scientific reports

Check for updates

OPEN

A hybrid classical-quantum approach to speed-up Q-learning

A. Sannia[1,2], A. Giordano[3], N. Lo Gullo[1,4,5], C. Mastroianni[3] & F. Plastina[1,4]✉

We introduce a classical-quantum hybrid approach to computation, allowing for a quadratic performance improvement in the decision process of a learning agent. Using the paradigm of quantum accelerators, we introduce a routine that runs on a quantum computer, which allows for the encoding of probability distributions. This quantum routine is then employed, in a reinforcement learning set-up, to encode the distributions that drive action choices. Our routine is well-suited in the case of a large, although finite, number of actions and can be employed in any scenario where a probability distribution with a large support is needed. We describe the routine and assess its performance in terms of computational complexity, needed quantum resource, and accuracy. Finally, we design an algorithm showing how to exploit it in the context of Q-learning.

Quantum algorithms can produce statistical patterns which are hard to manipulate on a classical computer; in turns, they may, perhaps, help recognize patterns that are difficult to identify classically. To pursue this basic idea, a huge research effort is being put forward to speed up machine learning routines by exploiting unique quantum properties, such as superposition, coherence and entanglement[1,2].

Within the realm of machine learning, the Reinforcement Learning (RL) paradigm has gained attention in the last two decades[3,4], as, in a wide range of application scenarios, it allows modeling an agent that is able to learn and improve its behavior through rewards and penalties received from a not fully known environment. The agent, typically, chooses the action to perform by sampling a probability distribution that mirrors the expected returns associated to each of the actions performed, conditioned to the state of the environment. The aim of the RL procedure is to maximize the total reward that corresponds to the achievement of a given task. This is obtained by devising a stochastic strategy to train the agent in performing a series of actions, each picked from a given set, which maximizes the total reward. The final output of the RL is a conditional probability distribution that correlates the state of the environment with the action to be taken by the agent to modify its state.

It turns out that the RL performances can be improved by the use of quantum routines, as recently reviewed in[5]. To date, various promising proposals have been put forward that exploit quantum accelerators to speed-up RL, including, e.g., the speed-ups of projective simulations[6–8], quantum models for RL policies with quantum circuits[9] and Boltzman machines[10,11], applications in measurement-based adaptation protocols[12,13], and their implementations in photonic platforms[14,15]. Moreover, in the same context, it is worth mentioning that some strategies based on the Grover's algorithm[16] have been proposed to generate the action probability distribution of a learning agent[17,18], which are suitable when the number of actions is finite.

Along this line, we introduce a novel algorithm that differentiates from the other Grover's algorithm-based approaches mainly for our exploitation and coordination of the actions of multiple oracles that are associated with action subsets. As we will show below, this allows us to tune the probability distributions of the subsets in the exploration and exploitation phases. Moreover, our method generalizes those mentioned before as it is able to approximate, in principle, any desired probability distribution, thus overcoming the existing limitations in applying the standard Grover's algorithm. Furthermore, our procedure does not require a prior knowledge of the probabilities to assign, as assumed in previous works[19,20]. Following the classification proposed in[21], our algorithm falls into the CQ framework, with a classical generating system and a quantum data processing device.

Specifically, in this work, we first introduce a routine to encode and update a probability distribution onto a quantum register and then we show how to embody it into a Q-learning based RL algorithm. The actions are clustered in a predetermined number of subsets (classes), each associated to a range with a minimum and a maximum value of the expected reward. The cardinality of each class is evaluated in due course, through a procedure built upon well-known quantum routines, i.e., quantum oracle and quantum counting[22,23]. Once this

[1]Dipartimento di Fisica, Università della Calabria, 87036 Arcavacata di Rende, (CS), Italy. [2]Institute for Cross-Disciplinary Physics and Complex Systems (IFISC) UIB-CSIC, Campus Universitat Illes Balears, 07122 Palma de Mallorca, Spain. [3]ICAR-CNR, 87036 Rende, Italy. [4]INFN, gruppo collegato di Cosenza, Cosenza, Italy. [5]Quantum Algorithms and Software, VTT Technical Research Centre of Finland Ltd, Espoo, Finland. ✉email: francesco.plastina@fis.unical.it

information is obtained, a classical procedure is run to assign a probability to each subset, in accordance to any desired distribution, while the elements within the same class are taken to be equally likely. This allows one to tune probabilities, in order to, e. g., assign a larger chance to the actions included in the range with maximum value of the expected reward. The probability distribution can also be changed dynamically, in order to enforce exploration at a first stage (allowing some actions associated to a low probability to be chosen) and exploitation at a second stage (to restrict the search to actions having a higher likelihood of occurrence). The quantum routine presented here allows re-evaluating the values after examining all the actions that are admissible in a given state in a single parallel step, which is possible due to quantum superposition.

Besides the RL scenario, for which our approach is explicitly tailored, the main advantageous features of the routine could be also exploited in other contexts where one needs to sample from a probability distribution, ranging from swarm intelligence algorithms (such as Particle Swarm Optimization and Ant Colony Optimization[24]), to Cloud architectures (where the objective is to find an efficient assignment of virtual machines to physical servers, a problem that is known to be NP-hard[25]). After presenting the quantum routine in detail in section "Preparing a quantum probability distribution", we focus on its use in the RL setting in section "Improving reinforcement learning", to show that it is, in fact, tailored for the needs of RL with a large number of state-action pairs. In section "Additional features of the algorithm" we give an assessment of the advantages of our quantum accelerated RL over a pure classical algorithm and of the needed quantum resources. Then, a more technical section follows, discussing details of the probability encoding routine, and evaluating its complexity and precision (section "Details on the probability distribution encoding algorithm"). Finally, we draw some concluding remarks in section "Conclusions".

Preparing a quantum probability distribution

Here, we introduce the quantum routine which encodes a classical probability distribution into a Quantum Register (QR). Let us assume that we have a random variable whose discrete domain includes J different values, $\{x_j : j = 1, \ldots, J\}$, which we map into the basis states of a J-dimensional Hilbert-space. Our goal is to prepare a quantum state for which the measurement probabilities in this basis reproduce the random variable probability distribution: $\{p_{x_1}, p_{x_2}, \cdots, p_{x_J}\}$.

The quantum routine starts by initializing the QR as:

$$|\psi\rangle = \frac{1}{\sqrt{J}} \sum_{k=1}^{J} |x_k\rangle|1\rangle_a \equiv |\phi\rangle|1\rangle_a \tag{1}$$

where $|\phi\rangle$ is the homogeneous superposition of the basis states $\{|x_k\rangle\}$, while an ancillary qubit is set to the state $|1\rangle_a$. In our approach, the final state is prepared by encoding the probabilities related to each state sequentially, which will require $J - 1$ steps.

At the i-th step of the algorithm ($1 \leq i < J$), Grover's iterations[16] are used to set the amplitude of the $|x_i\rangle$ basis state to $a_i = \sqrt{p_{x_i}}$. In particular, we apply a conditional Grover's operator: $\mathbb{I} \otimes \hat{\mathbf{\Pi}}_a^{(0)} + \hat{\boldsymbol{G}}_i \otimes \hat{\mathbf{\Pi}}_a^{(1)}$, where $\hat{\boldsymbol{G}}_i = \hat{\boldsymbol{R}}\hat{\boldsymbol{O}}_i$ and $\hat{\mathbf{\Pi}}_a^{(y)}$ is the projector onto the state $|y\rangle_a$ of the ancilla ($y = 0, 1$). It forces the Grover unitary, $\hat{\boldsymbol{G}}_i$, to act only on the component of the QR state tied to the ('unticked') state $|1\rangle_a$ of the ancillary qubit. The operator $\hat{\boldsymbol{R}} = 2|\phi\rangle\langle\phi| - \mathbb{I}$ is the reflection with respect to the uniform superposition state, whereas the operator $\hat{\boldsymbol{O}}_i = \mathbb{I} - 2|x_i\rangle\langle x_i|$ is built so as to flip the sign of the state $|x_i\rangle$ and leave all the other states unaltered. The Grover's operator is applied until the amplitude of $|x_i\rangle$ approximates a_i to the desired precision (see section "Details on the probability distribution encoding algorithm"). The state of the ancilla is not modified during the execution of the Grover's algorithm.

The i-th step ends by ensuring that the amplitude of $|x_i\rangle$ is not modified anymore during the next steps. To this end, we 'tick' this component by tying it to the state $|0\rangle_a$. This is obtained by applying the operator $\hat{\boldsymbol{F}}_i = |x_i\rangle\langle x_i| \otimes \hat{\boldsymbol{X}} + \left(\mathbb{I} - |x_i\rangle\langle x_i|\right) \otimes \mathbb{I}$, whose net effect is:

$$\hat{\boldsymbol{F}}_i|x_k\rangle|y\rangle_a = \begin{cases} |x_k\rangle \otimes \hat{\boldsymbol{X}}|y\rangle_a, & \text{if } k = i \\ |x_k\rangle|y\rangle_a, & \text{otherwise} \end{cases}$$

Where $\hat{\boldsymbol{X}}$ is the NOT-gate, with $1 \leq i \leq J$, $y \in \{0, 1\}$.

After the step i, the state of the system is:

$$|\psi\rangle = \sum_{j=1}^{i} a_j|x_j\rangle \otimes |0\rangle_a + b_i|\beta_i\rangle|1\rangle_a.$$

The state $|\beta_i\rangle$ has in general non-zero overlap with all of the basis states, including those states, $|x_1\rangle, \cdots, |x_{i-1}\rangle$, whose amplitudes have been updated previously. This is due to the action of the reflection operator $\hat{\boldsymbol{R}}$, which outputs a superposition of all the J basis states. However, this does not preclude us from extracting the value of the probability of the random variable correctly, thanks to the ancillary qubit. Indeed, at the end of the last step, the ancilla is first measured in the logical basis. If the outcome of the measurement is 0, then we can proceed measuring the rest of the QR to get one of the first $J - 1$ values of the random variable with the assigned probability distribution. Otherwise, the output is set to x_J, since the probability of getting 1 from the measurement of the ancilla is p_{x_J} due to the normalization condition ($p_{x_J} = a_J^2 = 1 - \sum_{k=1}^{J-1} a_k^2$).

The procedure can be generalized to encode a random distribution for which N elements, with $N > J$, are divided into J sub-intervals. In this case, a probability is not assigned separately to every single element; but, rather, collectively to each of the J sub-intervals, while the elements belonging to the same sub-interval are

assigned equal probabilities. This is useful to approximate a random distribution where the number of elements, N, is very large. In this case, the QR operates on an N-dimensional Hilbert space, and Grover's operators are used to amplify more than one state at each of the $J-1$ steps. For example, in the simplest case, we can think of having a set of J Grover operators, each of which acts on N/J basis states. In the general case, though, each Grover's operator could amplify a different, and a priori unknown, number of basis states. This case will be discussed in the following, where this quantum routine is exploited in the context of RL.

Improving reinforcement learning

We will now show how the algorithm introduced above can be exploited in the context of RL, and, specifically, in the Q-learning cycle. Figure 1 provides a sketch emphasising the part of the cycle that is involved in our algorithm. Our objective, in this context, is to update the action probabilities, as will be clarified in the rest of this section. An RL algorithm can be described in terms of an abstract *agent* interacting with an *environment*. The environment can be in one of the *states* s that belong to a given set S whereas the agent is allowed to perform *actions* picked from a set A_s, which, in general, depends on s. For each state $s \in S$, the agent chooses one of the allowed actions, according to a given policy. After the action is taken, the agent receives a *reward* r from the environment and its state changes to $s' \in S$. The reward is used by the agent to understand whether the action has been useful to approach the goal, and then to learn how to adapt and improve its behavior. Shortly, the higher the value of the reward, the better the choice of the action a for that particular state s. In principle, this behavior, or *policy*, should consist in a rule that determines the best action for any possible state. An RL algorithm aims at finding the optimal policy, which maximizes the overall reward, i.e., the sum of the rewards obtained after each action. However, if the rewards are not fully known in advance, the agent needs to act on the basis of an estimate of their values.

Among the various approaches designed to this end, the so called Q-Learning algorithm[4] adopts the Temporal Difference (TD) method to update the $Q(s, a)$ value, i.e., an estimation of how profitable is the choice of the action a when the agent is in the state s.

When choosing an action, at a given step of the algorithm, two key factors need to be taken into account: *explore* all the possible actions, and *exploit* the actions with the greatest values of $Q(s, a)$. As it is common, we resort to a compromise between exploration and exploitation by choosing the new action through a random distribution, defined so that the probability of choosing the action a in the state s mirrors $Q(s, a)$. For example, one could adopt a Boltzmann-like distribution $P(a|s) = e^{Q(s,a)/T}/Z$, where Z is a normalising factor, while the T parameter can vary during the learning process (with a large T in the beginning, in order to favor exploration, and lower T once some experience about the environment has been acquired, in order to exploit this knowledge and give more chances to actions with a higher reward).

A severe bottleneck in the performance of a TD training algorithm arises when the number of actions and/or states is large. For example, in the chess game, the number of states is $\sim 10^{120}$ and it is, in fact, impossible to deal with the consequent huge number of $Q(s, a)$ values. A workaround is to use a function $Q^*_{\vec{\theta}}(s, a)$ that *approximates* the values of $Q(s, a)$ obtained by the TD rule and whose properties depend upon a (small) set of free parameters $\vec{\theta}$ that are updated during the training. This approach showed its effectiveness in different *classical* approaches as, for example, in Deep Q-Learning[26–30], where the $Q^*_{\vec{\theta}}(s, a)$ function is implemented by means of a neural network whose parameters are updated in accordance with the experience of the agent.

In the quantum scenario, this approach turns out to be even more effective. Indeed, it is possible to build a parameter-dependent quantum circuit that implements $Q^*_{\vec{\theta}}(s, a)$; an approach that has been adopted by recent studies on near-term quantum devices[31–35]. This circuit allows us to evaluate the function $Q^*_{\vec{\theta}}(s, a)$ in a complete *quantum parallel* fashion; i.e., in one shot for all the admissible actions in a given state. With this approach, it is possible to obtain a quantum advantage in the process of building the probability distribution for the actions, using the algorithm presented in section "Preparing a quantum probability distribution". To achieve a significant quantum speed-up, and reduce the number of required quantum resources, thus making our algorithm suitable for near term NISQ processing units, we do not assign a probability to every action; but, rather, we aggregate actions into classes (i.e., subsets) according to their probabilities, as explained in the following. Let us consider the minimum (m) and maximum (M) of the $Q^*_{\vec{\theta}}(s, a)$ values, and let us divide the interval $[m, M]$ into

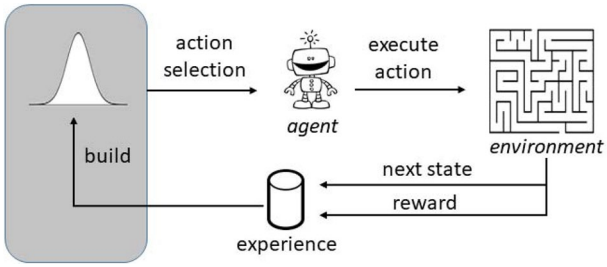

Figure 1. Sketch of the Q-learning cycle. An agent is schematically represented, performing a Q-learning cycle; namely, taking an action to solve a problem, getting back a reward and possibly changing its state and, finally, building a new probability distribution (by using the algorithm we are proposing), from which the next action is extracted.

J (non-overlapping, but not necessarily equal) sub-intervals I_j with $1 \leq j \leq J$. For a given state s, we include the action $a \in A_s$ in the class C_j if $Q^*_\theta(s,a) \in I_j$ (so that, $A_s = \bigcup_j C_j$).

The probability of each sub-interval, then, will be determined by the sum of the Q^*-values of the corresponding actions, $\sum_{a \in C_j} Q^*_\theta(s,a)$. All of the actions in C_j, will be then considered equally probable.

In this way, our algorithm requires only $J-1$ steps, each of which is devoted to amplify the actions belonging to one of the $J-1$ classes (while the J-th probability is obtained by normalization). Furthermore, we can also take advantage of the aggregation while encoding the probability distributions onto the QR: in this case, indeed, we can use predetermined Grover's oracles, each devoted to amplify the logical states corresponding to the actions belonging to a given C_j.

For the offloading of the distribution-update routine onto a quantum processor as a part of the Q-learning procedure, we need two QRs: $\mathcal{A}$ and $\mathcal{I}$, which encode the actions and the sub-intervals, respectively. These registers need $\lceil log_2(\max_s |A_s|) \rceil$ and $\lceil log_2(J) \rceil$ qubits, respectively. Let us consider the class $C_j = \{a \in A_s : Q^*(s,a) \in I_j\}$. Our goal is to assign to each action $a \in C_j$ a probability p_j, based on the sub-interval j, using the routine presented in section "Preparing a quantum probability distribution". The distribution building process starts by preparing the following uniform superposition:

$$|\psi_s\rangle = \frac{1}{\sqrt{|A_s|}} \sum_{a \in A_s} |a\rangle_{\mathcal{A}} |0\rangle_{\mathcal{I}},$$

where $|0\rangle_{\mathcal{I}}$ is the initial state of the register $\mathcal{I}$. In order to apply our algorithm, we need $J-1$ oracles $\hat{\boldsymbol{O}}_j$, one for each given I_j. To obtain these oracles, it is first necessary to define an operator $\hat{\boldsymbol{J}}$ that records the sub-interval j_a to which the value $Q^*_\theta(s,a)$ belongs. Its action creates correlations between the two registers by changing the initial state of the $\mathcal{I}$- register as follows:

$$\hat{\boldsymbol{J}} |a\rangle_{\mathcal{A}} |0\rangle_{\mathcal{I}} = |a\rangle_{\mathcal{A}} |j_a\rangle_{\mathcal{I}}.$$

To complete the construction of the oracles, we need to execute two unitaries: (i) the operator $\hat{\boldsymbol{O}}'_{j_a} = \mathbb{I}_{\mathcal{A}} \otimes (\mathbb{I}_{\mathcal{I}} - 2|j_a\rangle\langle j_a|)$, which flips the phase of the state $|j_a\rangle$ of the $\mathcal{I}$-register; and, (ii) the operator $\hat{\boldsymbol{J}}^\dagger$, which disentangles the two registers. The effective oracle operator entering the algorithm described in the previous section is then defined as $\hat{\boldsymbol{O}}_j = \hat{\boldsymbol{J}}^\dagger \hat{\boldsymbol{O}}'_j \hat{\boldsymbol{J}}$, its net effect being

$$\hat{\boldsymbol{O}}_j |a\rangle_{\mathcal{A}} |0\rangle_{\mathcal{I}} = \begin{cases} -|a\rangle_{\mathcal{A}} |0\rangle_{\mathcal{I}}, & \text{if a } \in C_j \\ |a\rangle_{\mathcal{A}} |0\rangle_{\mathcal{I}}, & \text{otherwise} \end{cases}$$

Eventually, we apply the reflection about average $\hat{\boldsymbol{R}}$ on the register $\mathcal{A}$, thus completing an iteration of the Grover operator. Let us notice that whereas the operator $\hat{\boldsymbol{R}}$ acts on the register of the actions, $\hat{\boldsymbol{O}}'_j$ acts on the register of the classes and on the Grover ancilla. This is shown schematically in Fig. 2.

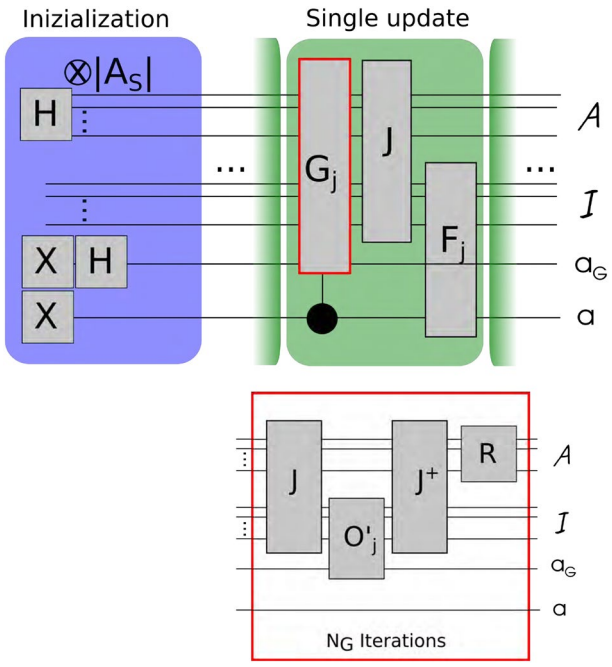

Figure 2. Quantum routine. Schematic representation of the quantum routine for the update of a classical probability distribution on the quantum register. After the initialization of the register the $J-1$ updates are performed to store the values of the probability distribution for each of the J classes for each state s. It shows also the single iteration of the modified Grover algorithm.

If the cardinality of each C_j is not decided from the beginning, in order to evaluate the right number of Grover iterations to be executed, we need to compute it (see section "Optimal number of iterations of the Grover algorithm" for details). This number of actions can be obtained, for each C_j, by running the quantum counting algorithm associated with $\hat{\mathbf{O}}_j$ (and before its action). It is then possible to apply the routine of section "Preparing a quantum probability distribution" in order to build the desired probability distribution. After the quantum state of the $\mathcal{A}$-register is obtained, the agent will choose the action measuring its state. Then, according to the outcome of the environment, it will update the $\vec{\theta}$ values classically, thus changing the behaviour of the operator $\hat{\boldsymbol{J}}$.

We summarize the key steps of our quantum enhanced RL algorithm in the box below, where classical and quantum operations are denoted as (C) and (Q), respectively. A schematic picture of the amplitude distribution upload circuit is depicted in Fig. 2 where we show the needed resources as well as the main gates required.

Scheme of the hybrid algorithm

Initialize $\vec{\theta}$ and start from state s (C)
Execute the cycle:
- Build the sub-intervals, classes and the quantum circuits for the oracles $\hat{\mathbf{O}}_i = \hat{\boldsymbol{J}}^\dagger \hat{\mathbf{O}}_i' \hat{\boldsymbol{J}}$ (C)
- Use quantum counting on $\hat{\mathbf{O}}_i$ to compute the number of actions belonging to each sub-interval (Q)
- Compute the number of Grover iterations for each class (C)
- Build the probability distribution for the admissible actions in s (Q)
- Measure, obtain an action and execute it (C)
- Get the new state s' and the reward r (C)
- Update $\vec{\theta}$ (C)

Additional features of the algorithm

The quantum enhanced RL algorithm presented above resorts to quantum acceleration to remove bottlenecks of classical approaches. We provide, here, an assessment of the advantages obtained.

In the case of a finite and yet large number of actions $|A_s| \gg 1\ \forall s$, one has that, classically, for a given state s, the number of calls to the function $Q^*_{\vec{\theta}}(s,a)$ increases asymptotically as $\mathbf{O}(|A_s|)$. Conversely, with our quantum protocol the number of calls of $\hat{\boldsymbol{J}}$, and therefore of $Q^*_{\vec{\theta}}(s,a)$, is asymptotically $\mathbf{O}(\sqrt{|A_s|})$. Nonetheless, the larger the number of actions, the lower the bound on the error, which is of $\mathbf{O}(1/\sqrt{|A_s|})$ (see section "Optimal number of iterations of the Grover algorithm" for details). Thus, as it is the case for all of the quantum algorithms based on Grover's, we obtained a quadratic speed-up over the classical algorithm of updating a probability distribution.

Our strategy of discretizing the $Q^*_{\vec{\theta}}(s,a)$ values into bins affects also the reinforcement learning procedure, both from the point of view of the accuracy with which we are reproducing the desired probability distribution, and for the exploitation-exploration interplay. Let ϵ be the error introduced by the discretization with respect to the target probability distribution. In general, it may depend on features of the target distribution which are independent of the total number of actions and of bins. However, for a rough quantitative estimate, we can take it to scale proportionally to the size of the bins, obtaining the upper bound $\epsilon \sim \frac{M-m}{J}$, as a direct consequence of the rectangle method accuracy to approximate the integral. In fact, it is important that in the process of the definition of sub-intervals, the maximum (M) and the minimum (m) values of $Q^*_{\vec{\theta}}(s,a)$ are computable in advance by means of well-established quantum routines that do not increase the complexity of our procedure[36,37].

The discretization has also a direct impact on the number of calls needed to update the values of the probability distribution. Indeed, the larger the number of bins the more the calls to Grover oracle, which scale linearly with the number of bins (again, see section "Optimal number of iterations of the Grover algorithm" for details). The total computational time for the training is, in general, strictly problem dependent. It depends on the chosen cost function as well as other details of the RL. As such, the size of the bins, although not estimable *a priori*, is yet another parameter that can be used to lower the convergence time of the learning.

Finally, let us quantify the quantum resources needed in order to implement our algorithm. In the ideal noiseless case, as reported above, the qubits needed to implement our strategy can be organized into two registers: $\mathcal{A}$ and $\mathcal{I}$. The first one has to encode, in each case, all the actions of the reinforcement learning protocol and, as a consequence, it will require $log_2(\max_s |A_s|)\rceil$ qubits. The second one is, instead, devoted to encoding the J classes used to discretize the Q^*-values, needing $\lceil log_2(J)\rceil$ qubits. Moreover, ancillary qubits can be necessary to implement the oracle $\hat{\mathbf{O}}_j$, but their number is strictly dependent on the specific problem. As a function of the number of actions, we can conclude that we have a logarithmic scaling of the number of required qubits.

In a more realistic scenario, in which decoherence affects the operations, one possibility to preserve the advantage is to resort to error correction algorithms. This unavoidably results in an increase in the amount of quantum resources. In our algorithm, the most sensitive part to noise is the amplification of the amplitudes via Grover's algorithm on each interval. In Ref.[38], an extensive analysis of the effect of noise on Grover search has been reported. The authors show that a [[7, 1]] Steane code[39] is an effective strategy to correct errors in the Grover's search algorithm. Allowing for a lower gain, the [[15, 7]] QBCH code[40] can reduce the total amount of resource needed. As suggested in Ref.[38], a hybrid approach can be considered, as a possible compromise between the two methods.

Details on the probability distribution encoding algorithm

In this more technical section, we provide some details on the routine presented in section "Preparing a quantum probability distribution", which are important for its actual implementation. Specifically, we address the problems of (i) how to compute the optimal number of iterations of the Grover algorithm in order to store a single instance of the probability distribution, (ii) how to compute quantities which are needed to link the update of the values of the probability distribution on different sub-intervals from one step to another, and, finally, (iii) evaluate its complexity.

Optimal Number of iterations of the Grover algorithm. In order to compute the optimal number of iterations in a single step we exploit the results reported in Ref.[41], where the Grover algorithm has been generalized to the case of an initial non-uniform distribution and define the following quantities :

$$\bar{K}^{(i)}(t) = \frac{1}{r_i}\sum_{j=1}^{r_i} k_j^{(i)}(t)$$

$$\bar{L}^{(i)}(t) = \frac{1}{N-r_i}\sum_{j=r_i+1}^{N} l_j^{(i)}(t)$$

where t is the number of Grover iteration already performed, N is the dimension of the Hilbert space (in our context it is the total number of actions : $N = |A_s|$) , $\{k_j^{(i)}(t)\}$ are the coefficients of the r_i basis states that will be amplified by the Grover iterations at the step i of the algorithm (in our application $r_i = |C_i|$), and $\{l_j^{(i)}(t)\}$ are the coefficients of all the other basis states, while $\bar{K}^{(i)}(t)$ and $\bar{L}^{(i)}(t)$ are their averages and we have labeled them with the step-counting variable i. Let's assume now that only one basis state at a time is amplified by Grover iterations, namely $r_i = 1$. Applying the results in Ref.[41] to our case we obtain:

$$\begin{aligned}\bar{K}^{(i)}(t) &= k^{(i)}(0)\cos(wt) + \bar{L}^{(i)}(0)\sqrt{N-1}\sin(wt)\\ \bar{L}^{(i)}(t) &= \bar{L}^{(i)}(0)\cos(wt) - k^{(i)}(0)\sqrt{\frac{1}{N-1}}\sin(wt),\end{aligned} \tag{2}$$

where $w = 2\arcsin(\sqrt{1/N})$. With the first of these equations we can compute the number of steps $t_f^{(i)}$ needed to set the coefficient $k(t)$ to the desired value with the wanted precision so as to bring the value of the probability distribution to $P(x_i) = |b_i\ k^{(i)}(t_f^{(i)})|^2$. Notice that we need the values of $k^{(i)}(0)$ and $\bar{L}^{(i)}(0)$ to perform this calculation, which values can be extracted from the form of the global state at the previous step and specifically from the last iteration of the Grover algorithm.

Variation of quantum state within the Grover iterations. Let us consider the quantum state of the action-register plus the ancilla system at a given iteration t of the Grover algorithm at a given step i:

$$|\psi(t)\rangle = \sum_{k=1}^{i-1} a_k|x_k\rangle|0\rangle_a + b_i|\beta_i(t)\rangle|1\rangle_a$$

where $b_i = (1-\sum_{k=1}^{j-1} a_k^2)^{1/2}$ and $t = 0$ at the beginning of the Grover algorithm. Let us write the state $|\beta_i(t)\rangle$ in a form which highlights its decomposition into three sets of basis states:

$$|\beta_i(t)\rangle = k^{(i)}(t)|x_i\rangle + \sum_{k=1}^{i-1} l_k^{(i)}(t)|x_k\rangle + \sum_{k=i+1}^{N}\alpha^{(i)}(t)|x_k\rangle, \tag{3}$$

where we have made explicit the dependence of the coefficients of the decomposition on the step i, for this will be useful in the following. The $|x_i\rangle$ is the one we want to use to encode the value of the probability distribution at the current step i of our algorithm, the $\{x_j\}$ with $j \in [1, i-1]$ basis states that are generated by the reflection operation around the mean and whose amplitudes have been updated in the previous $i-1$ steps, and the $\{x_j\}$ with $j \in [i+1, N]$ basis state, all having the same amplitude $\alpha^{(i)}(t)$ for all the operations performed up to this point did not change them.

Using this expression, it is possible to derive a recursive relation to compute $\alpha^{(i)}(t)$ iteratively as a function of $\bar{L}^{(i)}(t)$ and $k^{(i)}(t)$. As we shall see below this will be useful in order to compute the initial $k^{(i+1)}(0)$,$\bar{L}^{(i+1)}(0)$ and $\alpha^{(i+1)}(0)$ for the next step. To find this recursive relation let us first apply the Grover operator onto $|\beta_i(t)\rangle|1\rangle_a$ and then project onto any of the states $\{x_{i+1}, x_{i+2}, \cdots, x_N\}$. Without loss of generality we choose x_{i+1}:

$$\begin{aligned}\alpha^{(i)}(t+1) &= \langle x_{i+1}|\hat{\boldsymbol{R}}\hat{\boldsymbol{O}}_i|\beta_i(t)\rangle\\ &= \langle x_{i+1}|\Big(2|\phi\rangle\langle\phi| - \mathbb{I}\Big)\\ &\quad\times\left(-k^{(i)}(t)|x_i\rangle + \sum_{k=1}^{i-1} l_k^{(i)}|x_k\rangle + \sum_{k=i+1}^{N}\alpha^{(i)}(t)|x_k\rangle\right)\\ &= \langle x_{i+1}|\Big\{2\Big(-\frac{1}{\sqrt{N}}k^{(i)}(t) + \frac{N-1}{\sqrt{N}}\bar{L}^{(i)}(t)\Big)|\phi\rangle\\ &\quad + k^{(i)}(t)|x_i\rangle - \sum_{k=1}^{i-1} l_k^{(i)}|x_k\rangle - \sum_{k=i+1}^{N}\alpha^{(i)}(t)|x_k\rangle\Big\}\\ &= \frac{2}{N}\bar{L}^{(i)}(t)(N-1) - \frac{2}{N}k^{(i)}(t) - \alpha^{(i)}(t)\end{aligned} \tag{4}$$

Using Eqs. (2) and (4), as well as the initial values of $\alpha(0)$, $k(0)$ and $\bar{L}(0)$, at the beginning of the Grover iterations, together with the number of iterations $t_f^{(i)}$, with a simple *classical* iterative procedure, we can compute the final values of $\alpha^{(i)}(t_f^{(i)})$, $k^{(i)}(t_f^{(i)})$ and $\bar{L}^{(i)}(t_f)$. This concludes one step of the embedding algorithm.

Linking two consecutive steps of the distribution-encoding algorithm. Let us see how to use the $\alpha^i(t_f^{(i)})$, $k^i(t_f^{(i)})$ and $\bar{L}^i)(t_f^{(i)})$ computed at the end of step i to obtain the values $\alpha_{(i+1)}(0)$, $k^{(i+1)}(0)$ and $\bar{L}^{(i+1)}(0)$, which are needed in the following step $i+1$.

We first consider the global state at the end of the step i before applying the $\hat{\boldsymbol{F}}_i$ operator to mark the state $|x_i\rangle$ whose amplitude has just been updated:

$$\left|\psi(t_f^{(i)})\right\rangle = \sum_{j=1}^{i-1} a_j|x_j\rangle|0\rangle_a + b_i\left|\beta_i(t_f^{(i)})\right\rangle|1\rangle_a$$

where

$$\left|\beta_i(t_f^{(i)})\right\rangle = k^{(i)}(t_f^{(i)})|x_i\rangle + \sum_{k\neq i} l_k(t_f^{(i)})|x_k\rangle. \tag{5}$$

After applying $\hat{\boldsymbol{F}}_i$ we obtain the initial global state which will be the seed of the Grover iterations at the step $i+1$:

$$|\psi(0)\rangle = \sum_{k=1}^{i-1} a_k|x_k\rangle|0\rangle_a + a_i|x_i\rangle|0\rangle_a + b_{i+1}|\beta_{i+1}(0)\rangle|1\rangle_a, \tag{6}$$

where

$$|\beta_{i+1}(0)\rangle = k^{(i+1)}(0)|x_{i+1}\rangle + \sum_{k\neq i+1} l_k^{(i)}(0)|x_k\rangle$$

In this new state, the coefficient b_{i+1} ensures that $|\beta_{i+1}(0)\rangle$ is unit. By looking at the coefficient of $|x_i\rangle$ in both (5) and (6), it is easy to see that $a_i = b_i k^{(i)}(t_f^{(i)})$. In the same way, we can compare the coefficients that in (6) appear with the state $|1\rangle_a$ of the ancilla with the corresponding components in (5):

$$\begin{aligned} b_{i+1}|\beta_{i+1}(0)\rangle|1\rangle_a &= b_i\left(\left|\beta_i(t_f^{(i)})\right\rangle - k^{(i)}(t_f^{(i)})|x_i\rangle\right)|1\rangle_a \\ &= b_i\left(\sum_{k\neq i} l_k^{(i)}(t_f^{(i)})|x_k\rangle\right)|1\rangle_a \end{aligned}$$

It follows that:

$$|\beta_{i+1}(0)\rangle = \frac{b_i}{b_{i+1}}\sum_{k\neq i} l_k^{(i)}(t_f^{(i)})|x_k\rangle$$

Finally we obtain that:

$$\begin{aligned} \alpha^{(i+1)}(0) = k^{(i+1)}(0) &= \frac{b_i}{b_{i+1}}\alpha^{(i)}(t_f^{(i)}) \\ \bar{L}^{(i+1)}(0) &= \frac{\sum_{k\neq i+1} l_k^{(i+1)}(0)}{N-1} = \frac{b_i}{b_{i+1}}\frac{\sum_{k\neq i+1} l_k^{(i)}(t_f^{(i)})}{N-1} \\ &= \frac{b_i}{b_{i+1}}\left(\bar{L}^{(i)}(t_f^{(i)}) - \frac{\alpha(t_f^{(i)})}{N-1}\right) \end{aligned}$$

It is also possible to generalize these results in the case in which the states to be updated are superposition of more than one basis states. Let us assume that we have $r_i > 1$. In this case the general relations for $K^{(i)}(t)$ and $L^{(i)}(t)$ read :

$$\begin{aligned} \bar{K}^{(i)}(t) &= \bar{K}^{(i)}(0)\cos(wt) + \bar{L}^{(i)}(0)\sqrt{\frac{N-r_i}{r_i}}\sin(wt) \\ \bar{L}^{(i)}(t) &= \bar{L}^{(i)}(0)\cos(wt) - \bar{K}^{(i)}(0)\sqrt{\frac{r_i}{N-r_i}}\sin(wt). \end{aligned}$$

with $w = 2\arcsin(\sqrt{r_i/N})$. Since all the marked states have the same probability, we conclude that the probability of a single state is $\bar{K}^{(i)}(t)^2$. Moreover the expression for $\bar{L}^{(i+1)}(0)$ now is:

$$\bar{L}^{(i+1)}(0) = \frac{b_i}{b_{i+1}}\left(\frac{(N-r_i)\bar{L}^{(i)}(t_f^{(i)}) - r_{i+1}\alpha^{(i)}(t_f^{(i)})}{N-r_{i+1}}\right),$$

The other updating rules will be the same. It is important to underline that in this general case it is necessary to know in advance the number of states related to each oracle. As mentioned above, this can be set from the beginning or achieved by means of a quantum counting procedure.

Complexity and precision. In order to compute the complexity of the algorithm, we start from the observation, derived in Ref.[41], that the optimal number of Grover's iterations for a given step i is upper bounded by :

$$N_I^{(i)} = \frac{\frac{\pi}{2} - \arctan\left(\frac{\bar{K}(0)}{\bar{L}(0)}\sqrt{\frac{r_i}{N-r_i}}\right)}{\arccos\left(1-2\frac{r_i}{N}\right)}.$$

Expanding $N_I^{(i)}$ to the leading-order in our working assumptions ($N >> 1$), we obtain:

$$N_I^{(i)} \simeq -\frac{1}{2}\frac{\bar{K}(0)}{\bar{L}(0)} + \frac{\pi}{4}\sqrt{\frac{N}{r_i}}.$$

From this expression, we can conclude that the initial conditions of the state can only reduce the optimal number of calls of Grover's operators because we are dealing only with positive amplitudes, and we have $\bar{K}(0), \bar{L}(0) \geq 0$. It is worth to note that in our case we want to set the amplitude of each action not to the maximum value but to the value that is determined by a probability distribution, therefore the number of Grover's iterations is typically much lower than the upper bound given above.

As explained in section "Improving reinforcement learning", the number of times that the Grover procedure has to be executed is equal to the number of sub-intervals (J) chosen, and so the total complexity is:

$$\mathbf{O}\left(J\sqrt{N}\right) = \mathbf{O}\left(\sqrt{N}\right),$$

where we took into account that for a given state s, $N = |A_s|$, the complexity is equal to $\mathbf{O}\left(\sqrt{|A_s|}\right)$.

A useful quantity to compute in a practical implementation of the algorithm is the precision ΔP. It is the variation of the probability associated to an action between two consecutive iterations of the amplitude amplification (from t to $t+1$). This can be used as a criterion to stop the iterations. It can be quantified as follows:

$$\begin{aligned}
\Delta P &= |\langle x_i|\langle 1|_a|\psi(t+1)\rangle|^2 - |\langle x_i|\langle 1|_a|\psi(t)\rangle|^2 \\
&= b_i^2\left(\bar{K}(t+1)^2 - \bar{K}(t)^2\right) \\
&= b_i^2\Big(\bar{K}(0)^2[\cos(w(t+1))^2 - \cos(wt)^2] \\
&\quad + \bar{L}(0)^2\left(\frac{N-r_i}{r_i}\right)[\sin(w(t+1))^2 - \sin(wt)^2] \\
&\quad + \bar{K}(0)\bar{L}(0)\sqrt{\frac{N-r_i}{r_i}}[\sin(2w(t+1)) - \sin(2wt)]\Big).
\end{aligned} \tag{7}$$

Recalling that we assumed $|A_s| >> 1$ and considering the upper bound case in which $t \sim \sqrt{|A_s|}$ and $r_1 << N$, we can expand ΔP to the leading-order in $1/|A_s|$, we have:

$$\begin{aligned}
\Delta P \sim b_i^2\Big(&\bar{K}(0)^2\frac{1}{\sqrt{|A_s|}} + \bar{L}(0)^2|A_s|\frac{1}{\sqrt{|A_s|}} \\
&+ \bar{K}(0)\bar{L}(0)\sqrt{|A_s|}\Big)
\end{aligned}$$

Interestingly, the precision is bounded from below and the bound is obtained for $b_i = 1$, $K(0) \sim |A_s|^{-1/2}$ and $\bar{L}(0) \sim |A_s|^{-1/2}$:

$$\Delta P \sim \frac{1}{\sqrt{|A_s|}}.$$

The quantity ΔP can also be seen as the minimum error on the probability update, no matter how many iterations we perform.

Conclusions

In our work, we presented a routine, based on the Grover's algorithm, to encode a probability distribution onto a quantum register with a quadratic speed-up improvement. This quantum routine can find several useful applications in the context of hybrid classical-quantum workflows. In this spirit, we have shown how to exploit it for the training of the Q-learning strategy. We have shown that this gives rise to a quadratic quantum speed up of the RL algorithm, obtained by the inclusion of our quantum subroutine in the stage of action selection of

the RL workflow for a large but finite number of actions. This effectively enables achieving a trade off between exploration and exploitation, thanks to the intrinsic randomness embodied by the extraction from a QR of the action to be performed and, also, to the possibility of dynamically changing the relationship between the action and their values (and, thus, their relative probabilities). In the classical case, the trade off between exploitation and exploration needs to be implemented as an extra control parameter, typically via a random variable and a user-defined threshold that manages the rate of acceptance of non-optimal stat-action pairs.

Finally, we stress once again that, with our procedure, we can use Grover's oracles, which are given once and for all if i) the minimum and maximum range of action values, and ii) the number of intervals in which this range is divided are specified in advance.

Data availability

All data generated or analysed during this study are included in this published article.

Received: 5 September 2022; Accepted: 6 March 2023
Published online: 08 March 2023

References

1. Biamonte, J. *et al.* Quantum machine learning. *Nature* **549**(7671), 195–202 (2017).
2. Dunjko, V., Taylor, J. M. & Briegel, H. J. Quantum-enhanced machine learning. *Phys. Rev. Lett.* **117**(13), 130501 (2016).
3. Kaelbling, L. P., Littman, M. L. & Moore, A. W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **4**, 237–285 (1996).
4. Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An Introduction* (MIT Press, 2018).
5. Meyer, N., *et al.* A survey on quantum reinforcement learning. arXiv:2211.03464 (2022).
6. Saggio, V. *et al.* Experimental quantum speed-up in reinforcement learning agents. *Nature* **591**(7849), 229–233 (2021).
7. Paparo, G. D., Dunjko, V., Makmal, A., Martin-Delgado, M. A. & Briegel, H. J. Quantum speedup for active learning agents. *Phys. Rev. X* **4**, 031002 (2014).
8. Sriarunothai, T. *et al.* Speeding-up the decision making of a learning agent using an ion trap quantum processor. *Quantum Sci. Technol.* **4**(1), 015014 (2018).
9. Jerbi, S., Gyurik, C., Marshall, S., Briegel, H. J. & Dunjko, V. Variational quantum policies for reinforcement learning arXiv:2103.05577 (2021).
10. Crawford, D., Levit, A., Ghadermarzy, N., Oberoi, J. S. & Ronagh, P. Reinforcement learning using quantum Boltzmann machines. *Quantum Inf. Comput.* **18**(1–2), 51–74 (2018).
11. Levit, A., Crawford, D., Ghadermarzy, N., Oberoi, J. S., Zahedinejad, E. & Ronagh, P. Free energy-based reinforcement learning using a quantum processor. arXiv:1706.00074 (2017).
12. Olivares-Sánchez, J., Casanova, J., Solano, E. & Lamata, L. Measurement-based adaptation protocol with quantum reinforcement learning in a Rigetti quantum computer. *Quantum Rep.* **2**(2), 293–304 (2020).
13. Shenoy, K. S., Sheth, D. Y., Behera, B. K. & Panigrahi, P. K. Demonstration of a measurement-based adaptation protocol with quantum reinforcement learning on the IBM Q experience platform. *Quantum Inf. Process.* **19**(5), 1–13 (2020).
14. Flamini, F. *et al.* Photonic architecture for reinforcement learning. *New J. Phys.* **22**(4), 045002 (2020).
15. Lamata, L. Quantum reinforcement learning with quantum photonics. *Photonics* **8**(2), 33. https://doi.org/10.3390/photonics8020033 (2021).
16. Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, 212–219 (1996).
17. Dong, D., Chen, C., Li, H. & Tarn, T.-J. Quantum reinforcement learning. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **38**(5), 1207–1220 (2008).
18. Li, J.-A. *et al.* Quantum reinforcement learning during human decision-making. *Nat. Hum. Behav.* **4**, 294–307, 03 (2020).
19. Grover, L. & Rudolph, T. Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv e-print*sarXiv:quant-ph/0208112 *(2002).*
20. Gilliam, A., *et al.*. Foundational patterns for efficient quantum computing. arXiv:1907.11513 (2019).
21. Schuld, M. & Petruccione, F. *Supervised Learning with Quantum Computers* Vol. 17 (Springer, 2018).
22. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition* 10th edn. (Cambridge University Press, 2011).
23. Brassard, G., HØyer, P. & Tapp, A. Quantum Counting. In *Lecture Notes in Computer Science*, 820–831 (1998).
24. Bonabeau, E., Dorigo, M. & Theraulaz, G. *Swarm Intelligence: From Natural to Artificial Systems* (Oxford University Press, 1999).
25. Mastroianni, C., Meo, M. & Papuzzo, G. Probabilistic consolidation of virtual machines in self-organizing cloud data centers. *IEEE Trans. Cloud Comput.* **1**(2), 215–228 (2013).
26. Hester, T., *et al.* Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).
27. Van Hasselt, H., Guez, A. & Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* **30**(1) (2016).
28. Dulac-Arnold, G., *et al.* Deep reinforcement learning in large discrete action spaces. *arXiv preprint*arXiv:1512.07679 *(2015).*
29. Weisz, G., Budzianowski, P., Su, P.-H. & Gašić, M. Sample efficient deep reinforcement learning for dialogue systems with large action spaces. *IEEE/ACM Trans. Audio Speech Lang. Process.* **26**(11), 2083–2097 (2018).
30. Andriotis, C. & Papakonstantinou, K. Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliab. Eng. Syst. Saf.* **191**, 106483 (2019).
31. Jerbi, S., Trenkwalder, L. M., Nautrup, H. P., Briegel, H. J. & Dunjko, V. Quantum enhancements for deep reinforcement learning in large spaces. *PRX Quantum* **2**(1), 010328 (2021).
32. Skolik, A., Jerbi, S. & Dunjko, V. Quantum agents in the gym: A variational quantum algorithm for deep q-learning Quantum. **6**, 720 (2022).
33. Chen, S.Y.-C. *et al.* Variational quantum circuits for deep reinforcement learning. *IEEE Access* **8**, 141 007-141 024 (2020).
34. He, Z., Li, L., Zheng, S., Li, Y. & Situ, H. Variational quantum compiling with double q-learning. *New J. Phys.* **23**(3), 033002 (2021).
35. Lockwood, O. & Si, M. Reinforcement learning with quantum variational circuit. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* **16**(1), 245–251 (2020).
36. Ahuja, A. & Kapoor, S. A quantum algorithm for finding the maximum. arXiv:quant-ph/9911082 (1999).
37. Dürr, C. & Høyer, P. A quantum algorithm for finding the minimum. *arXiv*arXiv:quant-ph/9607014 *(1996).*
38. Botsinis, P. *et al.* Quantum error correction protects quantum search algorithms against decoherence. *Sci. Rep.* **6**(1), 38095. https://doi.org/10.1038/srep38095 (2016).
39. Steane, A. M. Error correcting codes in quantum theory. *Phys. Rev. Lett.* **77**, 793–797. https://doi.org/10.1103/PhysRevLett.77.793 (1996).

40. Grassl, M., Beth, T. & Pellizzari, T. Codes for the quantum erasure channel. *Phys. Rev. A* **56**(1), 33–38. https://doi.org/10.1103/physreva.56.33 (1997).
41. Biron, D., Biham, O., Biham, E., Grassl, M. & Lidar, D. A. Generalized Grover search algorithm for arbitrary initial amplitude distribution. In *Quantum Computing and Quantum Communications* (ed. Williams, C. P.) 140–147 (Springer, 1999).

Acknowledgements
This work was partially funded by the Italian MUR Ministry under the project PNRR National Centre on HPC, Big Data and Quantum Computing, PUN: B93C22000620006, and from the Spanish State Research Agency, through the QUARESC project (PID2019-109094GB-C21/AEI/ 10.13039/501100011033) and the Severo Ochoa and María de Maeztu Program for Centers and Units of Excellence in R &D (MDM-2017-0711), from CAIB through the QUAREC project (PRD2018/47).

Author contributions
All the authors conceived the idea, derived the technical results, discussed all stages of the project, and prepared the manuscript and the figure.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to F.P.

Reprints and permissions information is available at www.nature.com/reprints.

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