner of the world would very likely leave the optimal policy unchanged.

4.4 Proof of Theorem 2

In this section, we discuss the core of the proof of Theorem 2, leaving for Appendix F the most involved calculations. Given that the policy is a greedy-in-the-limit-function of the $\tilde{h}_m$ values, the proof of Theorem 2 follows if we show that:

$$\Delta_m(\boldsymbol{e}) \coloneqq \tilde{h}_m(\boldsymbol{e}) - q_*(\boldsymbol{e}) \tag{22}$$

converges to 0 with probability 1. In order to do so, we show that $\Delta_m(\boldsymbol{e})$ obeys an update rule of the form given in Theorem 1 and the four conditions of the theorem are satisfied.

We begin by deriving an update rule for the h-values between episodes. In the case where an edge $\boldsymbol{e}$ is not visited during the m-th episode, its corresponding h-value is left unchanged, i.e., $h_m(\boldsymbol{e}) = h_{m-1}(\boldsymbol{e})$. Otherwise, due to the decreasing value of the glow during the episode, in the m-th episode, the $h(\boldsymbol{e})$ value will accumulate a discounted sum of rewards given by:

$$D_m(\boldsymbol{e}) = \sum_{t=t_m(\boldsymbol{e})}^{T_m} \bar{\eta}^{t-t_m(\boldsymbol{e})} \lambda_t, \tag{23}$$

where $t_m(\boldsymbol{e})$ and T_m are the times at which the first visit to $\boldsymbol{e}$ during episode m occurred and at which the episode finished, respectively, and $\bar{\eta} = 1 - \eta$. Therefore, in general $h_m(\boldsymbol{e}) = h_{m-1}(\boldsymbol{e}) + \chi_m(\boldsymbol{e}) D_m(\boldsymbol{e})$, where $\chi_m(\boldsymbol{e})$ is given by

$$\chi_m(\boldsymbol{e}) = \begin{cases} 1 & \text{if } \boldsymbol{e} \text{ is visited during the } m\text{-th episode,} \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

We denote, respectively, by $N_m(\boldsymbol{e})$ and $\tilde{h}_m(\boldsymbol{e})$ the N-value and $\tilde{h}$-value associated to edge $\boldsymbol{e}$ at the end of episode m. Thus, we have that $\tilde{h}_m(\boldsymbol{e}) = h_m(\boldsymbol{e})/[N_m(\boldsymbol{e}) + 1]$ and it obeys an update rule of the form:

$$\tilde{h}_m(\boldsymbol{e}) = \frac{1}{N_m(\boldsymbol{e}) + 1} \Big\{ \big[N_{m-1}(\boldsymbol{e}) + 1 \big] \tilde{h}_{m-1}(\boldsymbol{e}) + \chi_m(\boldsymbol{e}) D_m(\boldsymbol{e}) \Big\}. \tag{25}$$

Noting that the variables $N_m(\boldsymbol{e})$ can be written in terms of $\chi_m(\boldsymbol{e})$ as the sum $N_m(\boldsymbol{e}) = \sum_{j=1}^{m} \chi_j(\boldsymbol{e})$, it follows from Eq. 25 that the variable $\Delta_m(\boldsymbol{e})$ given in Eq. 22 satisfies the recursive relation:

$$\Delta_m(\boldsymbol{e}) = [1 - \alpha_m(\boldsymbol{e})]\, \Delta_{m-1}(\boldsymbol{e}) + \alpha_m(\boldsymbol{e}) F_m(\boldsymbol{e}), \tag{26}$$

where the ratios

$$\alpha_m(\boldsymbol{e}) \coloneqq \frac{\chi_m(\boldsymbol{e})}{N_m(\boldsymbol{e}) + 1}, \tag{27}$$

play the role of learning rates, and $F_m(\boldsymbol{e})$ is defined as:

$$F_m(\boldsymbol{e}) \coloneqq \chi_m(\boldsymbol{e})\, (D_m(\boldsymbol{e}) - q_*(\boldsymbol{e}))\,. \tag{28}$$

The update rule in Eq. 26 is exactly of the form given in Theorem 1. Therefore, we are left with showing that $\alpha_m(\boldsymbol{e})$ satisfies Condition 2 in Theorem 1, and $F_m(\boldsymbol{e})$ satisfies Conditions 3 and 4. Below, we describe the general procedure to prove that, while most of the details can be found in Appendix F.

The fact that $\alpha_m(\boldsymbol{e})$ satisfies Condition 2 in Theorem 1 follows from noting that $\sum_m \alpha_m(\boldsymbol{e}) = \sum_n 1/n$ and $\sum_n \alpha_m^2(\boldsymbol{e}) = \sum_n 1/n^2$, which are, respectively, a divergent and a convergent series. Regarding Condition 3, note that by tweaking the free glow parameter in such a way that $\bar{\eta} = \gamma_{\text{dis}}$, the variable $D_m(\boldsymbol{e})$ becomes a truncated sample of the discounted return $G(\boldsymbol{e}, \gamma_{\text{dis}})$ given in Eq. 3. Thus, $\tilde{h}$ values undergo a similar update to that found in SARSA, with the difference that instead of a bootstrapping term an actual sample of rewards is used. Due to these similarities, we can use the same techniques used in the proof of convergence of RL methods (Jaakkola et al. 1994; Singh et al. 2000) and show that:

$$\|\mathrm{E}\{F_m(\cdot)|P_m\}\|_{\mathrm{W}} \leq f(\gamma_{\text{dis}}) \|\Delta_m(\cdot)\|_{\mathrm{W}} + c_m, \tag{29}$$

where c_m converges to 0 w.p.1 and $f(\gamma_{\text{dis}}) = \frac{2\gamma_{\text{dis}}}{1-\gamma_{\text{dis}}}$. This equation satisfies Condition 3 in Theorem 1 as far as $f(\gamma_{\text{dis}}) < 1$, which occurs for $\gamma_{\text{dis}} < 1/3$.

Finally, Condition 4 in Theorem 1 follows from the fact that rewards are bounded. This implies that $\tilde{h}$-values and, in turn, the variance of $F_m(\boldsymbol{e})$ are bounded as well. This concludes the proof of Theorem 2.

5 Conclusion

In this work, we studied the convergence of a variant of PS applied to episodic MDPs. Given that MDPs have a clear definition of a goal, characterized by the optimal policy, we took the approach of adapting the PS model to deal with this kind of problem specifically. The first visit glow version of PS presented in this work internally recovers a certain notion of state-action values, while preserving the locality of the parameter updates, crucial to guarantee a physical implementation of the model by simple means. We have shown that with this model a PS agent achieves optimal behavior in episodic MDPs, for a range of discount parameters. This proof and the theoretical analysis of the PS update rules shed light on how PS, or, more precisely, its policy, behaves in a general RL problem.

The PS updates that alter the h-values at every time step asynchronously pose a particular challenge for proving convergence. To deal with that, we analyzed the subsequence of internal parameters at the times when episodes end, thus recovering a synchronous update. We could then apply techniques from stochastic approximation theory to prove that the internal parameters of PS converge to the optimal q values, similarly as in the convergence proofs of other RL methods.

We have also chosen a specific glow update rule, which we have called first-visit glow. While other glow updates, like accumulating or replacing glow, show the same behavior at an intuitive level, trying to prove the convergence with those updates has proven to be more cumbersome. Therefore, from a practical point of view, several glow mechanisms could be potentially utilized, but convergence in the limit is, at the moment, only guaranteed for first-visit glow.

Although only episodic MDPs fall within the scope of our theorem, no constraints are imposed on the nature of the optimal policy. Hence, episodic problems where the optimal policy completely avoids terminal states (i.e., the probability that an agent reaches a terminal state by following that policy is strictly zero) can also be considered. Furthermore, the agent could be equipped with any policy, as far as the GLIE condition is satisfied. In this paper, we provided a particular example of a GLIE policy function, in the form of a softmax function with a global parameter, which depends exclusively on the episode index. In this particular case, the policy is compatible with local updates, in the sense that the probabilities to take an action given a state can be computed locally.

Funding information Open access funding provided by Austrian Science Fund (FWF). This work was supported, in part, by the Austrian Science Fund through the projects SFB FoQus F4212, SFB BeyondC F7102, and DK ALM W1259, and in part by the Dutch Research Council (NWO/OCW), through the Quantum Software Consortium programme (project number 024.003.037)

Appendix A: A review of RL methods

The following section is meant as a concise overview of standard RL methods, which we distilled and adapted from (Sutton and Barto 2018), to provide the necessary background before which the PS will we be discussed in Sections D (ref to Section D) and E (ref to Section E). For details we refer the reader to Ref. (Sutton and Barto 2018).

Among the model-free and gradient-based approaches, we can broadly distinguish between value function-based methods which are gradient-descent with respect to a so-called temporal difference (TD) error and direct policy methods which are gradient-ascent with respect to the expected return as shown in Fig. 2.

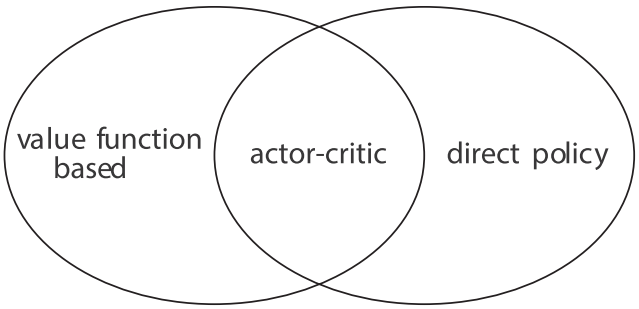

Fig. 2 Some types of gradient-based solution methods for RL-problems. Value function-based methods are gradient-descent with respect to a TD error, whereas direct policy methods are gradient-ascent with respect to the expected return. Actor-critic methods are depicted as their intersection since they combine both approaches. Being understood as "parametric" methods, this figure corresponds to the left branch of Fig. 3 in Clausen and Briegel (2018)

In what follows, we focus on actor-critic methods because they exhibit an "all in one" structure from which the other approaches can be deduced by simplifications. To make it short and provide an overall picture, the so-called update rules for a single time step are listed in Eq. 30 and will be explained in the remainder of this section.

$$\delta \leftarrow R + \gamma_{\text{dis}} u' - u \qquad \text{(episodic)}, \tag{30a}$$
$$\delta \leftarrow R - \bar{R} + u' - u \qquad \text{(continuing)}, \tag{30b}$$

$$z^u \leftarrow \gamma_{\text{dis}} \lambda^u_{\text{tra}} z^u + \Gamma \nabla u, \tag{30c}$$
$$\boldsymbol{\theta}^u \leftarrow \boldsymbol{\theta}^u + \alpha^u \delta z^u, \tag{30d}$$

$$z^\pi \leftarrow \gamma_{\text{dis}} \lambda^\pi_{\text{tra}} z^\pi + \Gamma \nabla \ln \pi, \tag{30e}$$
$$\boldsymbol{\theta}^\pi \leftarrow \boldsymbol{\theta}^\pi + \alpha^\pi \delta z^\pi, \tag{30f}$$

$$\Gamma \leftarrow \gamma_{\text{dis}} \Gamma \text{(episodic)}, \tag{30g}$$
$$\bar{R} \leftarrow \bar{R} + \eta\, \delta \text{(continuing)}. \tag{30h}$$

In Eq. 30, we have two players: an actor (the policy $\pi = \pi(A|S, \boldsymbol{\theta}^\pi)$) and a critic (the value function $u = u(S, \boldsymbol{\theta}^u)$); hence, all corresponding quantities are distinguished by their respective superscript. The value function is parameterized by a weight vector $\boldsymbol{\theta}$. The vector z is referred to as the eligibility trace.

The update equations for the actor are given by Eq. 30e–f and the updates for the critic are Eq. 30c–d, where the $\nabla := \frac{\partial}{\partial \boldsymbol{\theta}}$ denote the gradients with respect to the $\boldsymbol{\theta}$ vectors. These two sets of updates are identical except for the natural logarithm $\ln \pi$ of the policy taken in Eq.30e. This logarithm is a consequence of the policy gradient theorem and makes the gradient of the actual performance measure to be maximized (value of the start state of an episode or average rate of reward per time step for continuing problems, see below) independent of the derivative of the state distribution (that is, the effect of the policy on the state distribution), which depends on the environment and is unknown.

Equation (30) describes *approximative* methods, since they apply **function approximation** as a scalable way of generalizing from a state space much larger then the memory capacity of the agent. The **tabular** case can be recovered from this as a special *exact* case, in the sense that all encountered states and actions are represented individually. The function approximations are hidden in the gradients ∇u and $\nabla \ln \pi$ in Eqs. 30c and 30e and can be done in linear fashion (by scalar products $\boldsymbol{\theta} \cdot \boldsymbol{x}$ with feature basis vectors $\boldsymbol{x}(S)$ or $\boldsymbol{x}(S, A)$) or in nonlinear fashion (e.g., by neural networks with $\boldsymbol{\theta}$ as connection weights)[4]. Note that the two parametrizations $\boldsymbol{\theta}^u$ and $\boldsymbol{\theta}^\pi$ are entirely unrelated (and hence different-dimensional in general). In Appendix B, we show in the example of SARSA, how tabular methods can be recovered from Eq. 30.

In Eq. 30, we can use a state value function, $u = u(S, \boldsymbol{\theta}^u)$, because the policy is taken care of separately. Without it, i.e., when we only know the value $u(s)$ of the state s we are in, we would require an **environment model** $p(s', r|s, a)$ to decide on an action a. To remain model-free, we would then have to apply an action value function $u = u(s, a, \boldsymbol{\theta}^u)$ instead, from which could obtain the best action by search for $\text{argmax}_a u$.

Equation 30 contains six meta-parameters: $\eta > 0$ and the two $\alpha > 0$ are step sizes, $\gamma_{\text{dis}} \in [0, 1]$ is the discount-rate parameter, and the two $\lambda_{\text{tra}} \in [0, 1]$ are trace-decay rates that allow to vary the degree of **bootstrapping**, which denotes the updating of estimates by using other existing estimates (cf. Appendix C). In 30, these existing estimates involve the current values u' of subsequent (i.e., one time

[4] These examples are not exhaustive. In a wider sense, one may also mention decision trees with $\boldsymbol{\theta}$ defining the split points and leaf values.

step later) states or state-action pairs, which enter the TD-error δ in either (30a) or (30b) together with the reward R. Choosing λ_{tra} is thus a possibility to interpolate between the fully bootstrapping original one-step TD methods which are recovered for $\lambda_{\text{tra}} = 0$, and **Monte Carlo** (i.e., non-bootstrapping) methods, which are obtained in the limit $\lambda_{\text{tra}} = 1$. Monte Carlo methods rely exclusively on actual complete returns G_t received. In a strict sense, they update **off-line**, i.e., they store a whole episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$ in a separate memory and only at the end of an episode the respective current estimates are updated in reverse time order $t = T-1, T-2, \ldots, 0$ making use of the fact that $G_t = R_{t+1} + \gamma_{\text{dis}} G_{t+1}$. In contrast to the updates (30), which are are done **online** (i.e., are incremental step-by-step), strict Monte Carlo methods are thus incremental in an episode-by-episode sense, and are consequently only defined for the episodic case. Consequently, even for $\lambda_{\text{tra}}=1$ the online updates (30) approximate Monte Carlo methods only for infinitesimally small step sizes α.

In continuing problems, the interaction between agent and environment goes on forever without termination or start states. Discounting is here useful in the tabular case but problematic when used with function approximation, where the states cannot be clearly distinguished anymore. An alternative then is to use the average rate of reward $r := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} E(R_t) = \lim_{t\to\infty} E(R_t)$, i.e., the average reward per time step (assuming ergodicity). $E(R_t)$ is the respective expected reward and Eq. 3 is replaced with the differential return $G_t := \sum_{k=0}^{\infty} (R_{t+k+1} - r)$. In Eq. 30, we thus set $\gamma_{\text{dis}} = 1$ in such a case and apply Eqs. 30b and 30h instead of Eqs. 30a and 30g, respectively. $\bar{R}$ is the current estimate of the average reward r.

To actually run (30), $\bar{R}$ and $\boldsymbol{\theta}$ can be initialized arbitrarily (e.g., to 0). At the beginning of each episode, $\boldsymbol{z}$ is initialized to 0, and Γ is initialized to 1. At the end of each episode, the value of a terminal state is set to 0.

Equation 30 are **on-policy** and must be distinguished from **off-policy** methods (such as Q-learning) which train on a distribution of transitions that is different from the one corresponding to the targeted (desired) behavior and thus free what the agent is actually doing (behavior policy) from what it should do (target policy). While this is not fundamentally required for basic problems that can be treated with model-free learning, it becomes essential in (model-based) prediction and planning, where it allows parallel learning of many target policies from the (necessarily one) realized behavior policy. Combining function approximation, bootstrapping, and off-policy updates may lead to instability and divergence.

At first glance, Eq. 30 looks elaborate and one might wonder why, for instance, value function-based methods should not suffice. The short answer is that this depends on the type and intricacy of the problem. To be more specific, one reason is that the expected return of state-action-pairs, that value functions estimate, typically contains more information than needed to make decisions. For example, a transformation of u which leaves the order of its values unchanged (such as multiplication with a positive constant) has no effect on the respective optimal decision. As a consequence, value function-based methods are too strict, since u are well defined and one needs to separately decide on a policy in order to convert these value estimates to action selection probabilities. If, for example a so-called softmax policy (14) is used, a choice and schedule (i.e., time-dependence) of the so-called inverse temperature parameter β has to be made. In contrast, direct policy methods, for instance, internally work with numerical preferences $h(s, a, \boldsymbol{\theta}^{\pi})$ whose actual values do not have to represent anything meaningful and are thus free to evolve in parameter space.

Appendix B: Recovering SARSA from actor-critic methods

To recover a pure action value method from the actor-critic methods (30), we restrict attention to Eqs. 30a, 30c, and 30d, set $\Gamma=1$ and ignore the remaining updates. For u, we choose an action value function $u = u(S, A, \boldsymbol{\theta}^{u})$ which we name as q. Suppressing the superscript u but adding time step indices, this gives for the scalar TD error signal:

$$\delta_t^{\text{TD}} = R_{t+1} + \gamma_{\text{dis}} q_t(S_{t+1}, A_{t+1}) - q_t(S_t, A_t), \tag{31}$$

with which the remaining updates describe SARSA(λ_{tra}) with function approximation:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \delta_t^{\text{TD}} \boldsymbol{z}_t, \tag{32}$$

$$\boldsymbol{z}_{t+1} = \gamma_{\text{dis}} \lambda_{\text{tra}} \boldsymbol{z}_t + \nabla q_{t+1}(S_{t+1}, A_{t+1}). \tag{33}$$

In the tabular case, q becomes a table (matrix) with entries $q(S_i, A_j)$. The components of the parameter vector $\boldsymbol{\theta}$ are identified with just these entries, so that the gradient becomes a table of Kronecker delta symbols:

$$\nabla q(S, A)|_t = \frac{\partial q(S, A)}{\partial q(S_t, A_t)} = (\delta_{S,S_t} \delta_{A,A_t}) =: \boldsymbol{\delta}_t. \tag{34}$$

To clarify, the bold $\boldsymbol{\delta}_t$ has the same dimension as q (i.e., it is a matrix) with the single entry corresponding to (S_t, A_t) (i.e., the state-action-pair visited at time t) being equal to 1 and all other entries being 0 and must be distinguished from the non-bold δ_t used throughout Section 3, which refers to a single given state-action-pair, and is 1 (0), if this pair is (not) visited at time t. With the bold $\boldsymbol{\delta}_t$, the updates (32)–(33) reduce to tabular SARSA(λ_{tra}):

$$q_{t+1} = q_t + \alpha \delta_t^{\text{TD}} \boldsymbol{z}_t, \tag{35}$$

$$\boldsymbol{z}_{t+1} = \gamma_{\text{dis}} \lambda_{\text{tra}} \boldsymbol{z}_t + \boldsymbol{\delta}_{t+1}, \tag{36}$$

where z is here called an accumulating eligibility trace and also has the same dimension as q (i.e., it is also a matrix). Hence, (35) updates *all* entries of q. For $\lambda_{\text{tra}} = 0$, only the respective (i.e., visited) *single* entry of q is updated:

$$q_{t+1}(S_t, A_t) = q_t(S_t, A_t) + \alpha \delta_t^{\text{TD}}, \tag{37}$$

which corresponds to conventional one-step SARSA.

Appendix C: Notes on eligibility trace vectors

We can here only outline a sketch. Let us focus on value function methods, where for simplicity of notation we restrict attention to state values. Action value methods can be covered analogously by referring to (s, a)-pairs instead of states s. We may consider the mean squared value error:

$$\overline{VE}(\boldsymbol{\theta}) := \sum_s \mu(s) \left[u(s) - \hat{v}(s, \boldsymbol{\theta})\right]^2 \tag{38}$$

between the true value function $u(s)$ and a current estimate $\hat{v}(s, \boldsymbol{\theta})$ of it. $\mu(s)$ can be any probability distribution, but is typically chosen to be the fraction of time spent in s under on-policy training in which case it is called on-policy distribution (or stationary distribution in continuing tasks). Ideally, one would find a global optimum $\boldsymbol{\theta}^*$ such that $\overline{VE}(\boldsymbol{\theta}^*) \leq \overline{VE}(\boldsymbol{\theta})\ \forall \boldsymbol{\theta}$. The problem is that $u(S_t)$ is unknown; hence, we substitute a so-called update target ("backed-up" value) U_t as a random approximation of the true value $u(S_t)$ and apply stochastic gradient descent:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{1}{2}\alpha_t \nabla \left[U_t - \hat{v}(S_t, \boldsymbol{\theta}_t)\right]^2 \tag{39}$$

$$= \boldsymbol{\theta}_t + \alpha_t \left[U_t - \hat{v}(S_t, \boldsymbol{\theta}_t)\right] \nabla \hat{v}(S_t, \boldsymbol{\theta}_t). \tag{40}$$

If U_t is an unbiased estimate of $u(S_t)$, i.e., $\mathbb{E}[U_t | S_t = s] = u(s)\ \forall t$, then convergence to a local optimum (i.e., the above inequality holds in a neighborhood of $\boldsymbol{\theta}^*$) follows under the stochastic approximation conditions for the step-size parameter $\alpha_t > 0$:

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty. \tag{41}$$

One possible choice for U_t is the λ-return:

$$G_t^{\lambda} := (1 - \lambda) \sum_{n=1}^{\infty} \lambda_{\text{tra}}^{n-1} G_{t:t+n} \tag{42}$$

as a mixture ($\lambda_{\text{tra}} \in [0, 1]$) of n-step returns at time t:

$$G_{t:t+n} := \sum_{k=0}^{n-1} \gamma_{\text{dis}}^k R_{t+k+1} + \gamma^n \hat{v}(S_{t+n}, \boldsymbol{\theta}_{t+n-1}). \tag{43}$$

Referring to episodic tasks with (random) termination time T, i.e., $t \leq T - n$ in Eq. 43, and $G_{t:t+n} = G_{t:T} = G_t$ for $n \geq T - t$, one can decompose:

$$G_t^{\lambda_{\text{tra}}} = (1 - \lambda_{\text{tra}}) \sum_{n=1}^{T-t-1} \lambda_{\text{tra}}^{n-1} G_{t:t+n} + \lambda_{\text{tra}}^{T-t-1} G_t, \tag{44}$$

in order to demonstrate the TD(0)-limit $G_t^{\lambda} \overset{\lambda_{\text{tra}} \to 0}{\to} G_{t:t+1}$ of recovering the one-step return and the Monte Carlo limit $G_t^{\lambda} \overset{\lambda_{\text{tra}} \to 1}{\to} G_t$ of recovering the conventional (full) return (3).

The incremental updates (40) refer to time t but involve via (43) information that is only obtained after t. Thus, we must rearrange these updates such that they are postponed to later times following a state visit. This is accomplished by eligibility trace vector updates such as (30c). To see this, one may sum up all increments (30d) over an episode (with (30c) substituted in explicit form) and compare this with the sum of all increments (40) (with $U_t = G_t^{\lambda}$) over an episode. Both total increments are equal if we neglect the influence of the change of $\boldsymbol{\theta}$ on the target U_t. This approximation holds for sufficiently small α_t, but strictly speaking, the bootstrapping involved in G_t^{λ}, namely its dependence on $\boldsymbol{\theta}$ via $\hat{v}(S_{t+n}, \boldsymbol{\theta}_{t+n-1})$ renders the transition from Eqs. 39 to 40 only a ("semi-gradient") approximation.

While a Monte Carlo target $U_t = G_t$ is by definition unbiased, Monte Carlo methods are plagued with high variance and hence slow convergence. In contrast, bootstrapping reduces memory resources, is more data-efficient, and often results in faster learning, but introduces bias. This is reminiscent of the "bias-variance tradeoff" which originally refers to supervised learning. While in the latter case the "trading" is realized by methods such as "regularization" or "boosting" and "bagging," in the context of RL considered here, choosing λ_{tra} serves a similar function.

Appendix D: Choice of glow update

In the following, different types of glow updates are discussed, which are useful for the comparison with other RL methods in Appendix E. We will focus on two types of glow, which we refer to as replacing and accumulating glow, motivated by the respective eligibility traces of the same name, which were originally introduced as tabular versions of the vector z and represent in the methods discussed in Appendix A the counterpart of glow introduced in Section 3. While replacing glow defined in Eq. 11 is the version applied in all works on PS so far, accumulating glow defined in Eq. 53 and first-visit glow defined in Eq. 15 are introduced to simplify the expressions and analysis. From the perspective of the methods considered in Appendix A, accumulating glow is more similar to the z-updates (30c) and (30e) than replacing glow, for which

such a generalization is not clear. As far as the choice of an eligibility trace in tabular RL methods is concerned, the tabular case of Eq. 30 yields accumulating traces as is shown in Eqs. 35–36 in the example of SARSA.

D.1 Replacing glow

In what follows, we discuss the update rules for replacing glow, which is helpful for the comparison with other RL methods. We consider the h-value of an arbitrarily given single (s, a) pair at time t. It is determined by the sequence of rewards λ_{j+1} $(j = 0, \ldots, t-1)$ and times of visits, which can be described using the sequence of Kronecker delta symbols δ_j. It is convenient to define the parameters:

$$\bar{\eta} := 1 - \eta, \tag{45a}$$

$$\bar{\gamma} := 1 - \gamma, \tag{45b}$$

$$\chi := \frac{\bar{\eta}}{\bar{\gamma}}. \tag{45c}$$

With them, we can express the recursion Eq. 10 combined with Eq. 11 in explicit form at time step t (see Appendix D.3 for details):

$$h_t = h_t^{\mathrm{res}} + h_t^{\mathrm{exp}}. \tag{46}$$

The first term:

$$h_t^{\mathrm{res}} = \bar{\gamma}^t h_0 + \left(1 - \bar{\gamma}^t\right) h^{\mathrm{eq}} \tag{47}$$

describes a transition from the initial h_0 to the asymptotic value h^{eq}. In particular, $h_t^{\mathrm{res}} = h_0$ holds exactly for $\bar{\gamma} = 1$ or for $h_0 = h^{\mathrm{eq}}$, and otherwise $h_t^{\mathrm{res}} \approx h^{\mathrm{eq}}$ holds asymptotically for times long enough such that $\bar{\gamma}^t \ll 1$. This reward-independent term h_t^{res} is always present in Eq. 46, and Eq. 46 reduces to it if the agent never receives any reward, i.e., if the agent is at "rest". (Note that in case of an exponential policy function (14), h_t^{res} has no effect on the policy, but this is not of concern for the present discussion.) The second term in Eq. 46 encodes the agent's "experience" and is determined by the history of visits and rewards. Let us refer to time step $t+1$ for convenience:

$$h_{t+1}^{\mathrm{exp}} = \sum_{k=l_1}^{t} \left[\bar{\gamma}^{t-k} \bar{\eta}^{k-l(k)}\right] \lambda_{k+1} \tag{48a}$$

$$= \sum_{j=1}^{j_t} \bar{\gamma}^{t-l_j} G\left[l_j : \min(l_{j+1} - 1, t), \chi\right] \tag{48b}$$

$$= \left(\sum_{j=1}^{j_t} \bar{\gamma}^{t-l_j} - \sum_{j=2}^{j_t} \chi^{l_j - l_{j-1}}\right) G(l_j : t, \chi). \tag{48c}$$

Here, l_j, $j = 1, \ldots, j_t$, are the times at which the respective edge is visited, i.e., $1 \le l_1 \le l_2 \le \ldots \le l_{j_t} \le t$, and j_t is thus the number of visits up to time t. In Eq. 48a, $l(k)$ is the time of the last visit with respect to time step k, i.e., if $l_j \le k < l_{j+1}$, then $l(k) = l_j$. Consequently, $k - l(k)$ is the number of steps that have passed at time k since the last visit occurred. In Eq. 48b, we have defined a truncated discounted return:

$$G\,(t : t+T, \chi) := \sum_{k=0}^{T} \chi^k \lambda_{t+k+1}, \tag{49}$$

which obeys

$$G(t_1 : t_2, \chi) = \lambda_{t_1+1} + \chi G(t_1 + 1 : t_2, \chi) \tag{50}$$

and more generally

$$G(t_1 : t_3, \chi) = G(t_1 : t_2 - 1, \chi) + \chi^{t_2 - t_1} G(t_2 : t_3, \chi). \tag{51}$$

Note that in Eq. 49, discounting starts at the respective t and not at $t = 0$; hence,

$$\sum_{k=t_1}^{t_2} \chi^k \lambda_{k+1} = \chi^{t_1} G(t_1 : t_2, \chi). \tag{52}$$

D.2 Accumulating glow

In the following, we introduce accumulating glow, which is defined by the following update:

$$g_{t+1} = (1 - \eta) g_t + \delta_{t+1}, \tag{53}$$

where each visit adds a value of 1 to the current glow value of the respective edge. Writing the recursion Eq. 10 combined with Eq. 53 instead of Eq. 11 in explicit form yields:

$$h_{t+1}^{\mathrm{exp}} = \sum_{k=0}^{t} \left[\bar{\gamma}^{t-k} \sum_{j=1}^{(l_j \le k)} \bar{\eta}^{k-l_j}\right] \lambda_{k+1} \tag{54a}$$

$$= \sum_{j=1}^{j_t} \bar{\gamma}^{t-l_j} G(l_j : t, \chi) \tag{54b}$$

instead of Eq. 48. The difference is that the subtracted sum in Eq. 48c, which represents "multiple re-visits," is not included in Eq. 54b.

D.3 Derivation of the explicit expressions for the h-value

Writing the recursion Eq. 10 in explicit form gives:

$$h_n = h_n^{\mathrm{res}} + h_n^{\mathrm{exp}}, \tag{55}$$

which corresponds to Eq. 46 with h_n^{res} given by Eq. 47 and

$$h_n^{\mathrm{exp}} = \sum_{k=0}^{n-1} \bar{\gamma}^{n-k-1} g_k \lambda_{k+1}. \tag{56}$$

D.3.1 Replacing glow

The recursion (11) for replacing glow yields the explicit expression:

$$g_n = \bar{\eta}^n g_0 \prod_{j=1}^{n} \bar{\delta}_j + \sum_{k=1}^{n} \bar{\eta}^{n-k} \delta_k \prod_{j=k+1}^{n} \bar{\delta}_j, \tag{57}$$

where we have defined

$$\bar{\delta}_j := 1 - \delta_j \tag{58}$$

for convenience. Setting $g_0 = 0$ gives

$$g_n = \sum_{k=1}^{n} \bar{\eta}^{n-k} \Delta(k, n), \tag{59}$$

where

$$\Delta(k, n) := \delta_k \prod_{j=k+1}^{n} \bar{\delta}_j \tag{60}$$

is 1 if the last visit occurred at step k ($k \leq n$, i.e., by definition $\Delta(n, n) := 1$) and 0 otherwise. Together with Eq. 56, we obtain after renaming n as $t+1$:

$$h_{t+1}^{\text{exp}} = \sum_{k=0}^{t} \bar{\gamma}^{t-k} g_k \lambda_{k+1} \tag{61}$$

$$= \sum_{k=0}^{t} \sum_{l=1}^{k} \bar{\gamma}^{t-k} \bar{\eta}^{k-l} \Delta(l, k) \lambda_{k+1}. \tag{62}$$

Rearranging summation ($\sum_{k=0}^{t} \sum_{l=1}^{k} = \sum_{l=1}^{t} \sum_{k=l}^{t}$), applying (49) together with Eqs. 51 and 52, and resolving the Kronecker delta symbols then gives Eq. 48.

D.3.2 Accumulating glow

Similarly, the recursion (53) for accumulating glow yields the explicit expression:

$$g_n = \bar{\eta}^n g_0 + \sum_{k=1}^{n} \bar{\eta}^{n-k} \delta_k, \tag{63}$$

where we set again $g_0 = 0$. Together with Eq. 56, we obtain after renaming n as $t+1$ the same expression as (61), from which Eqs. 76) and Eq. 77 follow. In Eq. 77, we have again rearranged summation and applied (49) together with Eq. 52. Resolving the Kronecker delta symbols then gives Eq. 54.

D.4 Order in glow updating

Note that in the updating of the replacing edge glow applied in Mautner et al. (2015), the glow value of visited edges is first reset to 1, followed by a damping of all edges by multiplying g with $\bar{\eta}$. In contrast, the recursion (11) for replacing glow applies the damping first and then resets the glow value of visited edges to 1. We may understand (11) as first making up for the damping of the previous step and then do the actual resetting of the present step. As an embedding description, we may define an s-ordered replacing glow update:

$$g_{t+1} = s\delta_{t+1} + \bar{\eta}\bar{\delta}_{t+1} g_t \tag{64}$$

generalizing (11), where s is a real valued ordering parameter. $s = \bar{\eta}$ describes the case of (1) resetting and (2) damping as done in Mautner et al. (2015) and Melnikov et al. (2018), whereas $s = 1$ describes the case of (1) damping and (2) resetting as done in Eq. 11. In explicit form, Eq. 64 yields:

$$g_n = \bar{\eta}^n g_0 \prod_{j=1}^{n} \bar{\delta}_j + s \sum_{k=1}^{n} \bar{\eta}^{n-k} \delta_k \prod_{j=k+1}^{n} \bar{\delta}_j \tag{65}$$

instead of Eq. 57. Analogously, an s-ordered accumulating glow update:

$$g_{t+1} = s\delta_{t+1} + \bar{\eta} g_t \tag{66}$$

generalizes Eq. 53 by describing (1) incrementing and (2) damping for $s = \bar{\eta}$ and (1) damping and (2) incrementing for $s = 1$ as done in Eq. 53. The explicit form of Eq. 66 is:

$$g_n = \bar{\eta}^n g_0 + s \sum_{k=1}^{n} \bar{\eta}^{n-k} \delta_k \tag{67}$$

instead of Eq. 63. The only difference is the extra factor s in the second term in Eq. 65 and (67) compared with Eq. 57 and (63), respectively, with which the h_{t+1}^{exp} in Eq. 61 (which holds for both types of glow) and hence in Eq. 48 and Eq. 54 would have to be multiplied. The difference is therefore minor and irrelevant for our considerations.

Appendix E: Comparative analysis of projective simulation and other RL methods

A specific contribution, which the PS updates have to offer to RL consists in supplementing the usual forward discounting with a backward discounting enabled by the damping of the (s, a) pair values, which amounts to a generalization of the standard notion of return. On the other hand, the incompatibility of discounting with function approximation mentioned in Appendix A may also extend to damping.

In the following discussion, we want to analyze the difference between PS and other RL methods. The first observation is that neither Eq. 48 nor 54 updates averages, instead they add "double-discounted" rewards. In what follows, first we show in Appendix E.1 how averaging can be implemented before we show in Appendix E.2 some simple effects of forward and backward discounting, assuming that averaging is carried out. Averaging will not

be integrated into the PS, as we do not want to give up the simplicity of the PS updates. This discussion merely serves as a thorough analysis of the differences between PS and methods that use averaging.

In the language of Appendix A, the basic PS updates Eq. 10 constitute a tabular model-free on-policy online learning method. In the analysis in Appendix E.4, we show that among the *methods in Appendix A, it is tabular SARSA(λ) defined in Eqs. 35–36, which comes closest to Eq. 10, because it has an eligibility value $z(s,a)$ ascribed to each (s,a) pair that is the counterpart of the respective glow value $g(s,a)$ and a trace decay parameter λ, which may be "meta-learned" (i.e., optimized). Thus, in Appendix E.4, we analyze the differences and similarities between PS and SARSA.

E.0.1 Implementing a temporal averaging

In this section, we show how temporal averaging can be integrated by adding to the h- and g-value a third variable $N=N(s,a)$ to each (s,a) pair, which counts the number of visits by updating it according to:

$$N_{t+1} = N_t + \delta_{t+1}, \tag{68}$$

which is formally the same update as Eq. 53 for undamped accumulating glow. With it, we could consider the normalized $\tilde{h}_t = h_t / N_t$ and initialize with $N_0 = 1$ to avoid division by zero, so that explicitly $N_t = N_0 + \sum_{k=1}^{t} \delta_k = j_t + 1$. Alternatively, we can integrate the normalization into the update rule f, $h_{t+1} = N_{t+1}\tilde{h}_{t+1} = f(h_t) = f(N_t\tilde{h}_t)$ by replacing Eq. 10 with:

$$\begin{aligned}\tilde{h}_{t+1} &= \frac{\alpha_{t+1}}{\alpha_t}\tilde{h}_t - \gamma\left(\frac{\alpha_{t+1}}{\alpha_t}\tilde{h}_t - \alpha_{t+1}h^{\mathrm{eq}}\right) + \alpha_{t+1}g_t\lambda_{t+1} \\ &\approx \tilde{h}_t - \gamma(\tilde{h}_t - \alpha_t h^{\mathrm{eq}}) + \alpha_t g_t \lambda_{t+1}, \end{aligned} \tag{69}$$

$$\alpha_{t+1} = \frac{\alpha_t}{1+\alpha_t\delta_{t+1}}, \tag{70}$$

where the approximation (69) holds as soon as $\alpha_t \ll 1$. Instead of N, we thus then keep for each (s,a) pair a separate time-dependent learning rate $\alpha_t = N_t^{-1} = \alpha_t(s,a)$ and update it according to Eq. 70.

For accumulating glow, Eq. 54b sums over all visits j the backward-discounted returns that follow these visits up to the present t, and $\tilde{h}_{t+1}^{\mathrm{exp}}$ thus becomes (for large t) an estimate of the average backward-discounted return that follows a visit. In contrast, the first-visit counterpart (15) only depends on the time l_1 of the first visit, which is analytically more easily analyzed in an episodic environment, where after each episode, the glow values of all (s,a) pairs are reset to 0.

The updates involving (68) or (70) may be read as a laborious reinvention of an online approximation of an every-visit Monte Carlo method, but provide the following insight: For the action value methods in the context of Appendix **??**, the learning rate can in practice (especially when dealing with deterministic environments) often be kept constant rather than gradually decreasing it, where the precise value of this constant does not matter. For our updates of $\tilde{h}$, omitting the correction by N or α and working with the original h should work reasonably well, too, in such problems.

E.0.2 Effect of double discounting on a temporal average

As an elementary illustration of the effect of forward discounting via $\bar{\eta}$ and backward discounting via $\bar{\gamma}$ on agent learning consider a weighted arithmetic mean:

$$\bar{x}^{(t)} = \frac{\sum_{k=1}^{t} w_k x_k}{\sum_{k=1}^{t} w_k} \tag{71}$$

of random samples x_k with variable but known weights $w_k \geq 0$ ($w_1 > 0$). If the samples are drawn in succession $x_1, x_2, \ldots$, then the average can be updated incrementally:

$$\bar{x}^{(t)} = (1-\alpha_t)\bar{x}^{(t-1)} + \alpha_t x_t, \tag{72}$$

with a "learning rate"

$$\alpha_t = \frac{w_t}{\sum_{k=1}^{t} w_k}, \tag{73}$$

which in general fluctuates within $\alpha_t \in [0,1)$ depending on the weight sequence $w_1, w_2, \ldots$. (Note that an incremental formulation:

$$\alpha_{t+1} = \left(1 + \frac{w_t}{w_{t+1}\alpha_t}\right)^{-1} \tag{74}$$

would require that $w_k > 0$ holds $\forall k$.) Of particular interest for our discussion is an exponential choice of weights:

$$w_k = w^k, \quad \alpha_t = \frac{1-w^{-1}}{1-w^{-t}}. \tag{75}$$

In Eq. 75, we can distinguish the following cases:

(a) For $w = 1$ all samples are given equal weight and the learning rate $\alpha_t = t^{-1} \overset{t\to\infty}{\to} 0$ decays to 0 in a manner of Eq. 41. In the special case, when the x_k are drawn from i.i.d. random variables $X_k \equiv X$ with variance $\sigma^2(X) = \sigma^2$, the total variance $\sigma^2_{(t)} = t^{-1}\sigma^2$ of $\frac{\sum_{k=1}^{t} w_k X_k}{\sum_{k=1}^{t} w_k}$ vanishes with growing t and $\bar{x}^{(t)}$ converges to the expectation value $E(X)$. In the context of agent learning, we may interpret $\bar{x}^{(t)}$ as the agent's experience (e.g., a current value estimate of some given state-action pair). If after some longer time $t \gg 1$, the environment statistics changes ($X_k \equiv X$ no longer holds), the average $\bar{x}^{(t)}$ will start to change only slowly.

(b) The case $w < 1$ in Eq. 75 corresponds to the effect of a discounting from the beginning of learning towards the present t by the factor $\bar{\eta}^k$. The learning rate α_t

$\overset{t\gg 1}{\approx}(w^{-1}-1)w^t \overset{t\to\infty}{\to} 0$ decays to 0 exponentially and the agent will cease to learn anything after some finite time period of the order $\Delta t \approx -(\ln w)^{-1}$ due to decay of the weights in Eq. 75. After that time, the agent will behave solely according to this early experience "imprinted" into it.

(c) The case $w > 1$ in Eq. 75 corresponds to the effect of discounting from the present t toward the past by a damping factor $\bar{\gamma}^{t-k}$. The learning rate $\alpha_t \overset{t\to\infty}{\to} 1 - w^{-1}$ converges to a positive constant and the agent remains "fluid" in its ability to react to changes in the environment statistics. On the other hand, since its remembered experience reaches only a time period of the order $\Delta t \approx (\ln w)^{-1}$ from the present into the past, such an agent will just "chase the latest trend" without learning anything properly.

In the special case, when the x_k are drawn from i.i.d. random variables $X_k \equiv X$ with variance $\sigma^2(X) = \sigma^2$, the total variance $\sigma^2_{(t)} \overset{t\gg 1}{\approx} \frac{|w-1|}{w+1}\sigma^2$ of $\frac{\sum_{k=1}^t w_k X_k}{\sum_{k=1}^t w_k}$ converges to a positive constant in both cases (b) and (c), that is, when $w \neq 1$. The difference is that in case (c), the weighted mean $\bar{x}^{(t)}$ keeps fluctuating, whereas in case (b), this variance has been "crystallized" into a bias $\bar{x}^{(t)} - E(X)$ of the early experience $\bar{x}^{(t)}$ which is fixed by the samples in (71) with respect to the actual $E(X)$.

E.0.3 Description of a formal ensemble average

We restrict attention to the simpler accumulating glow (54), which we rewrite with the Kronecker delta symbols kept explicitly:

$$h^{\text{exp}}_{t+1} = \sum_{k=0}^{t}\sum_{l=1}^{k} \bar{\gamma}^{t-k}\bar{\eta}^{k-l}\lambda_{k+1}\delta_l \tag{76}$$

$$= \sum_{l=1}^{t} \bar{\gamma}^{t-l} G(l:t,\chi)\delta_l \tag{77}$$

(see Appendix D.3.1 for the corresponding expressions describing replacing glow). Each δ_l samples the "occupation" of the given (s,a) pair at time l, whose probability is given by $p_l(s,a)$. For an ensemble of independent agents running in parallel, we can thus replace the δ_l with these probabilities and write:

$$\left\langle h^{\text{exp}}_{t+1}\right\rangle_{\text{ens}} = \sum_{l=1}^{t} \bar{\gamma}^{t-l} G(l:t,\chi) p_l \tag{78}$$

$$= \left\langle \bar{\gamma}^{t-l} G(l:t,\chi)\right\rangle_{l=1,\ldots,t}. \tag{79}$$

While Eq. 77 sums for all times l the respective backward-discounted return $\bar{\gamma}^{t-l}G(l:t,\chi)$ from that time *under the condition that* the edge was visited, the ensemble average (78) performs an average with respect to the p_l over all times l up to the present t. The problem is that we do not know the distribution p_l, which itself is affected by both the environment and the agent's policy.

What we can do, however, is to consider the average return *that follows a visit* at given time l. The average number $n_l = Np_l$ of visits per unit of time at time l is for an ensemble of size N just given by p_l, with which we normalize each summand in Eq. 78. The sum over all times l of these average returns per visit with respect to time l can then be written as:

$$\widetilde{\left\langle h^{\text{exp}}_{t+1}\right\rangle}_{\text{ens}} = \frac{1}{N}\sum_{l=1}^{t} \frac{\bar{\gamma}^{t-l} G(l:t,\chi) p_l}{p_l} \tag{80}$$

$$= \frac{1}{N}\sum_{l=1}^{t} \bar{\gamma}^{t-l} G(l:t,\chi) \tag{81}$$

$$= \frac{1}{N}\frac{\bar{\gamma}^t}{1-\bar{\eta}}\left[G(0:t,\bar{\gamma}^{-1}) - G(0:t,\chi)\right]. \tag{82}$$

The normalization in (80) is analogous to the one that motivated the logarithm in (30e) as discussed in Appendix A and E: it makes the expression independent of the state distribution p_l. It also reveals that what we have called "double discounting," i.e., the convolution (81) of the return $G(l:t,\chi)$ with the exponential $\bar{\gamma}^l$ amounts to the difference (82) between two returns from the beginning $t=0$.

For a single agent in Appendix E.1, there cannot be more than one visit at each time l. We therefore had to take the sum h^{exp}_{t+1} at time t, and divide it by the total (cumulative) number of visits N_t that occurred up to this time, to get an estimate $\tilde{h}_t = h_t/N_t$ of the average return per visit. One possibility to implement (80) for a single agent consists in training the agent "off-policy" by separating exploration and exploitation, which can be done by choosing the softmax policy (14). During periods of exploration (e.g., if the agent is not needed), we choose a small β, whereas for exploitation, we temporarily disable the updating Eq. 10 and switch to a large β. By large (small) we mean values of β such that for all $x = h_{ij}$ encountered in Eq. 14, the argument of the exponential obeys $\beta x \gg 1$ ($\beta x \ll 1$). For graphs that have for symmetry reasons the property that $p_l \equiv p$ for a random walker (note that for ergodic MDPs the p_l eventually becomes independent of the initial conditions), we should be able to realize (80) during the periods of exploration. It is clear that this is impractical for all but small finite MDPs.

E.0.4 Relation of the PS updates to other RL methods

In this section, we compare PS to the standard RL methods presented in Appendix A. One may interpret the PS updates Eq. 10 as implementing a direct policy method. On the

other hand, these updates do not involve gradients. To draw connections between PS and direct policy methods, we consider the gradient ∇p of the probability $p_{ij} = p(a_j|s_i)$ of selecting action a_j in state s_i, i.e., one element of the policy π and restrict to our case (12), i.e., $p_{ij} = \frac{\Pi(h_{ij})}{\kappa_i}$. As in the derivation (34) of tabular SARSA, we identify the components of the parameter vector $\boldsymbol{\theta}$ (with respect to which we want to determine the gradient) with the edges h_{kl}. The gradient of p thus becomes a matrix, whose element kl reads:

$$(\nabla p_{ij})_{kl} = \frac{\partial p_{ij}}{\partial h_{kl}} = \frac{\Pi'(h_{il})}{\kappa_i}\left[\delta_{ki}\delta_{lj} - \frac{\Pi(h_{ij})}{\kappa_i}\delta_{ki}\right], \quad (83)$$

where $\Pi'(x) = \frac{d\Pi(x)}{dx}$. To obtain $\nabla \ln p$, we just multiply this with the factor $p_{ij}^{-1} = \frac{\kappa_i}{\Pi(h_{ij})}$. The term $\delta_{ki}\delta_{lj}$ is also present in the PS update, where it corresponds to the strengthening of a visited edge. The subtracted second term proportional to δ_{ki} represents a weakening of all edges connecting s_i, which is not present in the PS update.

With Eq. 83, we can now compare the PS updates Eq. 10 with the methods in Fig. 2. Among the action value methods, it is tabular SARSA(λ) defined in Eqs. 35–36, which resembles the PS updates most. Let us rewrite the SARSA updates in the notation used within this. After renaming the reward R as λ, the action value function q as h, the eligibility vector $\boldsymbol{z}$ as (matrix) g, the discount rate γ as γ_{dis}, and the trace-decay rate λ as λ_{tra} for clarity, the TD error (31) reads:

$$\delta_t^{\text{TD}} = \lambda_{t+1} + \gamma_{\text{dis}} h_t(s_{t+1}, a_{t+1}) - h_t(s_t, a_t), \quad (84)$$

with which (35)–(36) for SARSA(λ_{tra}) become:

$$h_{t+1} = h_t + \alpha \delta_t^{\text{TD}} g_t, \quad (85)$$

$$g_{t+1} = \gamma_{\text{dis}}\lambda_{\text{tra}} g_t + \boldsymbol{\delta}_{t+1}, \quad (86)$$

where $\boldsymbol{\delta}_t$ is a matrix of Kronecker deltas describing which of the (s, a) pairs has been visited at time t. A tabular direct policy method follows in the same way from (30a), (30e), and (30h) (setting again $\Gamma = 1$): (84) and (85) remain identical, merely (86) is replaced with:

$$g_{t+1} = \gamma_{\text{dis}}\lambda_{\text{tra}} g_t + (\nabla \ln p)_{t+1}, \quad (87)$$

where for $\nabla \ln p$ we substitute (83) together with the extra factor as explained in the text following (83). While (86) recovers the update (53) for accumulating glow ((53) considers a given (s, a) pair, Eq. 86 the whole matrix), Eq. 87 is in fact even more complex than (86).

One important difference is the presence of the term $h_t(s_{t+1}, a_{t+1})$ in Eq. 84 which persists even if we disable bootstrapping by setting $\lambda_{\text{tra}} = 1$. We can also rewrite SARSA in the "local" fashion of the PS updates (10), which we here repeat as:

$$h_{t+1} = h_t + \lambda_{t+1} g_t - \gamma h_t + \gamma h^{\text{eq}} \quad (88)$$

for convenience. To rewrite SARSA(λ_{tra}) in the form of (88), we proceed as in the justification of accumulating traces outlined in Appendix C. First, we sum all increments in (85) up to some time T, i.e., $h_T = h_0 + \alpha \sum_{t=0}^{T-1} \delta_t^{\text{TD}} g_t$, then rewrite the term involving $h_t(s_{t+1}, a_{t+1})$ in δ_t^{TD} as $\sum_{t=0}^{T-1} h_t(s_{t+1}, a_{t+1}) g_t = \sum_{t=1}^{T} h_{t-1}(s_t, a_t) g_{t-1}$ and substitute $g_{t-1} = \frac{g_t - \delta_t}{\gamma_{\text{dis}}\lambda_{\text{tra}}}$. If we now ignore (a) the change of the h-values over a single time step (which holds for small α), $h_{t-1}(s_t, a_t) \approx h_t(s_t, a_t)$, and (b) ignore the shift of argument in the summation (i.e., ignore the first and last sum terms), then identifying each term referring to a given t in the sum over all increments with an individual update leads to a "PS-style" form of SARSA(λ_{tra}):

$$h_{t+1} = h_t + \alpha\lambda_{t+1} g_t - \alpha h_t(s_t, a_t)\left[\lambda_{\text{tra}}^{-1}\boldsymbol{\delta}_t + (1 - \lambda_{\text{tra}}^{-1}) g_t\right], \quad (89)$$

in which γ_{dis} no longer appears (it remains in (86)).

We can simplify Eq. 89 if we disable bootstrapping by setting $\lambda_{\text{tra}} = 1$, so that it becomes even more similar to PS. On the other hand, if we take into account that PS does not use averaging, PS carries some similarities to (an online approximation of) Monte Carlo approaches. The type of glow then determines the corresponding type of Monte Carlo method. For example, using replacing glow relates it more to first-visit Monte Carlo, whereas accumulating glow relates it more to every-visit Monte Carlo.

Appendix F: Mathematical details of the convergence proof

In this appendix, we provide the mathematical details we skipped during the proof of Theorem 2 in Section 4. We are left with showing that $\alpha_m(\boldsymbol{e})$, given in Eq. 27 satisfies Condition 2 in Theorem 1, and $F_m(\boldsymbol{e})$, given in Eq. 28, satisfies Conditions 3 and 4. From these, Condition 3 is the most involving, and to some extent is the core of the proof. Condition 4 follows trivially under our assumption of bounded rewards. One can easily see that bounded rewards imply that $\tilde{h}$ values are upper and lower bounded. Given that optimal Q-values are bounded as well it follows that $\text{Var}\{F_m(\boldsymbol{e})|P_m\} \leq K'$ for some constant K'. The remaining two properties are proven in the following.

F.1 Proving that $\alpha_m(\boldsymbol{e})$ satisfies Condition 2 in Theorem 1

Let us recall that:

$$\alpha_m(\boldsymbol{e}) := \frac{\chi_m(\boldsymbol{e})}{N_m(\boldsymbol{e}) + 1} = \frac{\chi_m(\boldsymbol{e})}{\sum_{j=1}^{m} \chi_j(\boldsymbol{e}) + 1}, \quad (90)$$

where the $\chi_m(\boldsymbol{e})$ are given by:

$$\chi_m(\boldsymbol{e}) = \begin{cases} 1 & \text{if } \boldsymbol{e} \text{ was visited during the } m\text{th episode,} \\ 0 & \text{otherwise.} \end{cases} \tag{91}$$

Due to the fact that the policy guarantees infinite exploration, we know that the number of non-zero terms from the sequence $\mathcal{Q}_1 := \{\alpha_m(\boldsymbol{e})\}_{1\leq m<\infty}$ is infinite. Thus, let $\mathcal{Q}_2 := \{\tilde{\alpha}_n(\boldsymbol{e})\}_{1\leq n<\infty}$ be the subsequence of $\mathcal{Q}_1$ obtained by removing all zero elements, and relabeling the remaining ones as 1,2,3, etc. Clearly, we have that:

$$\sum_{m=1}^{\infty} \alpha_m(\boldsymbol{e}) = \sum_{n=1}^{\infty} \tilde{\alpha}_n(\boldsymbol{e}), \tag{92}$$

$$\sum_{m=1}^{\infty} [\alpha_m(\boldsymbol{e})]^2 = \sum_{n=1}^{\infty} [\tilde{\alpha}_n(\boldsymbol{e})]^2. \tag{93}$$

Furthermore, it is trivial to see that the non-zero terms $\tilde{\alpha}_n(\boldsymbol{e}) = 1/(n+1)$. Given that $\sum_n 1/n = \infty$ and $\sum_n 1/n^2 < \infty$, it follows that:

$$\sum_{m=1}^{\infty} \alpha_m(\boldsymbol{e}) = \infty, \tag{94}$$

$$\sum_{m=1}^{\infty} \alpha_m^2(\boldsymbol{e}) < \infty, \tag{95}$$

as we wanted to prove.

F.2 Contraction of $F_m(\boldsymbol{e})$

In this part of the appendix, we show that in the case where the glow parameter of the PS model $\bar{\eta}$ is set equal to the discount parameter γ_{dis} associated to the MDP, $F_m(\boldsymbol{e}) := D_m(\boldsymbol{e}) - q_*(\boldsymbol{e})$ satisfies:

$$\|\mathrm{E}\{F_m(\cdot)|P_m\}\|_{\mathrm{W}} \leq f(\gamma_{\text{dis}})\|\Delta_m(\cdot)\|_{\mathrm{W}} + c_m, \tag{96}$$

where $\Delta_m(e) := \tilde{h}_m(e) - q_*(e)$, $f(\gamma_{\text{dis}}) = \frac{2\gamma_{\text{dis}}}{1-\gamma_{\text{dis}}}$ and c_m converges to 0 w.p.1.

In the update rule for $\Delta_m(\boldsymbol{e})$ given in Eq. 26, $F_m(\boldsymbol{e})$ appears multiplied by the learning rate coefficient $\alpha_m(\boldsymbol{e})$. Given that $\alpha_m(\boldsymbol{e}) = 0$ in the case where $\chi_m(\boldsymbol{e}) = 0$, we can, w.l.o.g., define $F_m(\boldsymbol{e}) = 0$ for that case. This is made explicit by the factor $\chi_m(\boldsymbol{e})$ in the definition of $F_m(\boldsymbol{e})$ given in Eq. 28, which for $\bar{\eta} = \gamma$ leads to:

$$F_m(\boldsymbol{e}) = \chi_m(\boldsymbol{e}) \left(\sum_{j=0}^{T_m - t_m(\boldsymbol{e})} \gamma^j \lambda_{t_m(\boldsymbol{e})+j} - q_*(\boldsymbol{e}) \right). \tag{97}$$

Following this definition of $F_m(\boldsymbol{e})$, we have that:

$$\mathrm{E}\{F_m(\boldsymbol{e})|P_m\} = \mathrm{E}\{F_m(\boldsymbol{e})|\chi_m(\boldsymbol{e}) = 1, P_m\}. \tag{98}$$

To simplify the notation, in the following, we will always assume that $\boldsymbol{e}$ has been visited at least once during episode m, but for simplicity in our notation we omit writing the condition on $\chi_m(\boldsymbol{e}) = 1$ in the expected value.

The past of the process at episode m, P_m, includes every state, action, and reward received by the agent from $t = 0$ until the beginning of the episode m. In particular, it determines the set of $\tilde{h}_m$ values, which in turn determines the policy at the beginning of the mth episode. For the clarity of this proof, we will first consider the case where the policy is kept unchanged during episodes and only updated at the beginning of a new episode, showing that Eq. 96 holds under those assumptions. Later on, we relax that condition and show that the differences accumulated by the policy during the episode converge to 0 with probability 1. This allows us to prove Eq. 96 also in the case where the policy is updated every time step.

F.2.1 Constant policies during the episodes

Given that the number of time steps required by an agent to hit a terminal state is unbounded, the number of terms in $F_m(\boldsymbol{e})$ could be arbitrarily large. Therefore, we construct a family of truncated versions of $F_m(\boldsymbol{e})$, where the maximum number of terms is upper bounded. Let us define:

$$\begin{aligned} F_m^{(k)}(\boldsymbol{e}) := &\sum_{j=0}^{k} \Theta(T_m - t_m(\boldsymbol{e}) - j)\gamma_{\text{dis}}^j \lambda_{t_m(\boldsymbol{e})+j} \\ &+ \Theta(T_m - t_m(\boldsymbol{e}) - k)\gamma_{\text{dis}}^{t+1} \tilde{h}_m(S_{k+1}, A_{k+1}) - q_*(\boldsymbol{e}), \end{aligned} \tag{99}$$

where $\Theta(l) = 1$ for $l \geq 0$ and it is 0 otherwise, and we have defined $\tilde{h}(s_T, a) = 0$, for all terminal states s_T[5]. Comparing Eqs. 97 and 99, one can see that $F_m^{(k)}(\boldsymbol{e}) = F_m(\boldsymbol{e})$ in the case where the agent takes less than k time steps to finish the episode since $\boldsymbol{e}$ is visited for the first time during the mth episode, i.e., whenever $T_m - t_m(\boldsymbol{e}) < k$. Considering that the policy guarantees infinite exploration, the probability of not reaching a terminal state after k time steps (during a single episode) decays exponentially with k, and therefore:

$$\mathrm{E}\{F_m(\boldsymbol{e})|P_m\} = \lim_{k\to\infty} \mathrm{E}\{F_m^{(k)}(\boldsymbol{e})|P_m\}. \tag{100}$$

In the following, we construct an upper bound for $\mathrm{E}\{F_m^{(k)}(\boldsymbol{e})|P_m\}$ that holds for all k, and hence due to Eq. 100 it also bounds $\mathrm{E}\{F_m(\boldsymbol{e})|P_m\}$. We can write $F_m^{(k)}(\boldsymbol{e})$ in the

[5] Since episodes end when a terminal state is reached, the PS agent does not need to store h-values associated with terminal states; thus, we can define them equal to 0.

following form:

$$\begin{aligned}\mathrm{E}\{F_m^{(k)}(\boldsymbol{e})|P_m\} &= r(\boldsymbol{e}) \\ &+ \sum_{l=1}^{k} \sum_{\boldsymbol{e}^{(1)},\ldots\boldsymbol{e}^{(l)}} \Pr\left[\boldsymbol{e}, \boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(l)}|P_m\right] \gamma_{\mathrm{dis}}^{l} r(\boldsymbol{e}^{(l)}) \\ &+ \sum_{\boldsymbol{e}^{(1)},\ldots,\boldsymbol{e}^{(k+1)}} \Pr\left[\boldsymbol{e}, \boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(k+1)}|P_m\right] \gamma_{\mathrm{dis}}^{k+1} \tilde{h}_m(\boldsymbol{e}^{(k+1)}) \\ &- q_*(\boldsymbol{e}),\end{aligned} \tag{101}$$

where we used the short-hand notation $\Pr\left[\boldsymbol{e}, \boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(l)}|P_m\right]$ to denote the probability of an agent following the sequence of state-action pairs $\boldsymbol{e}, \boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(l)}$. Given that, for the moment, we are considering constant policies during episodes, these probabilities only depend on the episode index m. In addition, in order to have a simpler expression, w.l.o.g. we assume that transitions from a terminal state return with probability 1 to a terminal state with zero reward. Note that this assumption together with the fact that $\tilde{h}_m(s_T, a) = 0$ for all terminal states s_T allows us to write the summations in Eq. 101 over all edges, including those associated to terminal states.

The following step consists in writing $\mathrm{E}\{F_m^{(k)}(\boldsymbol{e})|P_m\}$ as a recursive relation in k. Given that the policies are kept constant during episodes they satisfy that:

$$\begin{aligned}&\Pr\left[\boldsymbol{e}, \boldsymbol{e}^{(1)}, \ldots \boldsymbol{e}^{(k+1)}|P_m\right] = \\ &\quad \Pr\left[\boldsymbol{e}, \boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(k)}|P_m\right] \Pr\left[\boldsymbol{e}^{(k)}, \boldsymbol{e}^{(k+1)}|P_m\right].\end{aligned} \tag{102}$$

Plugging this equation into Eq. 101 and adding canceling terms, we end up with the expression:

$$\begin{aligned}\mathrm{E}\{F_m^{(k)}(\boldsymbol{e})|P_m\} &= r(\boldsymbol{e}) \\ &+ \sum_{l=1}^{k-1} \sum_{\boldsymbol{e}^{(1)},\ldots,\boldsymbol{e}^{(l)}} \Pr\left[\boldsymbol{e}, \boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(l)}|P_m\right] \gamma_{\mathrm{dis}}^{l} r(\boldsymbol{e}^{(l)}) \\ &+ \sum_{\boldsymbol{e}^{(1)},\ldots,\boldsymbol{e}^{(k)}} \Pr\left[\boldsymbol{e}, \boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(k)}|P_m\right] \gamma_{\mathrm{dis}}^{k} \tilde{h}_m(\boldsymbol{e}^{(k)}) - q_*(\boldsymbol{e}) \\ &+ \sum_{\boldsymbol{e}^{(1)},\ldots,\boldsymbol{e}^{(k)}} \Pr\left[\boldsymbol{e}, \boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(k)}|P_m\right] \gamma_{\mathrm{dis}}^{k} \Big\{ -\tilde{h}_m(\boldsymbol{e}^{(k)}) \\ &+ q_*(\boldsymbol{e}^{(k)}) + r(\boldsymbol{e}^{(k)}) \\ &+ \sum_{\boldsymbol{e}^{(k+1)}} \Pr[\boldsymbol{e}^{(k)}, \boldsymbol{e}^{(k+1)}|P_m] \gamma_{\mathrm{dis}} \tilde{h}_m(\boldsymbol{e}^{(k+1)}) - q_*(\boldsymbol{e}^{(k)}) \Big\}.\end{aligned} \tag{103}$$

Comparing Eqs. 101 and 103, one can see that the first two lines in the previous equation equal $\mathrm{E}\{F_m^{(k-1)}(\boldsymbol{e})|P_m\}$. Furthermore, in the third and fourth lines, the quantities within curly brackets correspond to the definition of $\Delta_m(\boldsymbol{e}^{(k)})$ and $\mathrm{E}\{F_m^{(0)}(\boldsymbol{e}^{(k)})|P_m\}$ respectively. Hence, $|\mathrm{E}\{F_m^{(k)}(\boldsymbol{e})|P_m\}|$ obeys the following recursive relation:

$$\begin{aligned}\left|\mathrm{E}\{F_m^{(k)}(\boldsymbol{e})|P_m\}\right| &\leq \left|\mathrm{E}\{F_m^{(k-1)}(\boldsymbol{e})|P_m\}\right| \\ &+ \gamma_{\mathrm{dis}}^{k} \|\Delta_m(\cdot)\|_{\mathrm{W}} \\ &+ \gamma_{\mathrm{dis}}^{k} \|\mathrm{E}\{F_m^{(0)}(\cdot)|P_m\}\|_{\mathrm{W}}.\end{aligned} \tag{104}$$

By iterating the previous equation, we achieve the following bound:

$$\begin{aligned}&\left|\mathrm{E}\{F_m^{(k)}(\boldsymbol{e})|P_m\}\right| \\ &\leq \sum_{l=0}^{k} \gamma_{\mathrm{dis}}^{l} \|\mathrm{E}\{F_m^{(0)}(\cdot)|P_m\}\|_{\mathrm{W}} + \sum_{l=1}^{k} \gamma_{\mathrm{dis}}^{l} \|\Delta_m(\cdot)\|_{\mathrm{W}} \\ &\leq \frac{1}{1-\gamma_{\mathrm{dis}}} \left(\|\mathrm{E}\{F_m^{(0)}(\cdot)|P_m\}\|_{\mathrm{W}} + \gamma_{\mathrm{dis}} \|\Delta_m(\cdot)\|_{\mathrm{W}} \right),\end{aligned} \tag{105}$$

where we have used the relation $\sum_{l=0}^{\infty} \gamma_{\mathrm{dis}}^{l} = \frac{1}{1-\gamma_{\mathrm{dis}}}$ to obtain a bound that is independent of both $\boldsymbol{e}$ and k.

Notice that $F_m^{(0)}(\boldsymbol{e})$ corresponds to the kind of update term encountered in the single-step algorithm of SARSA. It has been proven in Singh et al. (2000) as part of the convergence proof of the SARSA method that $F_m^{(0)}$ satisfies:

$$\|\mathrm{E}\{F_m^{(0)}(\cdot)|P_m\}\|_{\mathrm{W}} \leq \gamma_{\mathrm{dis}} \|\Delta_m(\cdot)\|_{\mathrm{W}} + d_m, \tag{106}$$

where d_m converges to 0 w.p.1. as $m \to \infty$. Here, we recall from Singh et al. (2000) the main mathematical steps to prove Eq. 106 as they will be useful later, when we consider the general scenario with time-dependent policies. The expected value of $F_m^{(0)}(\boldsymbol{e})$ can be written explicitly as:

$$\begin{aligned}&\mathrm{E}\{F_m^{(0)}(s,a)|P_m\} \\ &= r(s,a) + \gamma_{\mathrm{dis}} \sum_{s'} \Pr(s'|s,a) \sum_{a'} \Pr\left(a'|s', P_m\right) \tilde{h}_m(s', a') \\ &\quad - q_*(s,a) \\ &= f_m(s,a) + \gamma_{\mathrm{dis}} \sum_{s'} \Pr(s'|s,a) g_m(s'),\end{aligned} \tag{107}$$

where we have defined:

$$\begin{aligned}f_m(s,a) &= r(s,a) + \gamma_{\mathrm{dis}} \sum_{s'} \Pr\left(s'|s,a\right) \max_b \{\tilde{h}_m(s',b)\} \\ &\quad - q_*(s,a), \\ g_m(s') &= \sum_{a'} \Pr\left(a'|s', P_m\right) \tilde{h}_m(s',a') - \max_b \{\tilde{h}_m(s',b)\}.\end{aligned} \tag{108}$$

The first term given above corresponds to the update term encountered in Q-learning algorithms and it has been proven to be bounded by:

$$|f_m(s,a)| \leq \gamma_{\mathrm{dis}} \|\Delta_m(\cdot)\|_{\mathrm{W}}, \forall s, a. \tag{109}$$

In order to bound $g_m(s)$, let us recall that the policy (under the assumption that it is kept constant during an

episode) is given by:

$$\Pr(a|s, P_m) = \frac{\exp[\beta_m \tilde{h}_m(s,a)]}{\sum_b \exp[\beta_m \tilde{h}_m(s,b)]}. \quad (110)$$

By simple algebraic manipulation, one can see that:

$$g_m(s) < \frac{n_a}{\beta_m}, \forall s, \quad (111)$$

where $n_a = ||\mathcal{A}||$ is the number of actions (assumed finite). Due to the fact that $\beta_m \to \infty$ as $m \to \infty$ it follows that $g_m(s)$ converges to 0 w.p.1. Equation 106 is recovered by plugging Eqs. 111 and 109 into Eq. 107 and defining $d_m = \gamma_{\text{dis}}\, n_a/\beta_m$. Finally, replacing Eq. 106 into Eq. 105, we obtain the desired bound:

$$\|\mathrm{E}\{F_m^{(k)}(\cdot)|P_m\}\|_{\mathrm{W}} \leq \frac{2\gamma_{\text{dis}}}{1-\gamma_{\text{dis}}}\|\Delta_m(\cdot)\|_{\mathrm{W}} + c_m \quad \forall k, \quad (112)$$

where $c_m \equiv \frac{1}{1-\gamma_{\text{dis}}} d_m$ also converges to 0 w.p.1.

Equation 112 proves the contraction property for the case where the policy is not updated during episodes. Below, we discuss the general case, where the policy may change every time step. As we will see, the only difference with respect to the case discussed above is that an additional term must be added to the right-hand side of Eq. 112. Since this additional term converges to 0 w.p.1, the contraction property also holds in that case.

F.2.2 Policy update at every time step

When online updates are considered, the policy might change every time step. Therefore, the probabilities in Eq. 103 no longer depend exclusively on P_m but rather also on the rewards received by the agent during the episode. However, since most of these probabilities are taken as common factors and upper bounded by 1, one can verify that most of the previous derivations still hold in the case where policies are changed every time step. In fact, the only point where the previous derivations have to be generalized is in Eq. 107. A time-dependent generalization of $g_m(s)$ can be defined by:

$$g_m'^{(t)}(s) = \sum_a \pi_m^{(t)}(a|s)\tilde{h}_m(s,a) - \max_b\{\tilde{h}_m(s,b)\}, \quad (113)$$

where t could be any time step in the interval $\mathcal{I}_m = [T_m + 1, T_{m+1}]$: i.e., the interval of time steps between the beginning and the end of episode m, and the policy is now given by:

$$\pi_m^{(t)}(a \mid s) = \frac{\exp[\beta_m \tilde{h}^{(t)}(s,a)]}{\sum_{b\in\mathcal{A}} \exp[\beta_m \tilde{h}^{(t)}(s,b)]}. \quad (114)$$

It is easy to verify that, similarly as in Eq. 111, if $\forall t \in \mathcal{I}_m$, $g_m'^{(t)}(s)$ is upper bounded by a sequence converging to 0, Eq. 106 also holds for time-dependent policies and hence Eq. 112 too.

Note that the only difference between Eqs. 110 and 114 is that in the former, the policy depends on the $\tilde{h}_m$ values while in the latter on the $\tilde{h}^{(t)}$, with $t \in \mathcal{I}_m$. The difference between these two, however, tends to 0 w.p.1 as the number of episodes increase. Given that rewards are bounded in the sense that $R^{(t)} \leq \mathcal{B}_R$, $\forall t$ for certain constant $\mathcal{B}_R$, we have that:

$$|\tilde{h}_t(\boldsymbol{e}) - \tilde{h}_m(\boldsymbol{e})| \leq \frac{1}{N_m+1}\left[|\tilde{h}_m(\boldsymbol{e})| + \frac{\mathcal{B}_R}{1-\gamma_{\text{dis}}}\right] \leq \frac{C}{N_m+1}, \quad (115)$$

where C is a constant. Since $N_m \to \infty$ w.p.1, the previous difference converges to 0.

To exploit Eq. 115, we bound $|g_m^{(t)}(s)|$ in the following way:

$$\begin{aligned} |g_m^{(t)}(s)| \leq & |\max_a\{\tilde{h}_m(s,a)\} - \max_a\{\tilde{h}^{(t)}(s,a)\}| \\ & + |\max_a\{\tilde{h}^{(t)}(s,a)\} - \sum_a \pi^{(t)}(a|s)\tilde{h}^{(t)}(s,a)| \\ & + \sum_a \pi^{(t)}(a|s)|\tilde{h}^{(t)}(s,a) - \tilde{h}_m(s,a)|. \end{aligned} \quad (116)$$

It follows from Eq. 115 that the first and third lines in the equation above are each upper bounded by $C/(N_m+1)$. Furthermore, the second term can be upper bounded by n_a/β_m in the exact same way as in Eq. 111. Thus,

$$|g_m'^{(t)}| \leq \frac{n_a}{\beta_m} + 2\frac{C}{N_m+1}, \quad (117)$$

which converges to 0 w.p.1. This implies that Eq. 107 also holds in the general case where the policy is updated every time step. This completes the proof of Eq. 112.

Appendix G: GLIE policy

In this appendix, we consider a policy given by the probabilities:

$$\Pr_m^{(t)}(a \mid s) = \frac{\exp[\beta_m \tilde{h}^{(t)}(s,a)]}{\sum_{b\in\mathcal{A}} \exp[\beta_m \tilde{h}^{(t)}(s,b)]}, \quad (118)$$

and derive conditions on the coefficients β_m in order to guarantee that this policy is GLIE on the $\tilde{h}$-values. We closely follow the derivation provided in Singh et al. (2000), and therefore will omit most of the details. Here, however, we focus on episodic tasks, in contrast to the continuous time tasks considered in Singh et al. (2000). Moreover, the coefficients β_m will depend exclusively on the episode index m, instead of the state s. In this way, we preserve a local model in the sense defined in the main text, where the β coefficient can be updated using exclusively environmental signals (in this case, the end of an episode).

Letting $\beta_m \to \infty$ is enough to guarantee that the policy is greedy in the limit. However, the speed at which β_m grows as a function of m must be upper bounded in order to additionally guarantee infinite exploration. In the following, we derive such a bound.

Let us denote as $\Pr_m(s,a)$ the probability that during episode m the state-action pair (s,a) is visited at least once. Hence, infinite exploration occurs if $\forall s, a$

$$\sum_{m=1}^{\infty} \Pr_m(s,a) = \infty. \tag{119}$$

Considering that $\sum_{m=1}^{\infty} c/m = \infty$ for any constant c, as a consequence of the Borel Cantelli lemma we have that a sufficient condition for Eq. 119 is given by:

$$\Pr_m(s,a) \geq c/m, \tag{120}$$

for some constant c. Therefore we would like to pick a bound on β_m in such a way that Eq. 120 is satisfied.

Let us denote by $\Pr_m(s)$ the probability that during episode m state s is visited at least once and let:

$$p_{\min}(m) = \min_{a,s,t\in\mathcal{I}_m} \left\{ \Pr_m^{(t)}(a|s) \right\} \tag{121}$$

be the minimum probability to take any action at any time step during the mth episode. It follows that:

$$\Pr_m(s,a) \geq \Pr_m(s)\, p_{\min}(m). \tag{122}$$

The first factor can in turn be bounded by a function of $p_{\min}(m)$ by noting the following. In a communicating MDP, any state can be reached from any other non-terminal state with nonzero probability. That means that independently of the initial state of the mth episode, there exists a sequence of actions that lead to any state s with some nonzero probability p_0. Such probability is constant and it is given by a product of transition probabilities of the MDP. In the worst case scenario, it could happen that s can only be reached by taking a specific sequence of actions that leads the agent to visit all nonterminal states before reaching s. Hence, $\Pr_m(s)$ can be bounded by the product of probabilities to pick those actions weighted by the transition probabilities of such a sequence. Given that the probability to take any action is in turn lower bounded by $p_{\min}(m)$, we conclude that:

$$\Pr_m(s) \geq p_0 \left[p_{\min}(m)\right]^{n_s - 1}, \tag{123}$$

where $n_s = ||\mathcal{S}|| - ||\mathcal{S}_T||$ is the number of nonterminal states.

In order to bound $p_{\min}(m)$ note that:

$$\begin{aligned} \Pr_m^{(t)}(a|s) &\geq \frac{1}{n_a} \exp\left[-\max_{a,b} \left\{ \tilde{h}^{(t)}(a|s) - \tilde{h}^{(t)}(b|s) \right\} \beta_m \right] \\ &\geq \frac{1}{n_a} \exp\left[-2\mathcal{B}_{\tilde{h}} \beta_m \right], \end{aligned} \tag{124}$$

where $\mathcal{B}_{\tilde{h}} \geq \tilde{h}^{(t)}(s,a), \forall s, a$ is an upper bound that exists because the rewards are bounded. Hence, it follows that $p_{\min}(m) \geq \exp\left[-2\mathcal{B}_{\tilde{h}}\beta_m\right]/n_a$. Replacing this inequality in Eq. 123 and using Eq. 122, we get that:

$$\begin{aligned} \Pr_m(s,a) &\geq p_0\, p_{\min}^{n_s}(m) \\ &\geq \frac{p_0}{n_a^{n_s}} \exp\left[-2n_s \mathcal{B}_{\tilde{h}} \beta_m \right]. \end{aligned} \tag{125}$$

Therefore, by choosing β_m in such a way that:

$$\beta_m \leq \frac{1}{2n_s \mathcal{B}_{\tilde{h}}} \ln(m), \tag{126}$$

Equation 120 holds; and thus the policy is guaranteed to preserve infinite exploration of all state-action pairs.

References

Bennett CH, DiVincenzo DP (1995) Towards an engineering era? Nature 377:389–390

Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S (2016) Quantum machine learning 549:11

Briegel HJ (2012) On creative machines and the physical origins of freedom. Sci Rep 2:522

Briegel HJ, las Cuevas GD (2012) Projective simulation for artificial intelligence. Sci Rep 2:400

Clausen J, Briegel HJ (2018) Quantum machine learning with glow for episodic tasks and decision games. Phys Rev A 97:022303

Dayan P, Sejnowski TJ (1994) TD (λ) converges with probability 1. Mach Learn 14(3):295–301

Dunjko V, Briegel H (2018) Machine learning & artificial intelligence in the quantum domain: a review of recent progress. Rep Prog Phys 81:7

Dunjko V, Taylor JM, Briegel HJ (2016) Quantum-enhanced machine learning. Phys Rev Lett 117:130501

Dvoretzky A et al (1956) On stochastic approximation

Hangl S, Ugur E, Szedmak S, Piater J (2016) Robotic playing for hierarchical complex skill learning. In: Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., pp 2799–2804. https://doi.org/10.1109/IROS.2016.7759434

Hangl S, Dunjko V, Briegel HJ, Piater J (2020) Skill learning by autonomous robotic playing using active learning and exploratory behavior composition. Frontiers in Robotics and AI 7:42. https://doi.org/10.3389/frobt.2020.00042. https://www.frontiersin.org/article/10.3389/frobt.2020.00042

Jaakkola T, Jordan MI, Singh SP (1994) Convergence of stochastic iterative dynamic programming algorithms. In: Advances in neural information processing systems, pp 703–710

Makmal A, Melnikov AA, Dunjko V, Briegel HJ (2016) Meta-learning within projective simulation. IEEE Access 4:2110

Mautner J, Makmal A, Manzano D, Tiersch M, Briegel HJ (2015) Projective simulation for classical learning agents: A comprehensive investigation. New Gener Comput 33:69

Melnikov AA, Makmal A, Briegel HJ (2018) Benchmarking projective simulation in navigation problems. IEEE Access 6:64639–64648

Melnikov AA, Makmal A, Dunjko V, Briegel HJ (2017) Projective simulation with generalization. Sci Rep 7:14430

Melnikov AA, Poulsen Nautrup H, Krenn M, Dunjko V, Tiersch M, Zeilinger A, Briegel HJ (2018) Active learning machine learns to create new quantum experiments. Proc Natl Acad Sci U.S.A 115:1221

Nielsen MA, Chuang IL (2000) Quantum computation and quantum information. Cambridge University Press, Cambridge

Nautrup HP, Delfosse N, Dunjko V, Briegel HJ, Friis N (2019) Optimizing quantum error correction codes with reinforcement learning. Quantum 3:215. https://doi.org/10.22331/q-2019-12-16-215

Paparo G, Dunjko V, Makmal A, Martin-Delgado MA, Briegel HJ (2014) Quantum speed-up for active learning agents. Phys Rev X 4:031002

Schuld M, Sinayskiy I, Petruccione F (2014) The quest for a quantum neural network. Quantum Inf Process 13:2567–2586

Singh S, Jaakkola T, Littman ML, Szepesvári C (2000) Convergence results for single-step on-policy reinforcement-learning algorithms. Mach Learn 38(3):287–308

Sriarunothai T, Wölk S, Giri GS, Friis N, Dunjko V, Briegel HJ, Wunderlich C (2017) Speeding-up the decision making of a learning agent using an ion trap quantum processor. arXiv:1709.01366

Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction, 2nd edn. MIT Press, Cambridge, MA

Watkins CJCH, Dayan P (1992) Q-learning. Machine learning 8(3-4):279–292