Quantum Machine Intelligence (2026) 8:58
https://doi.org/10.1007/s42484-026-00383-8

RESEARCH ARTICLE

Quantum reinforcement learning in dynamic environments

Oliver Sefrin[1] · **Manuel Radons**[2] · **Lars Simon**[2] · **Sabine Wölk**[1,3]

Received: 2 July 2025 / Accepted: 30 March 2026 / Published online: 12 May 2026
© The Author(s) 2026

Abstract
Combining quantum computing techniques in the form of amplitude amplification with classical reinforcement learning has led to the so-called "hybrid agent for quantum-accessible reinforcement learning", which achieves a quadratic speedup in sample complexity for certain learning problems. So far, this hybrid agent has only been applied to stationary learning problems, that is, learning problems without any time dependency within components of the Markov decision process. In this work, we investigate the applicability of the hybrid agent to dynamic RL environments and thus to more realistic scenarios. To this end, we enhance the hybrid agent by introducing a dissipation mechanism and, with the resulting learning agent, perform an empirical comparison with a classical RL agent in an RL environment with a time-dependent reward function. Our findings suggest that the modified hybrid agent can adapt its behavior to changes in the environment quickly, leading to a higher average success probability compared to its classical counterpart.

Keywords Quantum reinforcement learning · Hybrid algorithm · Continual reinforcement learning · Amplitude amplification

1 Introduction

From the success of convolutional neural networks for computer vision tasks in the early 2010s (Cireşan et al. 2011; Krizhevsky et al. 2017) to the nowadays ever-expanding usage of generative artificial intelligence based on diffusion (Sohl-Dickstein et al. 2015) or transformer (Vaswani et al. 2017) models, the vast field of machine learning (ML) has impacted many areas of modern life. Besides the continuous development of model architectures and training methods, the rapid improvement of specialized computing hardware in the form of graphics or tensor processing units (GPUs/TPUs) has led to today's prevalence of ML. This work focuses on reinforcement learning (RL), the ML subcategory encompassing interaction-based learning. While best known in the area of games with algorithmic breakthroughs such as deep Q-Learning (DQN) for Atari games (Mnih et al. 2015) or the MuZero algorithm (Schrittwieser et al. 2020), RL also serves other areas such as the training of the above mentioned language models through reinforcement learning from human feedback (RLHF) (Ziegler et al. 2020).

The combination of RL with quantum computing, which promises polynomial (Grover 1997; Hamann and Wölk 2022) or even exponential (Shor 1994; Dunjko et al. 2018; Liu et al. 2021; Molteni et al. 2024) speedups in specific problem tasks, has opened up the research field of quantum reinforcement learning (QRL) (Dong et al. 2008; Dunjko et al. 2016; Jerbi et al. 2021). Due to the restrictions of current quantum hardware in the so-called noisy, intermediate-scale quantum (NISQ) (Preskill 2018) era, however, the search for "quantum advantage" remains challenging (Schuld and Killoran 2022; Cerezo et al. 2025; Ciliberto et al. 2020). Here, the so-called "hybrid agent for quantum-accessible reinforcement learning" (Hamann and Wölk 2022), or briefly hybrid learning agent, provides an interesting outlook for the post-NISQ era: by incorporating amplitude amplification into the process of sampling action sequences, it allows to find rewarded sequences quadratically faster for specific RL environment types.

✉ Oliver Sefrin
oliver.sefrin@dlr.de

[1] Institute of Quantum Technologies, German Aerospace Center (DLR), Wilhelm-Runge-Straße 10, 89081 Ulm, Germany

[2] Bundesdruckerei GmbH, Kommandantenstraße 18, 10969 Berlin, Germany

[3] Center for Integrated Quantum Science and Technology (IQST), Ulm University, 89081 Ulm, Germany

This hybrid learning agent, however, has only been used in stationary environments so far, i.e., environments which do not undergo changes throughout the training process. Yet, many RL problems, especially regarding real-world applications (e.g., traffic control), do possess some kind of time-dependent dynamics (e.g., closed lanes due to road maintenance or temporary differences in traffic volume due to daily commuters). Classical RL features, in general, more refined algorithms and targeted research for techniques to compensate for such dynamics (Padakandla 2021; Ring 1994; Abel et al. 2023; Briegel and De las Cuevas 2012). QRL, on the other hand, is still in its infancy. Thus, to be able to challenge classical RL in real-world applications, it needs to be developed further by, e.g., also including corresponding techniques.

In such dynamic learning scenarios, a well-trained agent often performs worse after a sudden change of the environment compared to an untrained or a barely trained agent. As a consequence, a fast learning quantum agent often has a disadvantage compared to a slower learning classical agent when the environment changes. To compensate for this disadvantage and outperform classical agents again, the hybrid agent needs an efficient procedure to unlearn or relearn. Therefore, in this work, we take a first step by introducing a dissipation mechanism in the hybrid learning agent and by investigating its applicability and efficiency in such non-stationary environments. In this way, the hybrid learning agent becomes more robust to changes of the problem environment without the necessity to introduce complex mechanisms for explicitly detecting anomalies. To the best of our knowledge, this is the first work on QRL specifically targeting non-stationary environments.

Specifically, we compare the hybrid learning agent against a classical counterpart in the so-called Gridworld or maze environment, where we add time dependency to the reward function both on the intra- and inter-episode level. Our research questions are the following:

- Which modifications have to be applied to the original hybrid learning agent as defined by Hamann and Wölk (2022) to achieve high-performance learning in these kinds of non-stationary environments and how do these modifications compare to classical techniques for dynamic environments?
- Assuming a better learning performance on a first reward function, can the hybrid learning agent overcome the resulting worse initial success probability after a change of reward function?
- Does the less frequent updating of the agent's policy due to the episodes spent on amplitude amplification hinder the hybrid learning agent's capability to adapt in the dynamic environment?

This article is structured as follows: in Section 2, we recapitulate RL theory and the initial hybrid learning agent and summarize related work on QRL and continual RL. Section 3 features a description of the basic Gridworld problem as well as the modified environment options with non-stationary reward functions. In Section 4, we present our modifications of the hybrid learning agent for the non-stationary RL problems and discuss the methodology of our numerical simulations. Sections 5 and 6 comprise our simulation results and the discussion thereof. Finally in Section 7, we provide a conclusion and an outlook to future research.

2 Background

2.1 Reinforcement learning

Reinforcement Learning (RL) (Kaelbling et al. 1996; Sutton and Barto 2018) is a branch of machine learning focused on learning through interaction. Usually, this interaction takes place in a sequence of discrete time steps $t \in \mathbb{N}_0$ and is formalized as a *Markov Decision Process* (MDP) (Puterman 1994). The MDP is a five-tuple

$$M = \langle T, \mathcal{S}, \mathcal{A}, p_t(\cdot|s,a), r_t(s,a) \rangle. \tag{1}$$

T refers to the total amount of interaction steps and can be finite (in *episodic* tasks) or infinite (in *continuing* tasks). $\mathcal{S}$ and $\mathcal{A}$ denote the sets of states and actions, respectively. The *transition probability function*

$$p_t : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1] \tag{2}$$

denotes the probability $p_t(s'|s,a)$ that upon playing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ at time t, the state at time $t+1$ will be $s' \in \mathcal{S}$. Associated with this new state is a real-valued, bounded reward R_{t+1}, with the value determined by the *reward function*

$$r_t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}. \tag{3}$$

After playing action a in state s at time t, the agent receives the reward $R_{t+1} := r_t(s,a)$.

The agent's behavior is described by its *policy*

$$\pi : \mathcal{A} \times \mathcal{S} \to [0,1],$$

which describes the probability $\pi(a|s)$ to choose action a upon perceiving state s. The agent's goal is to adapt its policy such as to maximize the expected *cumulative reward*

$$G_t = \sum_{k=t+1}^{T} \chi^{k-t-1} R_k \tag{4}$$

at any time step t, where $\chi \in [0, 1]$ is a discount factor. In continuing tasks, setting $\chi < 1$ ensures that the cumulative reward does not diverge. In episodic tasks, the interaction terminates once the final time step $t = T$ is reached, upon which the interaction starts again at $t = 0$ from an initial state s_0.

The environment can be classified as *stationary* (*non-stationary/dynamical*) if the transition dynamics of the MDP are time-independent (time-dependent). In the stationary case, we may omit the index t from the transition probability function (Eq. (2)) and the reward function (Eq. (3)). RL in non-stationary environments can be further distinguished according to characteristics of the time dependency and the amount of information given to the agent about changes in the environment (Padakandla 2021). Relevant to this work is *continual RL* as defined by Ring (1994) and Abel et al. (2023), which describes an ongoing learning process in non-stationary environments where the "best agents never stop learning" (Abel et al. 2023). An example for this is an ensemble of MDPs from which an MDP is randomly sampled repeatedly after certain amounts of time steps, such that an agent must constantly adapt to the current MDP.

The speedup of the hybrid learning agent, as defined in the following section, has been proven for a special class of RL tasks, so-called *deterministic and strictly episodic* (DSE) tasks (or environments). "Strictly episodic" signifies that the episode length T is not only finite, but also has a fixed value for all episodes. "Deterministic" refers to the transition probability function of Eq. (2), which, in the deterministic case, returns for any state-action tuple (s, a) unity for one specific state s' and zero for all other states. Thus, we can define the deterministic version of p_t,

$$p_t^{(\mathrm{det})} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}, \tag{5}$$

which directly projects to the next state s'.

2.2 Hybrid learning agent

The "hybrid agent for quantum-accessible reinforcement learning" (Hamann and Wölk 2022), which we further refer to as the hybrid learning agent, is a hybrid quantum-classical algorithm that can be used in combination with classical RL algorithms to provide a quasi quadratic speedup in sample complexity. In brief, in its quantum component, the hybrid learning agent uses amplitude amplification (AA) to increase the probability to sample rewarded action sequences. This sampled action sequence is subsequently tested in an episode of classical interaction to determine the sequence of encountered percepts and the reward information, which can then be used to update the agent's policy.

In more detail, the algorithm operates in a quantum communication scenario between the agent and the environment. First, assuming finite action, percept, and reward spaces $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{R} \subset \mathbb{R}$, we use the following encoding:

- all actions $a \in \mathcal{A}$ are encoded in a set of orthonormal quantum states $|a\rangle_A$ in Hilbert space $\mathcal{H}_A$,
- all percepts $s \in \mathcal{S}$ are encoded in a set of orthonormal states $|s\rangle_S$ in Hilbert space $\mathcal{H}_S$,
- and all rewards $r \in \mathcal{R}$ are encoded in a set of orthonormal states $|r\rangle_R$ in Hilbert space $\mathcal{H}_R$.

The agent sends quantum states $|a\rangle_A$ to the environment and receives quantum states $|s\rangle_S$ and $|r\rangle_R$, accordingly. From here on, we consider DSE environments with a binary episode reward r. That is, exactly one interaction step in an episode may receive a reward of one; re-visiting the rewarded percept later in the episode does not yield another reward.[1] Further, we use vector notation $\vec{a} = (a_0, a_1, \ldots, a_{T-1})$ to denote a sequence of actions played in an episode or $\vec{s} = (s_0, s_1, \ldots, s_T)$ for the sequence of percepts encountered in an episode. As a short-hand notation for the sequence of encountered percepts and the reward outcome of playing action sequence $\vec{a}$ after the initial state s_0, we write $\vec{s}(s_0, \vec{a})$ and $r(s_0, \vec{a}) \in \{0, 1\}$, respectively. Lastly, let

$$|\vec{s}_{\mathrm{init}}\rangle_S = |(s_0, \varnothing, \ldots, \varnothing)\rangle_S \in \mathcal{H}_S^{\otimes(T+1)} \tag{6}$$

denote the initial percept quantum state before the interaction with a fiducial state $|\varnothing\rangle_S$.

We can model the response of the environment to an episode-long sequence of actions $\vec{a}$ as a unitary U_{env} acting on the multipartite state $|\vec{a}\rangle_A|\vec{s}_{\mathrm{init}}\rangle_S|0\rangle_R$ (Dunjko et al. 2016):

$$\begin{aligned} &U_{\mathrm{env}}|\vec{a}\rangle_A|\vec{s}_{\mathrm{init}}\rangle_S|0\rangle_R \\ =&|\vec{a}\rangle_A|\vec{s}(s_0, \vec{a})\rangle_S|r(s_0, \vec{a})\rangle_R. \end{aligned} \tag{7}$$

As shown by Dunjko et al. (2016), with one query of U_{env} for simple environments with action-independent percepts (*memoryless* environments) or two queries for regular DSE environments, one can create a phase-kickback oracle O_{env}:

$$\begin{aligned} &O_{\mathrm{env}}|\vec{a}\rangle_A|\vec{s}_{\mathrm{init}}\rangle_S|-\rangle_R \\ =&\ (-1)^{r(s_0,\vec{a})}|\vec{a}\rangle_A|\vec{s}_{\mathrm{init}}\rangle_S|-\rangle_R. \end{aligned} \tag{8}$$

Using the current policy π and, e.g., a mapping mechanism (see Appendix B), one can compute the probability to choose

[1] Strictly speaking, this *first-visit reward* violates the Markov property. Adding a first-visit flag to each percept remedies this formal violation.

action sequence $\vec{a}$ under said policy, which we denote $\hat{\pi}(\vec{a})$. This allows preparing the weighted superposition[2]

$$|\psi\rangle = \sum_{\vec{a}\in\mathcal{A}^{\otimes T}} \sqrt{\hat{\pi}(\vec{a})}|\vec{a}\rangle_A|\vec{s}_{\text{init}}\rangle_S|-\rangle_R \tag{9}$$

and defining the diffusion operator

$$D = \mathbb{I} - 2|\psi\rangle\langle\psi|. \tag{10}$$

With the Grover operator $G = DO_{\text{env}}$ we may thus perform AA (Grover 1997; Brassard et al. 2002). By denoting the current classical success probability

$$Q = \sum_{\{\vec{a}|r(\vec{a})=1\}} \hat{\pi}(\vec{a}), \tag{11}$$

we arrive at the probability of measuring a rewarded action sequence after k iterations of AA,

$$p_{AA}(Q,k) = \sin^2\left([2k+1]\arcsin(\sqrt{Q})\right). \tag{12}$$

The measurement gives a candidate action sequence $\vec{a}$, but reveals no percept or reward information due to the necessary uncomputation. Hence, an episode of classical play following the quantum part is required to determine $\vec{s}(\vec{a})$ and $r(\vec{a})$. Consecutively, the learning agent uses this information to update its policy according to some update mechanism as specified, e.g., in Projective Simulation (PS) (Briegel and De las Cuevas 2012), Q-Learning (Watkins and Dayan 1992), SARSA (Rummery and Niranjan 1994), REINFORCE (Williams 1992), or quantum natural policy gradient (Meyer et al. 2023).

Since the current success probability Q is typically not known in RL, the hybrid learning agent uses the AA variation for an unknown number of solutions presented by Boyer et al. (1998). This requires a lower bound estimate $Q_{\min}$ of Q.

The full hybrid learning agent in a version agnostic to the specific policy update mechanism is presented in Algorithm 1. In this notation, the policy update mechanism is specified in the subroutine `policy_update` (see Algorithm 2 in Appendix B for an examplary mechanism using PS), which uses the interaction history of an episode to update the current policy. Here, we update the policy just on positive feedback (i.e., if the chosen action sequence is rewarded), which is typical for RL with the PS algorithm. However, an update with every type of feedback after each iteration is possible by simply moving lines 21 and 22 of the algorithm into the while-loop and adding the reward information to the arguments of `policy_update`.

Algorithm 1 Hybrid learning agent of Hamann and Wölk (2022)

Require: MDP $\langle T, \mathcal{S}, \mathcal{A}, p_t^{(\text{det})}(s,a), r_t(s,a)\rangle$,
initial percept s_0,
Grover operator G,
`policy_update` subroutine

1: initialize algorithm constant $\lambda \leftarrow 5/4$
2: initialize lower bound estimate $Q_{\min}$
3: algorithm parameter $m \leftarrow 1$
4: `rewarded` $\leftarrow$ `false`
5: **while** `not rewarded` **do**
6: ▷ *Quantum Part* ◁
7: $k \leftarrow$ random integer in $[0, \mathrm{m})$
8: prep. $|\psi\rangle \leftarrow \sum_{\vec{a}\in\mathcal{A}^{\otimes T}} \sqrt{\hat{\pi}(\vec{a})}\,|\vec{a}\rangle_A\,|\vec{s}_{\text{init}}\rangle_S\,|-\rangle_R$
9: amplitude amplification $|\psi'\rangle \leftarrow G^k\,|\psi\rangle$
10: $\vec{a}' \leftarrow$ **measure** $|\psi'\rangle$
11: ▷ *Classical Part* ◁
12: **for** $i = 1$ **to** T **do**
13: $s_i \leftarrow p_i^{(\text{det})}(s_{i-1}, a_{i-1})$
14: $r_i \leftarrow r_i(s_{i-1}, a_{i-1})$
15: **if** $r_i = 1$ **then**
16: `rewarded` $\leftarrow$ `true`
17: $\vec{s} \leftarrow (s_0, s_1, \ldots, s_i)$
18: `break`
19: $m \leftarrow \min\left(\lambda \cdot m,\ Q_{\min}^{-1/2}\right)$
20: ▷ *Classical RL Update* ◁
21: update π using subroutine `policy_update`$(\vec{a}', \vec{s})$
22: update estimate $Q_{\min}$
23: repeat from step 3

2.3 Related work

Continual RL is also investigated in classical machine learning and classical approaches can be further classified according to the type of non-stationarity and the learning goals present in the problem. Learners which aim to adapt to changes in the MDP but at the same time retain good performance on previous tasks and thus avoid catastrophic forgetting (McCloskey and Cohen 1989) typically employ techniques such as context detection (da Silva et al. 2006) or meta-learning (Finn et al. 2017). This is in contrast to our learning algorithm, which actively employs a forgetting

[2] Preparing the state by explicitly computing the sampling probability of each action sequence $\vec{a}$ quickly becomes infeasible for larger sequence lengths due to the exponential scaling of the number of possible sequences. We do note, however, that the former can be achieved efficiently, e.g., by using quantum policies based on variational quantum circuits.

mechanism to learn the current task more quickly. For an overview of continual RL, its subtopics, and relevant approaches, we refer the reader to Khetarpal et al. (2022).

The term *quantum* reinforcement learning (QRL) can be interpreted in different ways, depending on which components of the problem are quantum or classical. Here, we classify QRL in accordance with Schuld and Petruccione (2021) as using a quantum (or quantum-enhanced) information processing device to process classical data. However, a strong connection to the complementary domain of classical RL algorithms operating on quantum data exists as well. For example, classical RL is used in designing or optimizing quantum circuits (Fösel et al. 2021; Altmann et al. 2024) and in quantum error correction (Andreasson et al. 2019; Nautrup et al. 2019), but most prominently in quantum control (Bukov et al. 2018; Niu et al. 2019; Zhang et al. 2019; Sivak et al. 2022; Li et al. 2025). Within these research fields, the task of *reward design* (Singh et al. 2009), i.e., modeling the reward function such that an agent may efficiently converge towards the optimal behavior, has received much attention (Yu and Zhao 2024; Acampora et al. 2025; Giordano et al. 2025).

The predominant category of QRL in the current NISQ era combines classical RL with so-called variational quantum circuits (VQCs) (Cerezo et al. 2021). In these algorithms, a quantum circuit with trainable, parameterized gates takes the place of an artificial neural network as the estimator for, e.g., the state-value function in value-based RL such as the DQN algorithm (Mnih et al. 2015) or the policy in policy-gradient-based RL such as the PPO (Schulman et al. 2017) or DDPG algorithms (Lillicrap et al. 2019). The feasibility of VQC-powered QRL algorithms has been tested in Gymnasium (Towers et al. 2025) tasks (Chen et al. 2020; Lockwood and Si 2020b; Jerbi et al. 2021; Skolik et al. 2022), standard maze problems (Chen et al. 2024), Atari games (Lockwood and Si 2020a), and quantum state preparation problems (Wu et al. 2025). In general, QRL approaches based on VQCs are able to learn to solve quantum as well as classical problems and can be implemented on existing NISQ devices. However, if and how it is in general possible to generate quantum advantages in this way is an important question which is still under scientific examination (Gil-Fuster et al. 2025).

QRL approaches which do not use VQCs typically aim at generating speedups via quantum sub-routines such as quantum mean estimation (Ganguly et al. 2023; Zhong et al. 2024), quantum linear system solvers (Cherrat et al. 2023), or a combination of quantum phase estimation and Grover search (Wiedemann et al. 2023). Other methods involve a quantum random walk approach to a variant of the Gridworld problem (Dalla Pozza et al. 2022) or quantum sampling methods in energy-based RL (Jerbi et al. 2021). Usually, these approaches are harder to implement on real quantum computer hardware but exhibit a provable quantum advantage.

The speed-up in learning time of the hybrid learning agent has been proven (Hamann and Wölk 2022) and demonstrated in a proof-of-principle experiment using a nanophotonic processor (Saggio et al. 2021). Further research on the hybrid learning agent has focused on the effect of changing oracles during the learning process (Hamann et al. 2021) and on developing episode length selection strategies for not strictly episodic learning problems (Sefrin and Wölk 2025). Similar to the hybrid learning agent, a QRL algorithm using amplitude amplification based on quantum random walks has been developed (Paparo et al. 2014) and experimentally verified on an ion trap device (Sriarunothai et al. 2018).

3 Problem setting

The hybrid learning agent, as described in Section 2.2, provides a quasi quadratic speedup in sample complexity for RL tasks which have deterministic and strictly episodic (DSE) environments. A standard example for a DSE environment is the so-called *Gridworld* or maze problem, which is described in Section 3.1. So far, however, the hybrid learning agent has only been used for stationary environments. In this work, we investigate the applicability of the hybrid agent to non-stationary environments similar to the ones considered by Abel et al. (2023) and Briegel and De las Cuevas (2012). Two classes of such non-stationary environments, exemplified as variations of the textbook maze problem, are presented in Section 3.2.

3.1 Gridworld

The Gridworld problem, in its most common form, is an environment consisting of cell states in a two-dimensional grid. In an episode, the agent starts from a fixed starting position and tries to reach a specified target cell. The set of actions are the four basic move directions $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$, the set of states $\mathcal{S}$ consists of all cells in the grid. The response of the environment to choosing a move direction is deterministic and time-independent: the agent's new position s_{t+1} is always one step away in the move direction from the old position s_t. Some of the Gridworld's cells may consist of walls, however. In that case, if the agent were to move into a wall cell, it remains at its old position. The agent receives a reward of one for reaching the target cell for the first time in the episode and a reward of zero for any other outcome. Typically, the episode concludes as soon as the agent reaches the

target cell for the first time. This, however, results in a variable episode length, as the agent may not always require the same number of steps to reach the target while learning. By imposing a fixed number of interaction steps for the episode, the Gridworld environment can be turned into a strictly episodic one. We will use this setting throughout this paper in order to concentrate on effects caused by time dependency of the learning problem. However, the hybrid agent can also deal with learning problems with variable episode length as investigated in Sefrin and Wölk (2025).

As such, the Gridworld problem is an example for a stationary environment since both the transition probability function and the reward function are time-independent. This means that the agent does not need to adapt to any changes during the learning.

3.2 Non-stationary reward environments

In this work, we are interested in the hybrid learning agent's capability to adapt to RL tasks with changing, i.e., time-dependent rewards. Especially, we want to investigate scenarios in which the hybrid learning agent's increased learning speed with an initial reward distribution is detrimental to its successive performance in a varied reward setting. This can be the case, e.g., in RL tasks with an initial set of rewarded action sequences $\mathcal{A}_1$ which is, at some point, exchanged with a disjoint set of rewarded action sequences $\mathcal{A}_2$. In such a case, training the policy to sample an element of $\mathcal{A}_1$ with high probability directly lessens the probability to sample an element from $\mathcal{A}_2$.

As our testbed for these investigations, we modify the Gridworld problem with fixed episode length from Sec. 3.1. In particular, we first introduce as a first step an explicit time-dependent reward by replacing the fixed target with a moving target. That is, the rewarded state follows a fixed route and moves one step along that route synchronously to each step of the agent. As in the standard Gridworld setting, the agent is rewarded upon encountering the rewarded state for the first time within the episode. If the agent has not found the rewarded state at the end of the route, the episode ends without a reward and both the agent and the reward are reset to their initial cells. Thus, the reward route length L determines the episode length. Sometimes, it might be disadvantageous for the agent to move from its current state because the reward state would move onto its current position. To accommodate for this fact, we expand the set of basic actions by the non-action, i.e., $\mathcal{A} = \{\text{left}, \text{right}, \text{up}, \text{down}, \text{do nothing}\}$.

In a second step, we change the path of the moving target at a certain point during the learning process. This leads to the two learning scenarios which we investigate in this article:

A Fixed Reward Path This is the basic moving target scenario as introduced in the first step above. Here, the non-stationarity is purely restricted to the reward response within the episode. Across episodes, the reward response remains the same. A depiction of this Gridworld layout is given in Fig. 1. We assume that the agent has knowledge about the fact that the reward path is fixed, rendering a forgetting mechanism obsolete in this scenario. Using the hybrid learning agent in its standard form, we can use this scenario to verify its speedup compared to its purely classical counterpart.

B Changing Reward Path In this scenario, we start in the same fashion as in scenario A. However, after some time, we exchange the previous reward path for a new one with the same length L. This adds non-stationarity on the inter-episode level, as the reward response additionally changes after some amount of episodes. Therefore, this type of RL environment can be categorized as a hidden-mode Markov decision process (HM-MDP) (Choi et al. 1999), where the different reward paths correspond to the different modes of the HM-MDP. (A similar problem of newly generated Gridworld environments after a certain number of steps was introduced in Luketina et al. (2022) as "Switching MDPs".) The agent knows that a reward path switch may happen, but it is not notified about when it happens. Thus, using a forgetting mechanism is necessary in this scenario. In this example problem, we use two different reward paths and switch from the first to the second path upon triggering a pre-defined criterion. With this scenario, we challenge the hybrid learning agent's capability to relearn given a new reward situation.

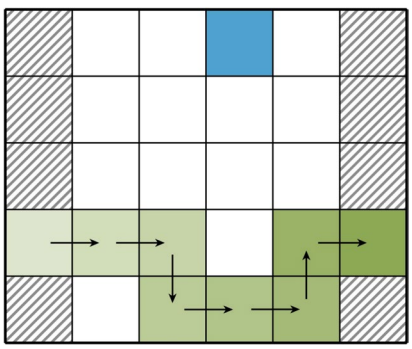


Fig. 1 Graphical depiction of Gridworld layout A, which is used for the fixed reward path scenario. The blue cell represents the agent's starting position, green cells indicate the reward path, which direction is indicated by arrows. Gray-shaded cells are impassable for the agent

4 Methods

In general, there are numerous methods to deal with changes in a learning problem. One option is to actively search for changes and then, e.g., reset the learning agent to its untrained starting status. This might be a good method if extensive changes are expected. More commonly, however, changes will be seldom and only of minor character. In such cases, a learning agent which is able to gradually adapt its behavior is more suited. Such a learning agent will be in general more robust since it neither sticks to a learned behavior nor inclines to sudden behavior changes. The dissipation mechanism of Projective Simulation (PS) (Briegel and De las Cuevas 2012) enables such a gradual adaptation. In Section 4.1, we introduce several modifications of our hybrid learning agent in order to use the dissipation mechanism of PS and capture changes in the MDP reasonably quickly. Subsequently, in Section 4.2, we present the simulation setup used to compare the adapted hybrid learning agent versus a purely classical RL algorithm in non-stationary environments.

4.1 Forgetting mechanism for the hybrid learning agent

Standard RL algorithms such as Q-Learning or SARSA are guaranteed to converge to the optimal behavior in stationary environments in the infinite time limit given sufficient exploration (infinite visits of all state-action pairs) and with a policy that converges to the greedy policy in that limit (Sutton and Barto 2018). In non-stationary environments, however, these algorithms tend to capture changes in the MDP dynamics poorly (Pieters and Wiering 2016; Padakandla 2021). Using Q-Learning in continual RL, for example, requires a policy which enables exploration even in the infinite time limit such as an ϵ-greedy policy with non-vanishing ϵ (Abel et al. 2023).

In this work, we pair the hybrid learning agent with a policy-update mechanism based on PS, which is a tabular, state-action-value based RL algorithm. In PS, a dissipation mechanism can be used to let the agent effectively adapt to changes in the environment and to enable more stable learning. This dissipation drives the policy towards a uniform distribution – therefore enhancing exploration – with the strength of the dissipation governed by a parameter $\gamma \in [0, 1]$. For an overview of PS, the dissipation mechanism, and its relevant extensions to our work, see Appendix B.

To adapt the hybrid learning agent to non-stationary reward environments, we make two modifications. The first modification targets the lower frequency of policy updates of the hybrid learning agent: To execute k iterations of amplitude amplification, the hybrid learning agent requires $2k + 1$ RL episodes. Thus, it can only perform a policy update after $2k + 1$ episodes, whereas a classical agent may update its policy after every episode. To ensure that the hybrid learning agent receives the same amount of dissipation than a classical agent, we increase the dissipation strength according to the number of "missed" update episodes. The adapted update rule for the hybrid learning agent with a equivalency proof is shown in Appendix B.1. The second modification is related to the lower bound estimate $Q_{\min}$ that the hybrid learning agent requires (cf. line 19 of Algorithm 1). In general, this estimation can be performed in many different ways, such as (i) determining an absolute lower bound $Q_{\min}$ based solely on general problem settings, (ii) estimating $Q_{\min}$ based on observed rewards, or (iii) estimating $Q_{\min}$ based on the agent's current policy and observed rewards. For our simulations, we used option (iii) as described in detail in Appendix C. To account for the possibility of changing rewards, we purge the set of found reward paths from the ones which are found to be no longer rewarded, which is further described in Appendix C.1.

4.2 Simulation methodology

We compare the performance on the Gridworld scenarios stated above between the hybrid learning agent paired with PS based policy updates (subsequently abbreviated as *hybrid PS*), classical PS, and classical Q-Learning.

For both classical and hybrid PS, we use the same PS specifications and hyperparameters. These are the softmax-based policy of Eq. (14) with a pseudo-temperature of $\beta = 1$, the glow extension with a fixed value of $\eta = 0.05$, and the ECM with mapping. As dissipation parameter values, we set $\gamma = 0$ for scenario A (i.e., turn the dissipation mechanism off) and test $\gamma \in \{0.01, 0.02, 0.05, 0.1\}$ for scenario B. Further details of how the results of the hybrid learning agent can be generated despite the infeasibility to run such a large quantum circuit are given in Appendix D.

For Q-Learning, we use an ϵ-greedy policy with a fixed step-size parameter $\alpha = 0.5$, which is identical to the *continual* Q-Learning algorithm used in the "Switching MDPs" example of Abel et al. (2023). All q-values are initially set to zero. Unlike the binary reward used in the PS policy updates, the agent receives a negative reward of -1 for each action until the goal is reached, which is in line with standard implementations of Q-Learning in Gridworld (cf. example 6.6 of Sutton and Barto (2018)). In scenario A, we initially set $\epsilon = 0.1$ and employ an exponential ϵ-decay with a decay factor of 0.99 applied after each episode. For scenario B, however, we fix $\epsilon = 0.05$ such that the Q-Learning agent continues exploring.

4.2.1 Gridworld layouts

To test scenario A, the fixed reward path scenario (cf. Section 3.2), we use the Gridworld layout shown in Fig. 1, which we denote layout A. With an episode length of seven, 5^7 unique action sequences may be chosen by the agent. 1330 of these action sequences have a rewarded outcome, such that the initial success probability with a uniform policy is roughly 1.7%.

For the changing reward path scenario, or scenario B, we use a different Gridworld layout, which is presented in Fig. 2 and hence named layout B. Here, we have two reward paths: The initial reward path goes down and then right, with an episode length of seven. After the switch happens, the second reward path is now active instead of the initial one. This path is mirrored to the first path along the top-left to bottom-right diagonal. The agent's starting position is on that diagonal, such that the number of rewarded action sequences is equal for both reward paths. Further, in this layout, the sets of rewarded action sequences of the two reward paths are fully disjoint. The key properties of both Gridworld layouts are summarized in Table 1.

4.2.2 Stopping criteria

In each scenario, we test two different settings for deciding how many episodes an agent may train with each reward path. These two stopping criteria allow comparing both the time required to reach a certain level of training as well as the performance after a fixed training time.

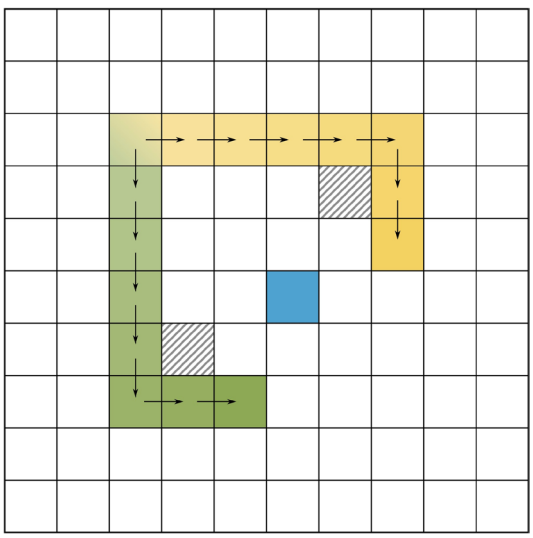


Fig. 2 Graphical depiction of Gridworld layout B, which is used for the changing reward path scenario. The blue cell represents the agent's starting position, gray-shaded cells are impassable for the agent. Green cells indicate the initial reward path, yellow cells represent the second reward path. The direction of either reward path is indicated by arrows

Table 1 Statistics of Gridworld layouts A and B

	Layout A	Layout B
Episode Length	7	7
Length of Shortest Reward Path	4	3
Initial Success Probability	0.017	0.041

1. "k out of n" successes With this setting, we implement a switching or stopping criterion based on the agent's recent performance. If it manages to get k rewards in the last n tries, we consider the agent to have learned and switch to the second reward path or end the training, respectively. In this case, the training time is the most relevant metric, as the different agents can be expected to have similar learning progress upon triggering the "k out of n" criterion.

2. Fixed number of episodes Here, we fix the number of interaction episodes in advance as N_1 and N_2 for the initial and (optional) second reward path. With the same amount of training time, we are now more interested in the agents' final success probability.

5 Results

5.1 Scenario A: Fixed reward path

We combine scenario A with both the "'k out of n' successes"-criterion, which we denote learning scenario A.1, and the "fixed number of episodes"-criterion, which we denote learning scenario A.2.

In the learning scenario A.1, we choose $k = 4$ and $n = 5$, such that training stops as soon as four out of the last five sampled action sequences are rewarded. The comparison of the number of played episodes to reach certain key events for the three different agents is shown in Table 2. The hybrid PS agent has the quickest average time to find the first reward as well as to reach the 20% threshold and to fulfill the stopping criterion. Classical PS, on the other hand, requires more than three times as many episodes as hybrid PS to reach the first reward. Between the 20% threshold and the completion of the learning scenario, the difference between classical and hybrid PS remains virtually the same

Table 2 Learning times in terms of episodes for the hybrid PS, classical PS, and Q-Learning in the fixed reward path scenario with layout A and a "4 out of 5" stopping criterion (scenario A.1). The results stem from simulations with $N = 1000$ runs per agent, the best value in each category is printed in boldface

	Classical		Hybrid
	PS	Q-Learning	PS
First Reward	58.4(1.8)	25.3(0.4)	**17.9(0.4)**
20% Success Prob.	84.6(1.9)	38.5(0.3)	**33.8(0.5)**
Completion	94.2(1.9)	108.1(0.3)	**44.5(0.5)**

with classical PS requiring roughly 50 episodes more time for each threshold. Classical Q-Learning lags behind hybrid PS by 7.4 episodes and 4.7 episodes for the first two key events, respectively, and thus requiring much less learning time than classical PS. However, it reaches the completion criterion last with roughly 108 episodes on average.

For learning scenario A.2 with a fixed number of interaction episodes, we set $N_1 = 250$ (and $N_2 = 0$, as we do not consider a switch of reward paths in scenario A). We show the learning progress measured in terms of the agents' success probabilities in Fig. 3. The hybrid PS agent's learning curve shows a quicker initial increase and already after 100 episodes, it has reached an average success probability of 99.3%. The classical PS agent reaches on average a success probability of 69.3% after 100 episodes. At the end of training after 250 episodes, the classical agent's success probability is 97.9%, thus, roughly as large as the hybrid agent's success probability after 75 episodes. For classical Q-Learning, the learning curve is less smooth, with a slower initial increase, but a rapid rise between episodes 80 to 120, reaching a final success probability of 97.1%.

5.2 Scenario B: Changing reward path

In the changing reward scenario, we again distinguish between the two stopping criteria into two learning scenarios, B.1 and B.2.

In scenario B.1, we use the "'k out of n' successes"-criterion with $k = 4$ and $n = 5$ both before and after the reward path switch. An overview of the learning times for each reward path as well as the total learning time for this scenario is given in Table 3. Regarding the PS agents and the first learning phase with the initial reward path, we can see that larger dissipation values consistently yield larger learning times for both agents. The hybrid PS agent achieves lower learning times compared to the classical PS agent for fixed values of γ. After the switch to the second reward path, hybrid PS still requires less learning time than classical PS across all tested dissipation values. Here, however, an intermediate value of $\gamma = 0.05$ results in the lowest learning time among all configurations. Classical Q-Learning features the second longest learning time for the first reward path across all configurations, but the shortest learning time for the second reward path. However, the hybrid PS agent with $\gamma = 0.05$ features the shortest aggregate learning time with roughly 113 episodes, followed by classical Q-Learning with 149 episodes. The classical PS agent with the least amount of total episodes, namely 187 episodes, is the one with the dissipation parameter set to $\gamma = 0.02$.

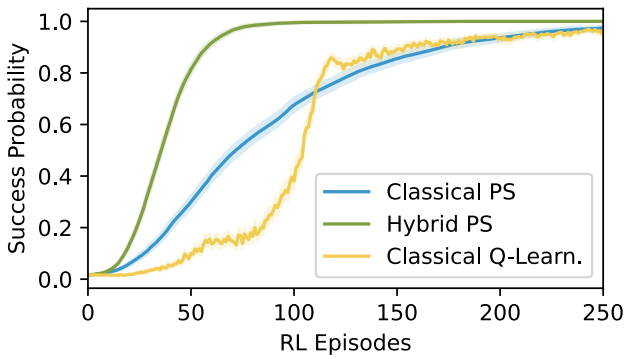


Fig. 3 Comparison of the learning progress between hybrid PS, classical PS, and classical Q-Learning for the fixed reward path scenario with layout A and $N_1 = 250$ training episodes (scenario A.2). Average values of $N = 1000$ runs, error bands represent the 95% confidence interval

Table 3 Results for the hybrid and classical PS algorithms with different dissipation values γ as well as classical Q-Learning in the changing reward path scenario with layout B and a "4 out of 5" stopping criterion for both reward path phases (scenario B.1). Values indicate the mean with its standard error across $N = 100$ runs per configuration. The lowest learning time of each category is printed in boldface

γ	Agent	Episodes before Switch	Episodes after Switch	Episodes total
0.01	Classical PS	57.2(2.7)	170.6(5.9)	227.8(7.2)
	Hybrid PS	**35.3(2.2)**	100.8(5.1)	136.1(6.1)
0.02	Classical PS	60.5(3.5)	126.7(4.2)	187.2(5.6)
	Hybrid PS	40.6(1.6)	84.1(3.4)	124.7(4.1)
0.05	Classical PS	108.1(8.0)	121.5(10.0)	229.6(12.8)
	Hybrid PS	50.0(2.6)	62.8(2.3)	**112.7(3.9)**
0.1	Classical PS	371.1(32.5)	380.8(33.8)	751.9(44.3)
	Hybrid PS	80.0(4.9)	85.1(5.1)	165.1(6.6)
—	Classical Q-Learning	112.3(0.8)	**37.0(1.3)**	149.3(1.3)

For scenario B.2, we set the interaction times for the first and second reward paths to $N_1 = 100$ and $N_2 = 300$, respectively. In Fig. 4 we give an overview of the learning curves for the three RL algorithms, where the dissipation value is fixed to $\gamma = 0.05$ for the two PS-based algorithms. A detailed comparison of classical and hybrid PS with all tested dissipation values is given in Fig. 7 of Appendix A. There, we also report in Table 4 the average success probabilities of all three algorithms for the different reward paths as well as in aggregate.

In the initial learning phase, hybrid PS is the only algorithm crossing 90% success probability, with classical PS reaching almost 60% and classical Q-Learning reaching 35% success probability. Immediately after the reward path switch, however, the success probability of hybrid PS is lower by roughly an order of magnitude compared to classical PS. For classical Q-Learning, interestingly, the success probability remains above the initial success probability of an uniform action selection. To illustrate this more clearly, Fig. 5 presents the learning curves of scenario B.2 on a logarithmic scale. Following the switch, the classical Q-Learning agent reaches first an average success probability above

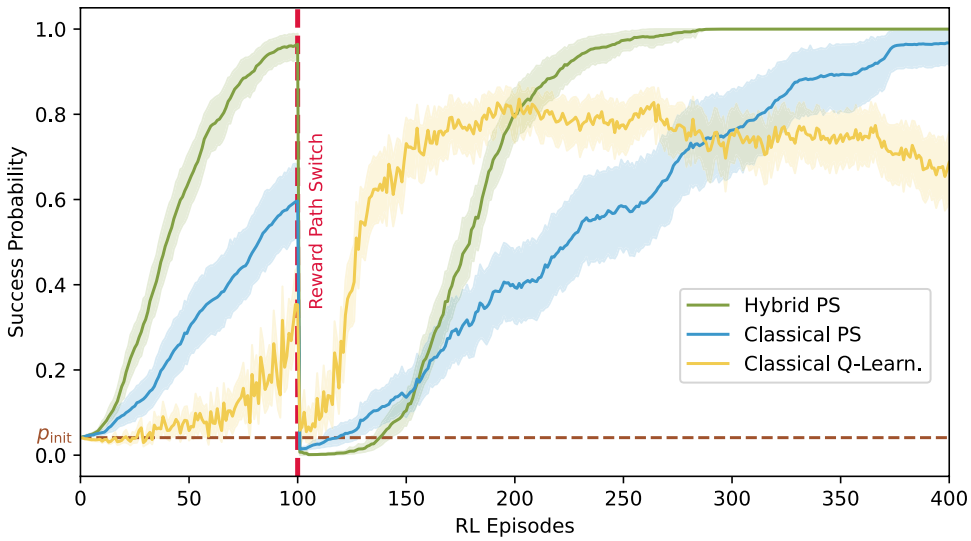


Fig. 4 Comparison of the learning progress between hybrid PS, classical PS, and classical Q-Learning for the changing reward path scenario with layout B and interaction times $N_1 = 100$ and $N_2 = 300$ (scenario B.2). The dashed red vertical line indicates the change of reward paths after 100 episodes. The dotted horizontal line indicates the agents' initial success probability. Average values of $N = 100$ runs, error bands represent the 95% confidence interval

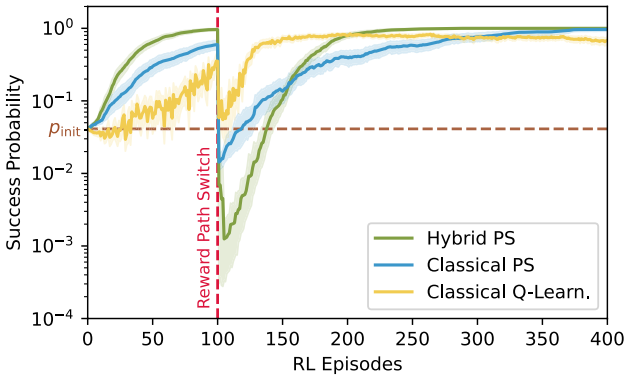


Fig. 5 Comparison of the success probability between hybrid PS, classical PS, and classical Q-Learning for the scenario B.2 on a logarithmic y-scale. Average values of $N = 100$ runs, error bands represent the 95% confidence interval

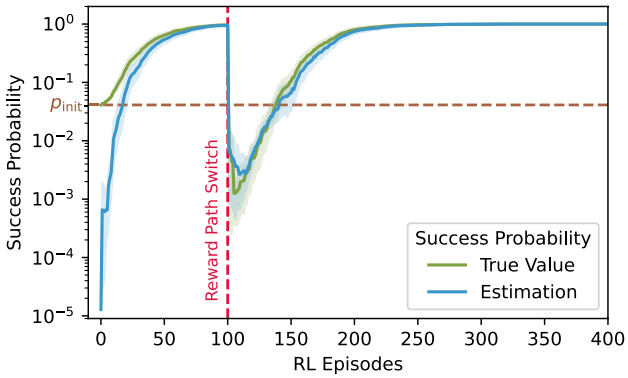


Fig. 6 Comparison of the true success probability Q and the estimate Q_{est} during learning in scenario B.2 with the hybrid PS agent and a dissipation value of $\gamma = 0.05$. Average values of $N = 100$ runs, error bands represent the 95% confidence interval

50%. However, for episodes 200 to 400 its success probability hovers between 60% to 80%. The hybrid PS agent, on the other hand, shows a quick increase similar to the first learning phase, converging to nearly 100% success probability. Classical PS shows a more gradual increase, reaching a final average success probability of 98.4%.

Lastly, for the configuration with the hybrid PS agent and $\gamma = 0.05$, we show the comparison between the true success probability, which we have looked at so far, and the agent's internal estimate Q_{est} in Fig. 6. The estimate diverges most from the true success probability at the start of learning and after the reward path switch. At the start of learning, the estimated success probability can be seen to be several orders of magnitude too low, before closing the gap as the agent learns.

After the reward path switch the estimated success probability is on average slightly larger than the true success probability for about 30 episodes due to the incomplete purging of previously rewarded action sequences. After this period, the estimated value is below the true value, matching the latter closely.

6 Discussion

6.1 Scenario A: Fixed reward path

Even though scenario A features a non-stationary reward function within the episode, the learning problem is similar to a purely stationary problem in the sense that the set of rewarded action sequences does not change. The resulting learning curves of Fig. 3, indicating quicker learning on average for the hybrid PS agent, are therefore in line with existing results (Hamann and Wölk 2022).

Further, the average time of classical PS to find a first reward in scenario A.1 of $\langle t_{\text{class}-\text{PS}} \rangle = 58.4 \pm 1.8$ episodes (cf. Table 2) matches the theoretical expected value of 58.7 episodes well. This expected value follows from

the idea that the classical PS agent's repeated sampling with the initial success probability $p_{\text{init}} = 0.017$ describes a geometric distribution, which terminates on average after $\frac{1}{p_{\text{init}}}$ tries. The lesser time to a first reward of classical Q-Learning stems from the fact that with the modified reward function, non-successful actions can effectively be penalized and thus suppressed in Q-Learning. For the hybrid PS agent, the average time to find the first reward is with $\langle t_{\text{hybr}-\text{PS}} \rangle = 17.9 \pm 0.4$ episodes even fairly below the theoretical upper bound (Hamann and Wölk 2022) of $\frac{9}{2}\sqrt{\langle t_{\text{class}-\text{PS}} \rangle} = 34.4$ episodes. This again is confirmation that the quadratic speedup in sample complexity persists in this scenario.

Lastly, the comparison of average learning times at different stages of the learning progress in Table 2 shows that the advantage of hybrid compared to classical PS in learning time is mainly gained at the early stages of learning, i.e., when the current success probabilities of the agents are low. In fact, the gradual ramp-up of the possible number of AA iterations with successive non-rewarded outcomes (cf. lines 7 and 19 of Algorithm 1) causes the hybrid agent to effectively converge to purely classical play in the limit of large success probabilities. The relatively slow learning progress of classical Q-Learning – despite finding a first reward quickly – can be attributed to its one-step update mechanism: information about "good" actions only propagates backwards from the rewarded state one step at a time, such that it takes some episodes until q-values have been updated even at the starting state. This is in contrast to the glow mechanism used with PS (cf. Appendix B), which immediately updates all states involved in a successful episode.

6.2 Scenario B: Changing reward path

The results of scenario B indicate that the adaptations made to the hybrid PS agent enable it to function well in such continual RL problems. Moreover, our results show that despite the larger success probability at the end of the first path and therefore a lesser chance for exploring new paths at the start of the second path phase, the hybrid PS agent eventually overtakes its classical counterpart in terms of learning progress in all tested configurations.

With respect to the dissipation mechanism, we note that a sensible choice of the dissipation value γ is crucial to balance the trade-off between reaching a high success probability before the reward path switch and unlearning that preference quickly enough after the switch. In the case of our problem layout B, a dissipation value of $\gamma = 0.05$ proves best overall both in scenario B.1 (4 out of 5 successes) and in scenario B.2 ($N_1 = 100$, $N_2 = 300$).

Comparing the classical learning algorithms, (classical) Q-Learning and (classical) PS, we note that PS appears better suited for this type of non-stationary RL problem. On the one hand, PS shows quicker learning progress and on the other hand, it converges closer towards a success probability of one, whereas Q-Learning shows more instability and generally appears not to reach larger success probability than 80% in scenario B.2 due to its ϵ-greedy policy.

Further, we note that the introduced adaptive procedure of computing Q_{est} for the hybrid PS agent matches the true success probability well throughout the learning process. Especially after the reward path switch, Q_{est} overshoots the true value Q only marginally, which is due to the efficient purging mechanism of previously rewarded action sequences. Except these first episodes after the switch, Q_{est} still acts as a lower bound of Q, as intended in the original algorithm (Hamann and Wölk 2022).

Lastly, we emphasize that the chosen scenario is most extreme in the sense that the two sets of rewarded action sequences for the respective reward paths are disjoint. In such a setting, good performance in the first learning phase is detrimental to the initial success chance in the second phase. If a switch were to happen between two reward paths with an partially overlapping set of rewarded action sequences, the hybrid agent's better learning performance in the initial phase could instead contribute positively instead of negatively to subsequent phases, thus further increasing its advantage.

7 Conclusion

In this work we have conducted a first empirical investigation of the hybrid learning agent, which combines a Grover-type quadratic speedup in sample complexity with a classical RL algorithm, in non-stationary RL scenarios. Using a modified Gridworld environment with different types of time-dependent reward functions, we compare the hybrid learning agent combined with the Projective Simulation (PS) RL algorithm to a purely classical PS agent with the same training hyperparameters as well as to a standard implementation of classical Q-Learning.

Our results show that the hybrid agent's speedup naturally persists in the phase before the learning scenario requires any forgetting or relearning. However, not least due to the minor modifications made to the hybrid agent introduced in this work, it is also capable to surpass the performance of its classical counterpart once a significant change in the environment has happened. Given that the problem scenario in this article is chosen such that the hybrid agent's initial good performance puts it at a maximal disadvantage after a change in the reward function, we expect the results of this toy problem

to generalize well to a wider range of problem cases. Our findings motivate the conjecture that the speedup in learning achieved by our hybrid learning agent also enables it to better adapt its behavior to changing environments. This could also enable learning in fast changing environments where a corresponding classical agent would be too slow to learn at all.

Further, we find that the PS algorithm with its dissipation mechanism enables quicker and more stable learning compared to the Q-Learning configuration tested in this work, thus providing a promising basis for the hybrid learning agent in such non-stationary environments. Nevertheless, the combination of the hybrid learning agent with different classical RL methods provides an interesting avenue of further research.

The dissipation mechanism used in this paper in the policy update is a very simple mechanism to adapt the behavior of a learning agent to changing environments. Here, the agent does not actively monitor possible changes in the environment, but adapts passively to them instead. This is a good method for making the learning of an agent more robust to, e.g., initial non-optimal behavior as well as small changes in the environment. If disruptive changes of the environment are expected, more proactive methods for adapting the learning of the hybrid agent to these changes can be beneficial. Therefore, further investigations include broadening the scope of non-stationarity to, e.g., time dependency in the transition probability function. In addition, the combination of the hybrid learning agent with algorithms dedicated to continual RL which, unlike the PS algorithm used here, prevent catastrophic forgetting, is another interesting direction for continued research.

Lastly, efforts towards standardized benchmarking in continual RL problems would be desirable not just for quantum reinforcement learning, but for RL in general. Additions of continual RL problems to, e.g., the popular *Gymnasium* (Towers et al. 2025) software package for "non-continual" RL, could greatly help to improve the quantitative understanding beyond this exploratory work.

Appendix A Additional plots and tables for scenario B.2

Table 4 Average success probabilities of hybrid PS, classical PS, and classical Q-Learning for the changing reward path scenario with interaction times $N_1 = 100$ and $N_2 = 300$ (scenario B.2) and dissipation values $\gamma \in \{0.01, 0.02, 0.05, 0.1\}$. Values indicate the mean with its standard error across $N = 100$ runs per configuration. The highest average success probability of each category is printed in boldface

γ	Agent	Avg. Success Probability [%]		
		Before Switch	After Switch	Total
0.01	Class. PS	50.0(0.4)	13.5(0.2)	22.7(0.2)
	Hybrid PS	**70.3(0.4)**	5.5(0.1)	21.6(0.2)
0.02	Class. PS	49.7(0.4)	39.2(0.3)	41.8(0.2)
	Hybrid PS	64.2(0.4)	45.8(0.3)	50.5(0.2)
0.05	Class. PS	29.4(0.4)	61.6(0.3)	53.5(0.2)
	Hybrid PS	56.8(0.4)	**73.2(0.2)**	**69.0(0.2)**
0.1	Class. PS	11.2(0.3)	28.8(0.3)	24.4(0.2)
	Hybrid PS	30.7(0.3)	66.4(0.2)	57.4(0.2)
—	Class. Q-Learn.	9.9(0.2)	69.7(0.2)	54.6(0.2)

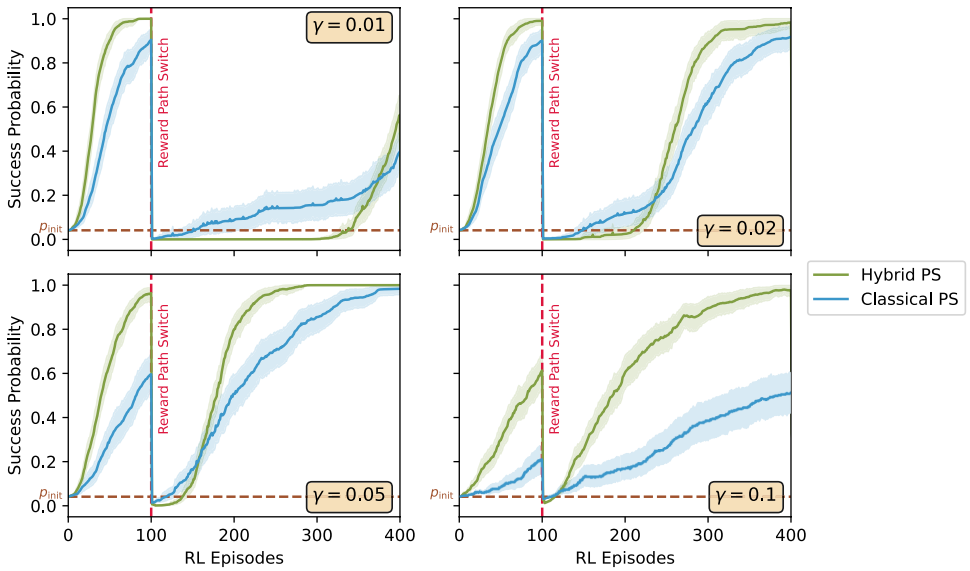


Fig. 7 Learning curves of hybrid and classical PS agents for the changing reward path scenario B.2 with interaction times $N_1 = 100$ and $N_2 = 300$ and dissipation values $\gamma \in \{0.01, 0.02, 0.05, 0.1\}$. The dashed red vertical line indicates the change of reward paths after 100 episodes. The dotted horizontal line indicates the agents' initial success probability. Average values of $N = 100$ runs, error bands represent the 95% confidence interval

Appendix B Projective simulation

Projective Simulation (Briegel and De las Cuevas 2012) is a (tabular) RL method based on a random walk along a learned network of nodes or *clips*, with transition probabilities governed by a clip's outgoing edge weights.

A clip may either be an action a or a percept s. Through interaction, the agent discovers connections from action to percept clips and vice versa. These connections are stored in form of a graph network called the *episodic & compositional memory* (ECM), which builds up over the process of learning. In an MDP, this network has the form of one layer of percept clips, which is fully connected to a layer of action clips as illustrated in Fig. 8. An input percept triggers a random walk starting at that respective percept clip and results in outputting the action associated to the clip at which the random walk terminates.

Edges from a percept s to an available action a are assigned a so-called *h-value* $h(s, a)$, which is initially set to one. The agent's policy π is determined by either linearly weighing the h-values

$$\pi(a|s) = \frac{h(s,a)}{\sum_{a\in\mathcal{A}} h(s,a)} \tag{13}$$

or by using the *softmax* function

$$\pi(a|s) = \frac{e^{\beta\cdot h(s,a)}}{\sum_{a\in\mathcal{A}} e^{\beta\cdot h(s,a)}} \tag{14}$$

with some pseudo-temperature parameter $\beta \geq 0 \in \mathbb{R}$, which was introduced by Melnikov et al. (2018).

After an interaction step, the h-values $h_{t+1}(s, a)$ of the subsequent time step $t + 1$ are updated by adding the received reward $r_t(s, a)$ to the h-values of the current time step t:

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

To this basic PS framework, several additions have been made, of which we mention the ones relevant to our problem. First, the *dissipation* mechanism introduced in the original paper (Briegel and De las Cuevas 2012) enables the agent to forget learned preferences, resulting in its adaptability to changing environments. With the introduction of the dissipation parameter $\gamma \in [0, 1]$, we add a term to the basic h-value update rule of Eq. (15):

$$h_{t+1}(s,a) = h_t(s,a) + r_t(s,a) - \gamma\big(h_t(s,a) - 1\big) \tag{16}$$

The effect of this term for values $\gamma \in (0, 1)$ is to drive all h-values to some degree towards their initial value of 1 after each interaction step. The case $\gamma = 0$ obviously retrieves the scenario without dissipation, whereas the case $\gamma = 1$ simplifies Eq. (16) to $h_t(s, a) = 1 + r_t(s, a)$ and thus essentially resets the agent to its initial h-values after each time step.

Second, we present the *edge glow* or *afterglow* mechanism which was introduced in Mautner et al. (2015). The purpose of this mechanism is to help the agent deal with delayed rewards in episodic environments. In many RL

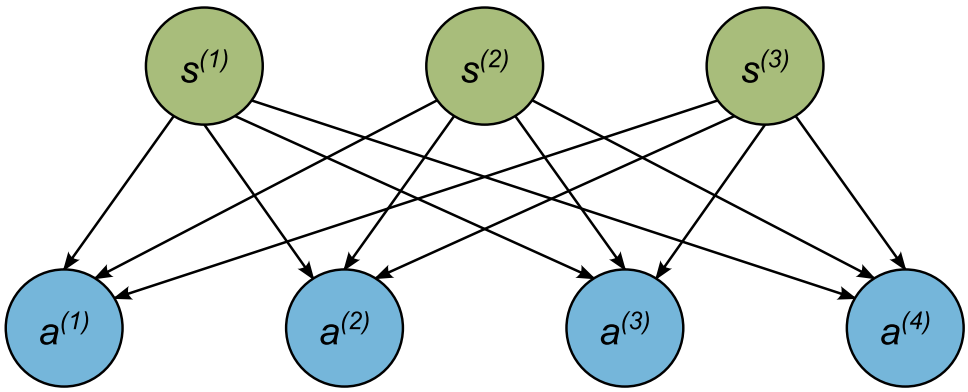


Fig. 8 Illustration of the basic ECM layout with exemplary percept and action sets $\mathcal{S} = \{ s^{(1)}, s^{(2)}, s^{(3)} \}$ and $\mathcal{A} = \{ a^{(1)}, a^{(2)}, a^{(3)}, a^{(4)} \}$, respectively. Arrows from the percept (green) to the action (blue) clips indicate possible transitions, with an *h-value* assigned to each of them

environments which are games, e.g., chess or black-jack, non-zero rewards are not issued during the game, but just upon the ultimate outcome. Thus, if such an episode consists of L time steps, $r_t(s,a) \neq 0$ just for $t = L$ and the combination of percept and action encountered/played in that time step. In the following, we use the shorthand notation r to denote this reward value at the end of the episode. Naturally, not just the final action in the episode, but potentially the entire sequence of preceding actions has an impact on the outcome of the episode.

The idea of edge glow is to attribute a contribution of previous actions to receiving a delayed reward, with the contribution being more significant the closer the action was temporally to the reward. To this end, this concept introduces the notion of *excitations* of edges in the ECM with the so-called glow value g. Initially, the glow value is set to zero for all transitions in the ECM, $g_0(s,a) = 0 \;\; \forall s \in \mathcal{S}, a \in \mathcal{A}$. An edge corresponding to selecting action a at percept s is excited when this (s, a) combination occurs in the interaction. Upon excitation at time t, the glow value is set to one for this edge, $g_t(s,a) = 1$. With time, these excitations decay towards zero following the update rule

$$g_{t+1}(s,a) = (1-\eta) \cdot g_t(s,a) \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{17}$$

with glow decay parameter $\eta \in [0,1]$. Upon updating the h-values, the glow value modulates the reward as

$$\begin{aligned} h_{t+1}(s,a) = \;& h_t(s,a) + g_t(s,a) \cdot r \\ & -\gamma\big(h_t(s,a) - 1\big). \end{aligned} \tag{18}$$

As actions played in a previous episode are not meaningful to a current episode's outcome, all glow values must be reset to zero at the start of a new episode in episodic environments.

In this work, we use PS solely for episodic RL tasks with a binary reward at the end of an episode. It is therefore evident that updates of the h-values at intermediate steps within an episode have just a contribution coming from the dissipation term. To enable quicker computation, we omit this dissipation update within the episode in our implementation such that we perform just one update at the end of each episode. Note that works which apply dissipation at each time step likely require lower values for the dissipation parameter due to the increased frequency of dissipation updates. The full procedure to update the h-values and the policy in PS using dissipation, glow, and a softmax policy is shown in pseudocode form in Algorithm 2.

Algorithm 2 Projective simulation version of `policy_update`

Input: sequence of played actions $\vec{a} = (a_0, a_1, \ldots, a_{L-1})$, sequence of encountered percepts $\vec{s} = (s_0, s_1, \ldots, s_L)$, boolean reward information `rewarded`

Require: table of h-values $h(s,a)$ for all actions $a \in \mathcal{A}$ and percepts $s \in \mathcal{S}$, dissipation parameter γ, glow parameter η, softmax temperature parameter β

initialize glow values $g(s,a) = 0 \;\; \forall s \in \mathcal{S}, a \in \mathcal{A}$
if `rewarded` **then**
  $r \leftarrow 1$
  for $i = 0$ **to** $L-1$ **do**
    ▷ *glow decay* ◁
    $g(s,a) \leftarrow (1-\eta) \cdot g(s,a) \;\; \forall s \in \mathcal{S}, a \in \mathcal{A}$
    ▷ *glow reset for excitation* ◁
    $g(s_i, a_i) \leftarrow 1$
else
  $r \leftarrow 0$
for $(s,a) \in \mathcal{S} \otimes \mathcal{A}$ **do**
  ▷ *h-value update* ◁
  $h(s,a) \leftarrow h(s,a) + g(s,a) \cdot r - \gamma\big(h(s,a) - 1\big)$
  ▷ *policy computation* ◁
  $\pi(a|s) \leftarrow \dfrac{e^{\beta \cdot h(s,a)}}{\sum_{a \in \mathcal{A}} e^{\beta \cdot h(s,a)}}$

Finally, the so-called *mapping* focuses on the compositional aspect of the ECM. Here, the agent builds the ECM network as a map of the environment to learn the connectivity between different states. This is contrary to the two-layer input-output coupling shown in Fig. 8. Instead, the node corresponding to a chosen action is connected to the node representing the next percept. If that percept has not yet been part of the learning history, a new node is created instead. This way, using its past knowledge, the agent can anticipate to which state an action may lead given its current state. In deterministic environments especially, an agent may thus not just compute the probability $\pi(a|s)$ according to its policy to execute action a in state s. Instead, it can also compute the probability to execute a sequence of actions $\vec{a}$ as it has knowledge of the intermediate states these actions will lead to (if the map is not built up yet for a certain part of the action sequence, the uniform probability is again

assumed). This possibility to compute a full action sequence probability $\hat{\pi}(\vec{a}|s)$ (we omit the condition on the state s in the main text as we always start from a fixed initial state s_0) is crucial for the hybrid learning agent.

B.1 Hybrid agent: repeated diffusion

To accommodate for the fact that the hybrid learning agent updates less frequently compared to a classical RL agent due to the overhead of interaction episodes required for amplitude amplification, we must increase the strength of the dissipation per update to enable an equal "forgetting" rate. One method to achieve this would be a $2k$-fold application of the dissipation update for each of the $2k$ episodes spent with amplitude amplification before the update with the actual reward information. This, however, appears largely inefficient. Our goal, therefore, is to adapt the dissipation part of the update rule of Eq. (16) for the hybrid agent such that the two following updates are equal:

- one update after $2k+1$ episodes of using the hybrid agent with no rewards during episodes 1 to $2k$ and a possible reward in episode $2k+1$,
- continuous updates in $2k+1$ episodes of classical play with no reward during episodes 1 to $2k$ and a possible reward in episode $2k+1$.In this case, as described above, we update just at the end of an episode instead of after every time step, we use the uppercase index $h^{(\tau)}(s,a)$ to denote the h-value of state-action pair (s, a) throughout episode τ and in the same manner $r^{(\tau)}$ to denote the binary reward received in episode τ. In this notation, the update rule of PS with dissipation of Eq. (16) reads:

$$h^{(\tau+1)}(s,a) = h^{(\tau)}(s,a) + r^{(\tau)} - \gamma\big(h^{(\tau)}(s,a) - 1\big). \tag{19}$$

We claim that the update rule

$$h^{(t+N)} = \big(h^{(t)} - 1\big)\cdot(1-\gamma)^N + 1 + r^{(t+N-1)} \tag{20}$$

is equivalent to apply $N-1$ updates with just dissipation and no reward, $r^{(\tau)} = 0$ for $\tau \in \{t, \ldots, t+N-2\}$, and one update with dissipation and a possible reward $r^{(t+N-1)}$ and, therefore, fulfills the goal defined above. This equivalence can be shown via induction as follows (to simplify notation, we abbreviate $h^{(\tau)}(s,a)$ as $h^{(\tau)}$). For $N=1$, some reshuffling of terms yields

$$\begin{aligned} h^{(t+1)} &\overset{(Equ19)}{=} h^{(t)} + r^{(t)} - \gamma(h^{(t)} - 1) \\ &= \big(h^{(t)} - 1\big)\cdot(1-\gamma)^1 + 1 + r^{(t)}. \end{aligned}$$

For the induction step $N \to N+1$, we have

$$\begin{aligned} & h^{(t+N+1)} \\ \overset{(Equ19)}{=}\ & h^{(t+N)} - \gamma(h^{(t+N)} - 1) + r^{(t+N)} \\ =\ & h^{(t+N)}(1-\gamma) + \gamma + r^{(t+N)} \\ \overset{(Equ20)}{=}\ & \Big[\big(h^{(t)} - 1\big)\cdot(1-\gamma)^N + 1 + r^{(t+N-1)}\Big](1-\gamma) \\ & + \gamma + r^{(t+N)} \\ =\ & \big(h^{(t)} - 1\big)\cdot(1-\gamma)^{(N+1)} + (1-\gamma) \\ & + r^{(t+N-1)}(1-\gamma) + \gamma + r^{(t+N)} \\ =\ & \big(h^{(t)} - 1\big)\cdot(1-\gamma)^{(N+1)} + 1 + r^{(t+N)}, \end{aligned}$$

where in the last equality we have used the fact that intermediate rewards are zero. This completes the proof.

Appendix C Estimation of the probability lower bound

As we have mentioned in our description of scenario A, the hybrid agent as presented in Algorithm 1 can be used without any adaptations. However, we need to specify how we initialize (line 2 of Algorithm 1) and later update (line 22 of Algorithm 1) the lower bound estimate $Q_{\min}$ of the current success probability.

As the initial estimate, we choose $Q_{\min} = |\mathcal{A}|^{-T}$, which corresponds to one rewarded path among the $|\mathcal{A}|^T$ possible paths for the chosen action set $\mathcal{A}$ and episode length T. Here, we must ensure that the chosen episode length is sufficient to yield at least one rewarded path. Further, we assume that each path is sampled with equal probability initially. In tabular versions of Q-Learning and SARSA with an equal initialization of q-values and an epsilon-greedy policy with random tie-breaks as well as in Projective Simulation, this assumption holds true.

As soon as we have found a rewarded action sequence, we can improve the estimate. We introduce the initially empty set $\mathcal{R}_{\text{found}}$ of rewarded action sequences the agent has encountered during its learning. If the agent now samples an action sequence $\vec{a}$ which is rewarded, we first investigate how many of the T actions were necessary to reach the target. Doing so, we can split the action sequence $\vec{a} = \vec{a}_{\text{trunc}} \oplus \vec{a}_{\text{rest}}$ into the truncated part of length $T_{\text{trunc}} \leq T$ and a rest part $\vec{a}_{\text{rest}}$ of length $T_{\text{rest}} \geq 0$, $T_{\text{trunc}} + T_{\text{rest}} = T$ such that executing $\vec{a}_{\text{trunc}}$ exactly reaches the target. Then, we add the truncated action sequence to $\mathcal{R}_{\text{found}}$. After a first reward is found, we estimate the lower bound for Q as:

$$Q_{\min} = \sum_{\vec{b} \in \mathcal{R}_{\text{found}}} \hat{\pi}(\vec{b}) \,, \tag{21}$$

which is still a lower bound by standard probabilistic arguments. Furthermore, to ensure that $Q_{\min}$ is strictly larger than zero, we must require the policy to fulfill $\pi(a|s) > 0 \; \forall a \in \mathcal{A}, s \in \mathcal{S}$.

The chosen procedure usually results quite quickly in "good" lower bounds $Q_{\min}$ close to the actual probability Q. However, it is also possible to use predetermined hard-coded lower limits. In this case, the effort to remember all rewarded action sequences and adapt them as described in the next section is not necessary.

In general, the hybrid learning agent is relatively robust to lower bounds $Q_{\min}$ which are either very low or even wrong because the value is too high. In the case of a very low bound, ramping up the maximal number of Grover iterations and randomly sampling the number of amplitude amplification rounds k (see Algorithm 1) solves the problem that the resulting probability of finding a rewarded sequence of action decreases again if k is chosen too large. If the estimation of $Q_{\min}$ is too large (such as in our example problem after the environment has changed) the hybrid agent will perform on average less steps k of amplitude amplification. In this case, the hybrid agent still achieves a speedup in learning. However, no quadratic scaling can be achieved, as discussed in detail in Hamann and Wölk (2022).

C.1 Adaptation for changing reward paths

To enable the hybrid learning agent for scenario B, which introduces a switch to a different reward path after some time during training, we must ensure two aspects: On the one hand, a forgetting mechanism is crucial to unlearn the preference for the initial path. In this regard, the hybrid agent does not differ from classical RL algorithms. However, this forgetting mechanism is typically implemented in the RL algorithm the hybrid agent is used in combination with. On the other hand, the estimation method of the current success probability must be adapted, which is unique to the hybrid agent. While the forgetting mechanism lowers occurrence probabilities for the previously rewarded action sequences over time, as they are no longer reinforced, this happens slowly and does not correct for the fact that these action sequences should no longer contribute to the estimate $Q_{\min}$.

To continue with the estimation method as in Eq. (21) in the changing reward path scenario, we must remove action sequences from $\mathcal{R}_{\text{found}}$ if we find them to be no longer rewarded. This *purging* procedure can be implemented as follows. Whenever we sample an action sequence $\vec{a}$ which is not rewarded, we compare it to all action sequences of $\mathcal{R}_{\text{found}}$. If the sequence $\vec{a}$ or some possible truncation $\vec{a}_{\text{trunc}}$ is an element of $\mathcal{R}_{\text{found}}$, it is no longer rewarded after the switch and hence dropped from the set $\mathcal{R}_{\text{found}}$.

The obvious implication of this procedure is that the estimate built from summing the occurrence probabilities of all action sequences in $\mathcal{R}_{\text{found}}$ is no longer a lower bound of the true success probability. To avoid ambiguity in Algorithm 3, which describes the adapted hybrid agent for the changing reward path scenario, we rename the estimate $Q_{\min}$ as Q_{est}. Whether the hybrid agent can perform well in the post-switch phase with this new estimation scheme is a key question in our experiments. Furthermore, it may occur that all previously rewarded action sequences are purged from $\mathcal{R}_{\text{found}}$ before a new rewarded sequence is found. In this case, a fall-back option to estimate Q_{est} is to use the lowest occurrence probability of any action sequence. This however, requires computing the occurrence probability of all action sequences – which has an exponential computational cost with respect to the episode length – because the the algorithm has learned in the meantime and the starting estimate of $|\mathcal{A}|^{-T}$ likely no longer holds.

Algorithm 3 Adapted hybrid learning agent

Require: MDP $\langle T, \mathcal{S}, \mathcal{A}, p_t^{(\mathrm{det})}(s,a), r_t(s,a)\rangle$,
initial percept s_0,
Grover operator G,
`policy_update` subroutine

initialize algorithm constant $\lambda \leftarrow 5/4$
initialize set of found rewards $\mathcal{R}_{\mathrm{found}} \leftarrow \emptyset$
initialize lower bound estimate $Q_{\min} \leftarrow |\mathcal{A}|^{-T}$
algorithm parameter $m \leftarrow 1$
`rewarded` $\leftarrow$ `false`
▷ *Quantum Part* ◁
$k \leftarrow$ random integer in $[0, \mathrm{m})$
prepare $|\psi\rangle \leftarrow \sum_{\vec{a}\in\mathcal{A}^{\otimes T}} \sqrt{\hat{\pi}(\vec{a})}\,|\vec{a}\rangle_A\,|\vec{s}_{\mathrm{init}}\rangle_S\,|-\rangle_R$
amplitude amplification $|\psi'\rangle \leftarrow G^k\,|\psi\rangle$
$\vec{a}' \leftarrow$ **measure** $|\psi'\rangle$
▷ *Classical Part* ◁
for $i = 1$ **to** T **do**
  $s_i \leftarrow p_i^{(\mathrm{det})}(s_{i-1}, a_{i-1})$
  $r_i \leftarrow r_i(s_{i-1}, a_{i-1})$
  if $r_i = 1$ **then**
    `rewarded` $\leftarrow$ `true`
    $\vec{s} \leftarrow (s_0, s_1, \ldots, s_i)$
    $\vec{a}'_{\mathrm{trunc}} \leftarrow (a'_0, a'_1, \ldots, a'_{i-1})^\top$
    $\mathcal{R}_{\mathrm{found}} \leftarrow \mathcal{R}_{\mathrm{found}} \cup \{\vec{a}'_{\mathrm{trunc}}\}$
▷ *Classical RL Update* ◁
update π using subroutine `policy_update`$(\vec{a}', \vec{s}, $`rewarded`$)$
▷ *Update of Probability Estimate* ◁
if `not rewarded` **then**
  purge $\vec{a}'$ (or a possible truncation $\vec{a}'_{\mathrm{trunc}}$) from $\mathcal{R}_{\mathrm{found}}$ if $\vec{a}' \in \mathcal{R}_{\mathrm{found}}$
if $|\mathcal{R}_{\mathrm{found}}| \geq 1$ **then**
  $Q_{\min} \leftarrow \sum_{\vec{a}\in\mathcal{R}_{\mathrm{found}}} \hat{\pi}(\vec{a})$
else
  $Q_{\min} \leftarrow |\mathcal{A}|^{-T}$
if `rewarded` **then**
  repeat from step 4
else
  $m \leftarrow \min\left(\lambda \cdot m,\ Q_{\min}^{-1/2}\right)$
  repeat from step 5

Appendix D Implementation details

In this section, we provide details specifically for simulating the hybrid learning agent without requiring an actual (or simulated) quantum computer. Since Grover's algorithm is out of reach for current NISQ hardware for all but the most trivial problems, we instead simulate the outcome of performing amplitude amplification on an ideal quantum computer similarly to Sefrin and Wölk (2025).

To do so, we continuously keep track of the probabilities of all action sequences, which allows us to compute the success probability obtained by classical sampling at any time. Having this probability p_{class}, we can compute the success probability p_{AA} after some amount k of Grover iterations using the well-known equation (Brassard et al. 2002)

$$p_{\mathrm{AA}} = \sin^2\left([2k+1]\arcsin p_{\mathrm{class}}^{1/2}\right). \tag{22}$$

Thus, whenever a measurement is performed in the hybrid learning agent algorithm, we sample from the set of rewarded sequences with probability p_{AA} and from the set of non-rewarded sequences with probability $(1 - p_{\mathrm{AA}})$. Within each set, we draw the samples in accordance with their current probability of being played.

The bookkeeping steps necessary to continuously keeping track of the success probabilities as described above are the following:

1. For each layout and reward path, divide the set of possible action sequences into rewarded and non-rewarded ones. This needs to be computed only once.
2. Compute the initial probability of being played for each action sequence (for the algorithms in our case, we always start with a uniform distribution). Then, sum up the probabilities of all sequences in the rewarded set to receive the current (classical) success probability.
3. Whenever an update of the agent's policy-defining parameters (such as q-values or h-values in our case) takes place, compute the probability of being played for each action sequence based on the updated policy. Then, sum up the probabilities of all sequences in the rewarded set to receive the current (classical) success probability.
4. Repeat step 3 until training has finished.

Acknowledgements This article was written as part of the Qu-Gov project, which was commissioned by the German Federal Ministry of Finance. M. R. and L. S. want to extend their gratitude to Kim Nguyen, Manfred Paeschke, Oliver Muth, Andreas Wilke, and Yvonne Ripke for their continuous encouragement and support. O. S. and S. W. thank Annette Zapf for helpful comments.

Author Contributions S.W. and O.S. conceived the study. O.S. generated the data for the example problems, implemented the algorithms, and provided the visualization and aggregation of the results. All authors contributed to analyzing and interpreting the results. S.W. and O.S. wrote the manuscript. All authors contributed to editing and revising the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

Clinical trial number Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Abel D, Barreto A, Van Roy B, Precup D, Hasselt HP, Singh S (2023) A definition of continual reinforcement learning. Adv Neural Inf Process Syst vol 36. https://proceedings.neurips.cc/paper_files/paper/2023/hash/9d8cf1247786d6dfeefeeb53b8b5f6d7-Abstract-Conference.html

Acampora G, Cuzzocrea A, Lapegna M, Schiattarella R, Vitiello A (2025) Fuzzy reward-based reinforcement learning for clifford circuit synthesis. In: IEEE Int Conf Fuzzy Syst (FUZZ), pp 1–6. https://doi.org/10.1109/FUZZ62266.2025.11152245

Altmann P, Stein J, Kölle M, Bärligea A, Zorn M, Gabor T, Phan T, Feld S, Linnhoff-Popien C (2024) Challenges for reinforcement learning in quantum circuit design. In: IEEE Int Conf Quantum Comput Eng (QCE), vol 01, pp 1600–1610. https://doi.org/10.1109/QCE60285.2024.00187

Andreasson P, Johansson J, Liljestrand S, Granath M (2019) Quantum error correction for the toric code using deep reinforcement learning. Quantum 3:183. https://doi.org/10.22331/q-2019-09-02-183

Boyer M, Brassard G, Høyer P, Tapp A (1998) Tight bounds on quantum searching. Fortschr Phys 46(4–5):493–505. https://doi.org/10.1002/(SICI)1521-3978(199806)46:4/5%3C;493::AID-PROP493%3E;3.0.CO;2-P

Brassard G, Høyer P, Mosca M, Tapp A (2002) Quantum amplitude amplification and estimation. Contemp Math 305:53–74. https://doi.org/10.1090/conm/305/05215

Briegel HJ, De las Cuevas G (2012) Projective simulation for artificial intelligence. Sci Rep 2(1):400. https://doi.org/10.1038/srep00400

Bukov M, Day AGR, Sels D, Weinberg P, Polkovnikov A, Mehta P (2018) Reinforcement learning in different phases of quantum control. Phys Rev X 8:031086. https://doi.org/10.1103/PhysRevX.8.031086

Cerezo M, Arrasmith A, Babbush R, Benjamin SC, Endo S, Fujii K, McClean JR, Mitarai K, Yuan X, Cincio L, Coles PJ (2021) Variational quantum algorithms. Nat Rev Phys 3(9):625–644. https://doi.org/10.1038/s42254-021-00348-9

Cerezo M, Larocca M, García-Martín D, Diaz NL, Braccia P, Fontana E, Rudolph MS, Bermejo P, Ijaz A, Thanasilp S, Anschuetz ER, Holmes Z (2025) Does provable absence of barren plateaus imply classical simulability? Nat Commun 16(1):7907. https://doi.org/10.1038/s41467-025-63099-6

Chen SY-C, Yang C-HH, Qi J, Chen P-Y, Ma X, Goan H-S (2020) Variational quantum circuits for deep reinforcement learning. IEEE Access 8:141007–141024. https://doi.org/10.1109/ACCESS.2020.3010470

Chen H-Y, Chang Y-J, Liao S-W, Chang C-R (2024) Deep Q-learning with hybrid quantum neural network on solving maze problems. Quantum Mach Intell 6(1):2. https://doi.org/10.1007/s42484-023-00137-w

Cherrat EA, Kerenidis I, Prakash A (2023) Quantum reinforcement learning via policy iteration. Quantum Mach Intell 5(2):30. https://doi.org/10.1007/s42484-023-00116-1

Choi SPM, Yeung D-Y, Zhang NL (1999) Hidden-mode markov decision processes. In: Proc 16th Int Jt Conf Artif Int (IJCAI-99), Stockholm, Sweden

Ciliberto C, Rocchetto A, Rudi A, Wossnig L (2020) Statistical limits of supervised quantum learning. Phys Rev A 102:042414. https://doi.org/10.1103/PhysRevA.102.042414

Cireşan DC, Meier U, Masci J, Gambardella LM, Schmidhuber J (2011) Flexible, high performance convolutional neural networks for image classification. In: Proc 22nd Int Jt Conf Artif Int. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-210

da Silva BC, Basso EW, Bazzan ALC, Engel PM (2006) Dealing with non-stationary environments using context detection. In: Proc 23rd Int Conf Mach Learn. https://doi.org/10.1145/1143844.1143872

Dalla Pozza N, Buffoni L, Martina S, Caruso F (2022) Quantum reinforcement learning: the maze problem. Quantum Mach Intell 4(1):11. https://doi.org/10.1007/s42484-022-00068-y

Dong D, Chen C, Li H, Tarn T-J (2008) Quantum reinforcement learning. IEEE Trans Syst Man Cybern B Cybern 38(5):1207–1220. https://doi.org/10.1109/TSMCB.2008.925743

Dunjko V, Taylor JM, Briegel HJ (2016) Quantum-enhanced machine learning. Phys Rev Lett 117:130501. https://doi.org/10.1103/PhysRevLett.117.130501

Dunjko V, Liu Y-K, Wu X, Taylor JM (2018) Exponential improvements for quantum-accessible reinforcement learning. arxiv:1710.11160

Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: Proc 34th Int Conf Mach Learn

Fösel T, Niu MY, Marquardt F, Li L (2021) Quantum circuit optimization with deep reinforcement learning. arxiv:2103.07585

Ganguly B, Wu Y, Wang D, Aggarwal V (2023) Quantum computing provides exponential regret improvement in episodic reinforcement learning. arxiv:2302.08617

Gil-Fuster E, Gyurik C, Pérez-Salinas A, Dunjko V (2025) On the relation between trainability and dequantization of variational quantum learning models. arxiv:2406.07072

Giordano S, Sen K, Martin-Delgado MA (2025) Hybrid reward-driven reinforcement learning for efficient quantum circuit synthesis. arxiv:2507.16641

Grover LK (1997) Quantum mechanics helps in searching for a needle in a haystack. Phys Rev Lett 79:325–328. https://doi.org/10.1103/PhysRevLett.79.325

Hamann A, Wölk S (2022) Performance analysis of a hybrid agent for quantum-accessible reinforcement learning. New J Phys 24(3):033044. https://doi.org/10.1088/1367-2630/ac5b56

Hamann A, Dunjko V, Wölk S (2021) Quantum-accessible reinforcement learning beyond strictly epochal environments. Quantum Mach Intell 3(2):22. https://doi.org/10.1007/s42484-021-00049-7

Jerbi S, Gyurik C, Marshall SC, Briegel HJ, Dunjko V (2021) Parametrized quantum policies for reinforcement learning. In: Adv Neural Inf Process Syst 34, Virtual Event. https://proceedings.neurips.cc/paper_files/paper/2021/file/eec96a7f788e88184c0e713456026f3f-Paper.pdf

Jerbi S, Trenkwalder LM, Poulsen Nautrup H, Briegel HJ, Dunjko V (2021) Quantum enhancements for deep reinforcement learning

in large spaces. PRX Quantum 2:010328. https://doi.org/10.1103/PRXQuantum.2.010328
Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: A survey. J Artif Intell Res 4:237–285. https://doi.org/10.1613/jair.301
Khetarpal K, Riemer M, Rish I, Precup D (2022) Towards continual reinforcement learning: A review and perspectives. J Artif Intell Res 75:1401–1476. https://doi.org/10.1613/jair.1.13673
Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90. https://doi.org/10.1145/3065386
Li S, Fan Y, Li X, Ruan X, Zhao Q, Peng Z, Wu R-B, Zhang J, Song P (2025) Robust quantum control using reinforcement learning from demonstration. NPJ Quantum Inf 11(1):124. https://doi.org/10.1038/s41534-025-01065-2
Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2019) Continuous control with deep reinforcement learning. arxiv:1509.02971
Liu Y, Arunachalam S, Temme K (2021) A rigorous and robust quantum speed-up in supervised machine learning. Nat Phys 17(9):1013–1017. https://doi.org/10.1038/s41567-021-01287-z
Lockwood O, Si M (2020a) Playing atari with hybrid quantum-classical reinforcement learning. In: Proc Mach Learn Res, Virtual Event. https://proceedings.mlr.press/v148/lockwood21a.html
Lockwood O, Si M (2020b) Reinforcement learning with quantum variational circuit. Proc AAAI Conf Artif Intell Interact Digit Entertain 16(1):245–251. https://doi.org/10.1609/aiide.v16i1.7437
Luketina J, Flennerhag S, Schroecker Y, Abel D, Zahavy T, Singh S (2022) Meta-gradients in non-stationary environments. In: Proc 1st conf lifelong learn agents proc mach learn res. PMLR, Montréal, Canada, vol 199, pp 886–901. https://proceedings.mlr.press/v199/luketina22a.html
Mautner J, Makmal A, Manzano D, Tiersch M, Briegel HJ (2015) Projective simulation for classical learning agents: a comprehensive investigation. New Gener Comput 33:69–114. https://doi.org/10.1007/s00354-015-0102-0
McCloskey M, Cohen, NJ (1989) Catastrophic interference in connectionist networks: the sequential learning problem. Psychol Learn Motiv Academic Press, vol 24, pp 109–165. https://doi.org/10.1016/S0079-7421(08)60536-8
Melnikov AA, Makmal A, Briegel HJ (2018) Benchmarking projective simulation in navigation problems. IEEE Access 6:64639–64648. https://doi.org/10.1109/ACCESS.2018.2876494
Meyer N, Scherer DD, Plinge A, Mutschler C, Hartmann MJ (2023) Quantum natural policy gradients: towards sample-efficient reinforcement learning. In: IEEE int conf quantum comput eng (QCE). https://doi.org/10.1109/QCE57702.2023.10181
Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533. https://doi.org/10.1038/nature14236
Molteni R, Gyurik C, Dunjko V (2024) Exponential quantum advantages in learning quantum observables from classical data. arxiv:2405.02027
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
Niu MY, Boixo S, Smelyanskiy VN, Neven H (2019) Universal quantum control through deep reinforcement learning. NPJ Quantum Inf 5(1):33. https://doi.org/10.1038/s41534-019-0141-3
Padakandla S (2021) A survey of reinforcement learning algorithms for dynamically varying environments. ACM Comput Surv 54(6):1–25. https://doi.org/10.1145/3459991
Paparo GD, Dunjko V, Makmal A, Martin-Delgado MA, Briegel HJ (2014) Quantum speedup for active learning agents. Phys Rev X 4:031002. https://doi.org/10.1103/PhysRevX.4.031002
Pieters M, Wiering MA (2016) Q-learning with experience replay in a dynamic environment. In: IEEE symp ser comput intell (SSCI). https://doi.org/10.1109/SSCI.2016.7849368
Preskill J (2018) Quantum computing in the NISQ era and beyond. Quantum 2:79. https://doi.org/10.22331/q-2018-08-06-79
Puterman ML (1994) Markov decision processes: discrete stochastic dynamic programming, 1st edn. John Wiley & Sons Inc, Hoboken, NJ, USA
Ring MB (1994) Continual learning in reinforcement environments. PhD thesis, The University of Texas, Austin, TX, USA
Rummery GA, Niranjan M (1994) On-line Q-learning using connectionist systems, vol 37. University of Cambridge, Department of Engineering, Cambridge, UK
Saggio V, Asenbeck BE, Hamann A, Strömberg T, Schiansky P, Dunjko V, Friis N, Harris NC, Hochberg M, Englund D, Wölk S, Briegel HJ, Walther P (2021) Experimental quantum speed-up in reinforcement learning agents. Nature 591(7849):229–233. https://doi.org/10.1038/s41586-021-03242-7
Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T, Lillicrap T, Silver D (2020) Mastering atari, go, chess and shogi by planning with a learned model. Nature 588(7839):604–609. https://doi.org/10.1038/s41586-020-03051-4
Schuld M, Killoran N (2022) Is quantum advantage the right goal for quantum machine learning? PRX Quantum 3:030101. https://doi.org/10.1103/PRXQuantum.3.030101
Schuld M, Petruccione F (2021) Machine learning with quantum computers. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-83098-4
Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arxiv:1707.06347
Sefrin O, Wölk S (2025) A hybrid learning agent for episodic learning tasks with unknown target distance. Quantum Mach Intell 7(1):52. https://doi.org/10.1007/s42484-025-00269-1
Shor PW (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: Proc 35th annu symp found comput sci. https://doi.org/10.1109/SFCS.1994.365700
Singh S, Lewis RL, Barto AG (2009) Where do rewards come from? In: Proc 31st annu conf cogn sci soc, pp 2601–2606. https://all.cs.umass.edu/pubs/2009/singh_l_b_09.pdf
Sivak VV, Eickbusch A, Liu H, Royer B, Tsioutsios I, Devoret MH (2022) Model-free quantum control with reinforcement learning. Phys Rev X 12:011059. https://doi.org/10.1103/PhysRevX.12.011059
Skolik A, Jerbi S, Dunjko V (2022) Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning. Quantum 6:720. https://doi.org/10.22331/q-2022-05-24-720
Sohl-Dickstein J, Weiss E, Maheswaranathan N, Ganguli S (2015) Deep unsupervised learning using nonequilibrium thermodynamics. In: Proc. 32nd int conf mach learn. https://proceedings.mlr.press/v37/sohl-dickstein15.html
Sriarunothai T, Wölk S, Giri GS, Friis N, Dunjko V, Briegel HJ, Wunderlich C (2018) Speeding-up the decision making of a learning agent using an ion trap quantum processor. Quantum Sci Technol 4(1):015014. https://doi.org/10.1088/2058-9565/aaef5e
Sutton RS, Barto AG (2018) Reinforcement learning: An introduction. 2nd Edition. A Bradford Book, Cambridge, MA, USA. http://incompleteideas.net/book/the-book-2nd.html
Towers M, Kwiatkowski A, Terry J, Balis JU, Cola GD, Deleu T, Goulão M, Kallinteris A, Krimmel M, KG A, Perez-Vicente R, Pierré A, Schulhoff S, Tai JJ, Tan H, Younis OG (2025) Gymnasium: a standard interface for reinforcement learning environments. arxiv:2407.17032

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Adv neural inf process syst. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8(3):279–292. https://doi.org/10.1007/BF00992698

Wiedemann S, Hein D, Udluft S, Mendl C (2023) Quantum policy iteration via amplitude estimation and grover search – towards quantum advantage for reinforcement learning. arxiv:2206.04741

Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8:229–256. https://doi.org/10.1007/bf00992696

Wu S, Jin S, Wen D, Han D, Wang X (2025) Quantum reinforcement learning in continuous action space. Quantum 9:1660. https://doi.org/10.22331/q-2025-03-12-1660

Yu H, Zhao X (2024) Deep reinforcement learning with reward design for quantum control. IEEE Trans Artif Intell 5(3):1087–1101. https://doi.org/10.1109/TAI.2022.3225256

Zhang X-M, Wei Z, Asad R, Yang X-C, Wang X (2019) When does reinforcement learning stand out in quantum control? a comparative study on state preparation. NPJ Quantum Inf 5(1):85. https://doi.org/10.1038/s41534-019-0201-8

Zhong H, Hu J, Xue Y, Li T, Wang L (2024) Provably efficient exploration in quantum reinforcement learning with logarithmic worst-case regret. arxiv:2302.10796

Ziegler DM, Stiennon N, Wu J, Brown TB, Radford A, Amodei D, Christiano P, Irving G (2020) Fine-tuning language models from human preferences. arxiv:1909.08593

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.