Quantum Reinforcement Learning via Policy Iteration

El Amine Cherrat[1], Iordanis Kerenidis[1, 2], and Anupam Prakash[2]

[1]Université de Paris, CNRS, IRIF, F-75006, Paris, France
[2]QC Ware, Palo Alto, USA and Paris, France

Abstract

Quantum computing has shown the potential to substantially speed up machine learning applications, in particular for supervised and unsupervised learning. Reinforcement learning, on the other hand, has become essential for solving many decision making problems and policy iteration methods remain the foundation of such approaches. In this paper, we provide a general framework for performing quantum reinforcement learning via policy iteration. We validate our framework by designing and analyzing: *quantum policy evaluation* methods for infinite horizon discounted problems by building quantum states that approximately encode the value function of a policy π; and *quantum policy improvement* methods by post-processing measurement outcomes on these quantum states. Last, we study the theoretical and experimental performance of our quantum algorithms on two environments from OpenAI's Gym.

Emails: cherrat@irif.fr, jkeren@irif.fr, anupamprakash1@gmail.com

1 Introduction

Reinforcement learning has had a great impact in decision making problems, in particular combined with artificial neural networks [1, 2]. Nevertheless, alternatives to neural networks are still needed for a number of different reasons, first, because the amount of data is expected to continue to grow along with its dimensionality, and, second, neural networks carry vulnerabilities that make them prone to adversarial attacks [3]. A possible alternative to deep learning for further improving machine learning can be found in quantum computing that has shown to be able to perform tasks beyond the reach of classical computing [4]. The field of quantum machine learning explores how to design and implement quantum algorithms that could enable machine learning that is faster, more expressive, or more explainable. Using quantum computers, a number of quantum machine learning algorithms have been published for supervised and unsupervised learning [5–11]. Here, we are interested in reinforcement learning and in particular the policy iteration algorithm [12]:

*Policy iteration is an algorithm that, given a Markov Decision Process,
generates a sequence of policies converging to the optimal policy in a finite number of steps.*

The research on quantum reinforcement learning is so far rather limited. The first approach was developed by Dong et al. [13] where a quantum environment using a superposition of states and actions is proposed, while temporal difference learning is used for policy evaluation and Grover techniques [14] for policy improvement. Cornelissen [15] proposed a quantum algorithm to evaluate the value function using a phase reward oracle and used quantum gradient estimation [16] to improve the policy. Ronagh [17] presents a quantum dynamic programming algorithm for solving the finite-horizon processes case. Several works in quantum reinforcement explore the use of variational circuits for value-based and policy-based algorithms [18–21]. More recent work from Wang et al. [22] combines quantum mean estimation with the quantum maximum searching algorithm to estimate the optimal policy and value function when a generative model for the environment is available. This approach provides a polynomial speedup over the classical value iteration algorithm but does not generalize to the case where such model is not available.

In this work, we define a general framework for quantum reinforcement learning based on quantum policy iteration, by extending the classical approach in [23]. We provide algorithms for performing quantum policy iteration and we extend them to the approximate case with linear value functions. Our *quantum policy iteration* algorithms alternate between two steps as their classical counterparts. The *quantum policy evaluation* step uses quantum linear system solvers to produce quantum states that approximately encode the policy value function. The *quantum policy improvement* step improves the actual policy based on measurements performed on these quantum states. In Section 3, we provide an in-depth analysis of our quantum policy iteration algorithm where we prove tight convergence bounds similar to the classical case, and provide efficient methods for building quantum access to the environment and policy parameters. In Section 4, we generalize our approach to the approximate case with linear value function approximation and provide a model-free implementation.

The quantum policy iteration methods we develop use quantum linear system solvers for evaluating the policies and hence the running time of these procedures depend explicitly on parameters of the matrices involved in these linear systems (for example the condition number, sparsity or rank) and also on how efficient it is to access these matrices in a quantum way (in other words constructing efficient block encodings).

In fact, we believe reinforcement learning is an advantageous case for quantum linear algebra precisely due to the character of the linear systems which are usually sparse and well-conditioned. Much of the effort in the paper is to provide explicit constructions of the block encodings for all cases, which enables to bound the parameters in the running time and have a clear idea of when to expect a quantum advantage. For example, we will see in Section 3 that for certain environments like the FROZENLAKE, mazes or other board games, the running time of our quantum method can be thought of as $\mathcal{O}(SA + \log(SA)/\epsilon^2)$, where S, A are the states and actions of the game, and ϵ is the accuracy for retrieving the solution from the quantum linear system solver, which one can compare with the $\mathcal{O}((SA)^\omega)$ running time of the classical linear system method. We also provide a similar comparison between running times of classical and quantum approximate policy iteration methods in Section 4. Last, in Section 5, we simulate our quantum algorithms for the FROZENLAKE and INVERTEDPENDULUM environments to show that our quantum algorithms converge well and can be considerably faster in practice, and we also describe how to build the necessary block encodings.

Overall, our methodology provides a general framework for infinite-horizon problems in the model-based and model-free case, and it encompasses many different ways of performing policy evaluation and improvement, including deep learning techniques, thus, enabling theoretical analysis of quantum reinforcement learning.

2   Preliminaries

2.1   Reinforcement learning

The aim of reinforcement learning[1] is to train an agent to discover the policy that maximizes the agent's performance in terms of the discounted future reward, while interacting with the environment, receiving only a reward signal. The agent can take actions in a set of possible actions based on a policy that maps each state with actions to take. This interaction is summarized in Figure 1.

More formally, we consider the infinite-horizon discounted decision problem with a *state set* $\mathcal{S}$ and a finite *action set* $\mathcal{A}$. At each time-step t, the agent receives a representation of the environment's *state* $s_t \in \mathcal{S}$, selects an *action* $a_t \in \mathcal{A}$ and receives a *reward* $r_t \in [0, 1]$. Denoting by $p(s, a, s')$ the probability that s' will occur and by $r(s, a)$ the average reward perceived after taking action a whilst in state s, the usual framework used to describe the environment's elements in reinforcement learning are *Markov Decision Processes* (MDP) which are fully defined by giving tuples of the form $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where $P = [p(s, a, s')]_{s,a,s'} \in \mathbb{R}^{S \times A \times S}$ is the transition matrix, $R = [r(s, a)]_{s,a} \in \mathbb{R}^{S \times A}$ is the reward vector and $\gamma \in (0, 1)$ is the discount factor. For the rest of the paper, we will denote by S the size of the state space $\mathcal{S}$, by A the size of the action space $\mathcal{A}$ and by Γ the effective time horizon:

$$\Gamma = \frac{1}{1 - \gamma}$$

---

[1]For a more detailed introduction to reinforcement learning, we recommend [12].
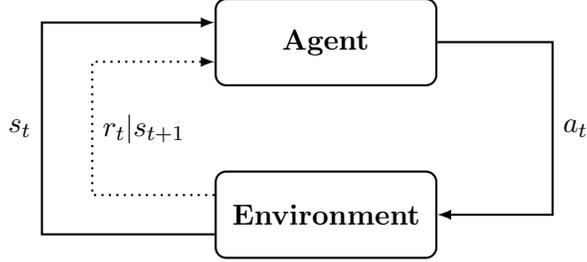
Figure 1: The agent–environment interaction [12].

The agent's behavior is modeled by some policy π, where $\pi(s, a)$ represents the probability of selecting action a given the state s. Moreover, for every policy π, we define its value function $Q^\pi \in \mathbb{R}^{S \times A}$ as:

$$Q^\pi(s, a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \Big| s_0 = s, a_0 = a, a_t \sim \pi(s_t), s_{t+1} \sim p(s_t, a_t) \Big]$$

which denotes the cumulative reward received by the agent when starting from (s, a) and playing according to π. The value function Q^π is the unique solution of the *Bellman equation* $\mathcal{T}^\pi Q^\pi = Q^\pi$ where $\mathcal{T}^\pi$ is the operator acting on $\mathbb{R}^{S \times A}$ such that:

$$\mathcal{T}^\pi Q(s, a) = r(s, a) + \gamma \sum_{s'a'} p(s, a, s') \pi(s', a') Q(s', a')$$

The goal in reinforcement learning is to find the optimal value function $Q^* = \sup_\pi Q^\pi$ over all policies π that maximizes the value function for all state-action pairs. A classical result about MDPs is the existence of a policy, referred to as the optimal policy π^* that reaches these optimal values such that $Q^{\pi^*} = Q^*$ and verifies $\pi^*(s) \in \arg\max\{Q^*(s, a) | a \in \mathcal{A}\}$ for every $s \in \mathcal{S}$.

2.2 Policy iteration

The idea of *Policy Iteration* (PI) is to build a sequence of deterministic policies $\{\pi_t\}_{t \in \mathbb{N}}$ that converges to the optimal policy π^* (Figure 2a). The algorithm is initialized with a random policy π_0 and iteratively alternates between two phases. The first phase, called *policy evaluation*, computes the value function Q^π of the actual policy π while the second phase, called *policy improvement*, uses this value function to output an improved policy π', usually using a greedy approach with respect to the current value function.

A generalization of policy iteration is *Approximate Policy Iteration* (API) algorithms which are used when the environment model is unknown or the state and action spaces are large. Exact representations of the value function Q^π and the policy π can be replaced by adjustable parameters and many approximation algorithms [24, 25] follow the scheme in Figure 2b. Next, we present a classical result in reinforcement learning that guarantees the convergence of approximate policy iteration:
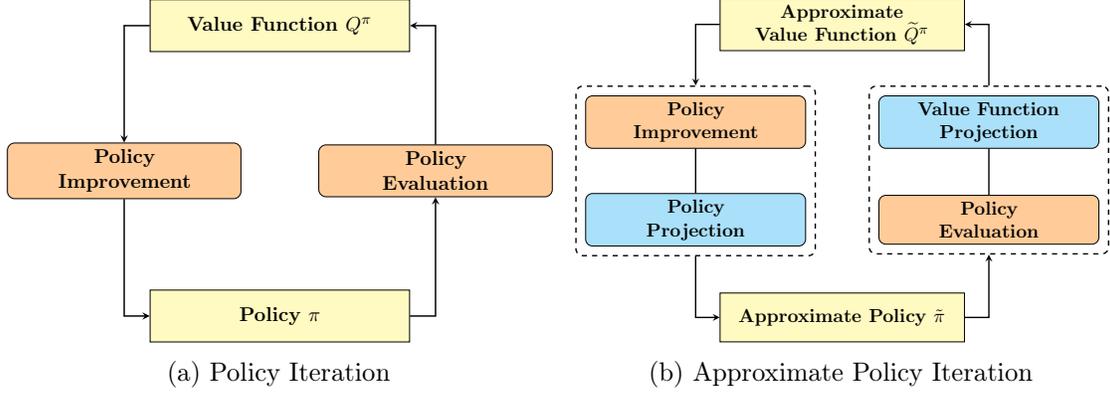
4

(a) Policy Iteration   (b) Approximate Policy Iteration

Figure 2: General schemes of PI (2a) and API (2b) as reproduced in [23].

Theorem 2.1 (Error bound of API [23, 26])**.** *Let $\{\pi_t\}_{t\in\mathbb{N}}$ be the sequence of policies generated by an approximate policy iteration with a greedy update and let $\{\widehat{Q}^{\pi_t}\}_{t\in\mathbb{N}}$ be the corresponding approximate value functions. Then, this sequence satisfies the following suboptimality bound:*

$$\limsup_{t\to+\infty} \|Q^* - Q^{\pi_t}\|_\infty \leq 2\gamma\Gamma^2 \limsup_{t\to+\infty} \|\widehat{Q}^{\pi_t} - Q^{\pi_t}\|_\infty$$

2.3 Quantum computing

Quantum computing[2] is a new paradigm for computing that uses the postulates of quantum mechanics in order to encode and compute with information. While classical systems can be only in one state at a time, namely a bit can be either in state 0 or 1, quantum systems can be in a superposition of multiple states at the same time, namely a *qubit*, which is the carrier of quantum information, can be in a superposition of the states $|0\rangle$ and $|1\rangle$, i.e. it can be written as $|x\rangle = \alpha |0\rangle + \beta |1\rangle$. The qubit $|x\rangle$ corresponds to a unit vector of the Hilbert space $\mathcal{H} = \text{span}\{|0\rangle, |1\rangle\}$ with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.

The qubit can be generalized to n-qubit states, which are unit vectors of $\mathcal{H}_n = \otimes^n \mathcal{H} \simeq \mathbb{C}^{2^n}$. Denoting by $[2^n]$ the set $\{0, \ldots, 2^n - 1\}$ and by $\{|i\rangle\}_{i\in[2^n]}$ the computational basis of $\mathcal{H}_n$, an n-qubit state can be written as $|x\rangle = \sum_{i\in[2^n]} \alpha_i |i\rangle$ with $\alpha_i = \langle x|i\rangle \in \mathbb{C}$ and $\sum_{i\in[2^n]} |\alpha_i|^2 = 1$. Quantum states evolve by applying unitary operators on them, namely applying a unitary operator U (a $2^n \times 2^n$ unitary matrix) on an n-qubit state $|x\rangle$ results in the quantum state $|Ux\rangle$. In addition, quantum states can be measured and the probability that the measurement of the state $|x\rangle$ gives outcome i is $|\alpha_i|^2$. Note that this is a probability distribution over $[2^n]$.

The quantum state corresponding to a vector $\mathbf{b} = [b_i]_{i\in[n]} \in \mathbb{R}^n$ is defined as the $\lceil \log n \rceil$-qubit state $|\mathbf{b}\rangle \in \mathbb{C}^n$ such that:

$$|\mathbf{b}\rangle = \frac{1}{\|\mathbf{b}\|} \sum_{i\in[n]} b_i |i\rangle$$

where $|i\rangle$ represents the vector e_i the i-th vector of the standard basis of $\mathbb{R}^n$. Next, we define the notion of quantum access to a matrix as used in the block-encoding framework [28].

---

[2]For a detailed introduction to quantum computing, we recommend [27].

5

Definition 2.2 (Matrix block-encoding [28]). *Let $r \in \mathbb{N}$ and $\mu \in \mathbb{R}_+$. We say that we have a μ-block-encoding $U_{\mathbf{A}}$ of a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $\|\mathbf{A}\| \leq \mu$, in total cost $T_{\mathbf{A}} = T + r$ if there exists a unitary $U_{\mathbf{A}}$ acting on $\lceil \log n \rceil + r$ qubits that can be implemented using T elementary gates such that:*

$$U_{\mathbf{A}} = \begin{pmatrix} \mathbf{A}/\mu & \cdot \\ \cdot & \cdot \end{pmatrix}$$

This framework, introduced in [28, 29], represents any sub-normalized matrix as the top-left block of a unitary $U_{\mathbf{A}}$. This definition generalizes to any matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ by building the block-encoding of its symmetrized version $\overline{\mathbf{A}} \in \mathbb{R}^{(m+n) \times (m+n)}$ such that:

$$\overline{\mathbf{A}} = \begin{pmatrix} 0 & \mathbf{A} \\ \mathbf{A}^\dagger & 0 \end{pmatrix}$$

The property of constructing a block-encoding of a matrix $\mathbf{A} = [a_{ij}]_{i,j} \in \mathbb{R}^{m \times n}$ can be reduced to being able to perform certain mappings regarding the rows of the matrix $\mathbf{A}^{(p)}$ and the columns of the matrix $\mathbf{A}^{(1-p)}$ for some $p \in [0, 1]$ where $\mathbf{A}^{(k)}$ denotes the matrix with elements $(a_{ij})^k \in \mathbb{C}$. If we define $s_q(\mathbf{A}) = \max_{i \in [m]} \|a_i\|_q^q$ to be the maximum ℓ_q-norm over the rows a_i of $\mathbf{A}$, then we have the following result:

Lemma 2.1 (Constructing block-encodings [28, 29]). *Let $p \in [0, 1]$ and $\alpha, \beta \in \mathbb{R}_+$. Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a matrix with $s_{2p}(\mathbf{A}) \leq \alpha^2$ and $s_{2(1-p)}(\mathbf{A}^\top) \leq \beta^2$. We can implement an $\alpha\beta$-block-encoding of $\mathbf{A}$ by applying a constant number of times the unitaries O^r and O^c such that:*

$$O^r : |i, 0\rangle \longrightarrow \frac{1}{\alpha} \Big[\sum_{j \in [m]} a_{ij}^p |i, j\rangle \Big] + |G_i^\perp\rangle$$

$$O^c : |0, j\rangle \longrightarrow \frac{1}{\beta} \Big[\sum_{i \in [n]} a_{ij}^{1-p} |i, j\rangle \Big] + |G_j^\perp\rangle$$

where $|G_k^\perp\rangle$ denotes some unnormalized garbage quantum state such that $\langle G_k^\perp | i, j\rangle = 0$ for all i, j.

Next, we introduce different algorithms and results for quantum linear algebra using block-encodings. The first result (Theorem 2.3) describes techniques for performing matrix arithmetics in this framework. The second algorithm is the *quantum linear system solver* whose running time depends on the quantity μ and the *condition number* of the matrix $\mathbf{A}$ defined as:

$$\kappa = \frac{\max_{\mathbf{b} \neq 0} \{\|\mathbf{A}\mathbf{b}\| / \|\mathbf{b}\|\}}{\min_{\mathbf{b} \neq 0} \{\|\mathbf{A}\mathbf{b}\| / \|\mathbf{b}\|\}}$$

The general idea of solving linear algebra problems with quantum computing is based on the singular value decomposition of matrices. This decomposition is a generalization of eigendecomposition of a positive semidefinite normal matrix and can be used to accelerate algebra and optimization procedures. We state the cost guarantees for the state-of-the-art linear algebra procedures using block-encodings [28] (Theorem 2.4). Another quantum algorithm we use is a way to recover efficiently a classical approximation to any quantum state in the ℓ_∞-norm [9] (Theorem 2.5).

Theorem 2.3 (Matrix arithmetics with block-encodings [28, 29]). *Suppose that we have a μ_i-block-encoding U_i of the matrix $\mathbf{A}_i$ at a cost T_i for all $i \in [m]$. Then we can implement with cost $\mathcal{O}(\sum_{i \in [m]} T_i)$ a $(\prod_{i \in [m]} \mu_i)$-block-encoding of $\prod_{i \in [m]} \mathbf{A}_i$ and a $(\sum_{i \in [m]} |\lambda_i| \mu_i)$-block-encoding of $\sum_{i \in [m]} \lambda_i \mathbf{A}_i$.*

Theorem 2.4 (Linear algebra with block-encodings [28]). *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a matrix such that $\|\mathbf{A}\| = 1$ and let $\epsilon > 0$ be the precision parameter. Given a μ-block-encoding of $\mathbf{A}$ with cost $T_\mathbf{A}$ and a procedure preparing a state $|\mathbf{b}\rangle$ with cost $T_\mathbf{b}$, then there exists quantum algorithms such that with probability at least $(1 - 1/\text{poly}(n))$ return a state $|\mathbf{x}\rangle$ such that $\| |\mathbf{x}\rangle - |\mathcal{A}\mathbf{b}\rangle \| \leq \epsilon$ for $\mathcal{A} \in \{\mathbf{A}, \mathbf{A}^{-1}, \mathbf{A}^\top\}$ with cost[3] $\mathcal{O}(\kappa(\mu T_\mathbf{A} + T_\mathbf{b}) \text{polylog}(\kappa/\epsilon))$.*

Theorem 2.5 (Vector tomography [9]). *Let $\mathbf{x} \in \mathbb{R}^n$ be a normalized vector. Given a procedure preparing $|\mathbf{x}\rangle$ with cost $T_\mathbf{x}$, there is a tomography algorithm that with probability at least $(1 - 1/\text{poly}(n))$ produces a unit vector $\widetilde{\mathbf{x}} \in \mathbb{R}^n$ such that $\|\mathbf{x} - \widetilde{\mathbf{x}}\|_\infty \leq \epsilon$ with cost $\mathcal{O}(T_\mathbf{x} \log(n)/\epsilon^2)$.*

Moreover, we also introduce this claim from [10] that bounds the distance between two quantum states in terms of the distance between the corresponding unnormalized vectors:

Claim 2.6. *Let θ be the angle between two vectors $\mathbf{x}$ and $\mathbf{y}$ and assume that $\theta < \pi/2$. Then $\|\mathbf{x} - \mathbf{y}\| \leq \epsilon$ implies $\| |\mathbf{x}\rangle - |\mathbf{y}\rangle \| \leq \sqrt{2}\epsilon/\|\mathbf{x}\|$.*

3 Quantum policy iteration

3.1 General framework for quantum policy iteration

We start by providing a general framework for quantum reinforcement learning by appropriately extending the policy iteration scheme to the quantum case. In this section we look at the case where we work directly with exact representations of the value function Q^π, while in Section 4, we generalize the algorithm to the approximate case. Similarly to the classical case, we want to build quantum methods that generate a sequence of policies improving at each iteration and converging to an approximation of the optimal policy π^*. We refer to our framework as *Quantum Policy Iteration* (QPI) and we summarize the general procedure in Figure 3.

We define the *quantum policy evaluation* step as a quantum procedure (one can think of this as a unitary operation or a quantum circuit) that takes as input a classical policy π and performs a mapping to create a quantum state that approximates or more generally contains some information about the classical value function Q^π. Again here one can define different quantum outputs that contain information about the value function and we will provide examples in the remaining of the paper. Similarly, we define the *quantum policy improvement* step, as a quantum procedure (one may want to think of this as a generalized measurement operation), that takes as input the output quantum states from the quantum policy evaluation procedure, and extracts classical information by performing measurements on them, in order to compute a new policy π' based on some policy update rule.

---

[3]If $\|\mathbf{A}\| \neq 1$, then we rescale the factor μ to $\mu/\|\mathbf{A}\|$.
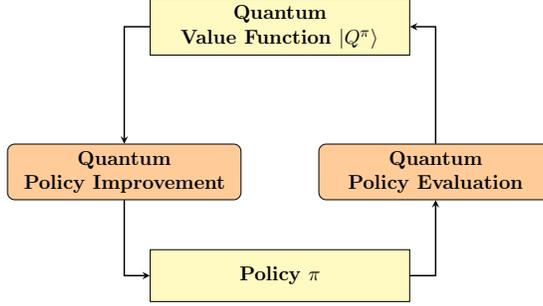
Figure 3: Quantum Policy Iteration

3.2 Quantum policy evaluation

We will instantiate here our quantum policy iteration framework by providing one specific example of a quantum policy iteration method. We start by defining for any policy π the quantum state:

$$|Q^\pi\rangle = \frac{1}{\|Q^\pi\|} \sum_{sa} Q^\pi(s,a) |s,a\rangle$$

which encodes the classical value function Q^π in the amplitudes of a normalized quantum state, and we call this state the quantum value function state. As we can see, this state contains information about the value function Q^π, though one needs to be careful since a copy of this state cannot recreate the complete function Q^π. It can still provide useful information, in particular we can use this state to sample a pair (s,a) with probability proportional to $(Q^\pi(s,a))^2$. Moreover, there exist efficient quantum procedures for producing approximations to this state that we describe below.

We assume we have quantum access to the MDP parameters P and R so that we can construct, for any policy π, a block-encoding of the policy transition matrix defined as:

$$P^\pi := \big[p(s,a,s')\pi(s',a')\big]_{sa,s'a'}$$

In Subsection 3.4, we discuss how one can get quantum access to the parameters of any MDP $\mathcal{M}$ and policy π assuming that the transition matrix P and the reward function R can be efficiently computed, which is the case for many classes of environments. Denoting $\mathbf{A}^\pi = I - \gamma P^\pi$ and $\mathbf{b} = R$, it follows from the Bellman equation that the value function Q^π is the solution of the linear system $\mathbf{A}^\pi Q^\pi = \mathbf{b}$. Using the *quantum linear system solver* from Theorem 2.4, we can build an ϵ-approximation (in ℓ_2-norm) state $|\widehat{Q}^\pi\rangle$ to the quantum value function state $|Q^\pi\rangle$:

Theorem 3.1 (Quantum policy evaluation). *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ be a finite Markov decision process, π a policy and $\epsilon > 0$ the precision parameter. Suppose there exists a μ_{P^π}-block-encoding of the policy transition matrix P^π that can be implemented with cost T_{P^π}. Also suppose we can prepare the reward vector $|R\rangle$ with cost T_R. Then there exists a quantum algorithm that with probability at least $(1 - 1/\operatorname{poly}(SA))$ returns a quantum state $|\widehat{Q}^\pi\rangle$ such that $\| |\widehat{Q}^\pi\rangle - |Q^\pi\rangle \| \le \epsilon$ with cost:*

$$\mathcal{O}((\mu_{P^\pi}\mathrm{T}_{P^\pi} + \mathrm{T}_R)\Gamma \operatorname{polylog}(\Gamma/\epsilon))$$

Proof. Using Theorem 2.3, we can implement with cost $T_{\mathbf{A}} = \mathcal{O}(T_{P^\pi})$ a $(1 + \gamma\mu_{P^\pi})$-block-encoding of $\mathbf{A}^\pi = I - \gamma P^\pi$ as a linear combination of the trivial 1-block-encoding of I and the μ_{P^π}-block-encoding of P^π. Then, we apply the quantum linear system solver from Theorem 2.4 with the procedure that generates the state $|\mathbf{b}\rangle = |R\rangle$ to return a quantum state $|\widehat{Q}^\pi\rangle$ ϵ-close to $|Q^\pi\rangle = |(\mathbf{A}^\pi)^{-1}\mathbf{b}\rangle$ with cost:

$$\mathcal{O}\big(\kappa(\mathbf{A}^\pi)\big(\|\mathbf{A}^\pi\|^{-1}(1 + \gamma\mu_{P^\pi})T_{P^\pi} + T_R\big)\operatorname{polylog}(\kappa(\mathbf{A}^\pi)/\epsilon)\big)$$

Since P^π is a row-stochastic matrix, we have $\|P^\pi\| = 1$ which implies that the singular values $\sigma_{\mathbf{A}}$ of $\mathbf{A}^\pi$ range in $[1-\gamma, 1+\gamma]$. We finish the proof by plugging the following upper bounds in the total cost: $1 + \gamma\mu_{P^\pi} = \mathcal{O}(\mu_{P^\pi})$, $\kappa(\mathbf{A}^\pi) = \frac{\max\sigma_{\mathbf{A}}}{\min\sigma_{\mathbf{A}}} \leq \frac{1+\gamma}{1-\gamma} = \mathcal{O}(\Gamma)$ and $\|\mathbf{A}^\pi\|^{-1}\kappa(\mathbf{A}^\pi) = \frac{1}{\min\sigma_{\mathbf{A}}} \leq \frac{1}{1-\gamma} = \Gamma$. $\quad\square$

We can now go ahead and define our specific quantum policy evaluation method as the one that given a policy π, uses a quantum linear system solver to create a quantum state which is an approximation of the state $|Q^\pi\rangle$.

3.3 Quantum policy improvement

We will now describe a quantum policy improvement method that works together with the specific quantum policy evaluation method we described above, where approximations $|\widehat{Q}^\pi\rangle$ to $|Q^\pi\rangle$ are produced. Let us assume that these states can be produced in time T_{Q^π}.

The quantum policy improvement method consists of first performing a number of M measurements of the state $|\widehat{Q}^\pi\rangle$ for a total cost of $\mathcal{O}(M \times T_{Q^\pi})$. Denote by $M(s,a)$ the number of times outcome (s,a) is observed. Then, we define the following strategy for the policy update:

$$\pi'(s) = \arg\max_a M(s,a) \approx \arg\max_a Q^\pi(s,a)$$

which takes time $\mathcal{O}(SA)$.

Let us analyze this policy improvement method. A measurement of the state $|\widehat{Q}^\pi\rangle$ outputs some state-action pair (s,a) with probability $|\langle\widehat{Q}^\pi|s,a\rangle|^2$ and we have:

$$\lim_{M\to+\infty} \frac{M(s,a)}{M} = |\langle\widehat{Q}^\pi|s,a\rangle|^2$$

We make now some remarks on the appropriate value of M. Note first that the approximation state $|\widehat{Q}^\pi\rangle$ is ϵ-close in ℓ_2-norm to the quantum value function state $|Q^\pi\rangle$ and the cost of creating these states depends only logarithmically on the parameter ϵ. Thus, we can take this parameter very small, and so if the number of measurements M guarantees that we can closely reconstruct the state $|\widehat{Q}^\pi\rangle$, then this guarantee will carry over to the state $|Q^\pi\rangle$. Thus, if we want to guarantee an ϵ-approximation of the quantum value function state $|Q^\pi\rangle$ in ℓ_2-norm, one would need $M = \widetilde{\mathcal{O}}(SA/\epsilon^2)$, and for ℓ_∞-norm, which is what is used in reinforcement learning, $M = \widetilde{\mathcal{O}}(1/\epsilon^2)$. In this case, we are able to reconstruct an ϵ-approximation in ℓ_∞-norm of the normalized quantum value function.

In practice, setting M to be $\widetilde{\mathcal{O}}(1/\epsilon^2)$ may be more than what is needed, since our goal is not to recreate Q^π but to find $\pi'(s) = \arg\max_a Q^\pi(s,a)$. This number M can be adjusted in practice

---

Algorithm 1 Quantum Policy Iteration

---

    input MDP $\mathcal{M}$, number of measurements M, number of iterations T, precision ϵ.
    initialize policy π_0.
    for $t = 0$ **to** $T - 1$ **do**
        initialize measurement histogram $M_t(s, a) = 0$ for every pair (s, a).
        for $m = 0$ **to** $M - 1$ **do**
            use the quantum linear solver with precision ϵ to obtain $|\widehat{Q}^{\pi_t}\rangle \approx |(I - \gamma P^{\pi_t})^{-1}R\rangle$.
            measure $|\widehat{Q}^{\pi_t}\rangle$ to get pair (s, a) with probability $|\langle \widehat{Q}^{\pi_t}|s, a\rangle|^2$.
            update measurement histogram $M_t(s, a) = M_t(s, a) + 1$.
        end for
        improve policy as $\pi_{t+1}(s) = \arg\max_a M_t(s, a)$ for every s.
    end for
    output policy π_T

---

until the given method provides good results and in fact, in our experiments it was tuned to be significantly smaller than the theoretical value $\widetilde{\mathcal{O}}(1/\epsilon^2)$.

It would be interesting to understand theoretically the number of samples needed for a successful implementation of a quantum policy improvement scheme, though we believe that in the end this would be use case-specific. Last, note that for the policy update rule we do not estimate directly the value function Q^π but its normalized version q^π, i.e. we do directly the measurement outcomes $M(s, a)$, since the norm does not change the $\arg\max$ calculation.

To sum up, we have defined a quantum policy improvement method as the one that given access to a quantum procedure that outputs a quantum value function state $|Q^\pi\rangle$, performs a number M of measurements in order to create a measurement histogram from which the policy is updated via an $\arg\max$ computation.

The quantum policy iteration method we presented appears in Algorithm 1. In the next subsections we prove convergence, how to construct the necessary block-encodings for the Markov decision process, and analyze its running time.

3.4   Constructing block-encodings

We are now going to show how to construct the block-encoding of P^π and the unitary that prepares the quantum state associated to R that we need in order to perform quantum policy iteration. In the classical case, one needs to have access to the transition matrix P, the policy π and the reward vector R in order to compute the policy-transition matrix P^π and the corresponding value function $Q^\pi = (I - \gamma P^\pi)^{-1}R = (\mathbf{A}^\pi)^{-1}\mathbf{b}$. In this subsection, we will discuss the access we need in the quantum case. More precisely, we will assume quantum access to the parameters $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ of the MDP $\mathcal{M}$ and to the policy π which will be used to build the block-encoding of $\mathbf{A}^\pi$ and the state $|\mathbf{b}\rangle$ as in Theorem 3.1.

Next, we specify formally what we mean by quantum access to the MDP $\mathcal{M}$:

Definition 3.2 (Quantum access to $\mathcal{M}$). *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ be a finite MDP and let $c_P \in \mathbb{R}_+$ such that $c_P \geq s_1(P^\top) = \max_{s'} \sum_{sa} p(s, a, s')$. We say that we have quantum access to $\mathcal{M}$ with costs (T_P, T_R) if we can implement:*

1. *An oracle for the rows of the transition matrix with cost T_P such that*

$$O_P^r : |s, a\rangle \, |0_s\rangle \longrightarrow \sum_{s'} \sqrt{p(s, a, s')} \, |s, a\rangle \, |s'\rangle$$

2. *An oracle for the columns of the transition matrix with cost T_P such that:*

$$O_P^c : |0_s, 0_a\rangle \, |s'\rangle \longrightarrow \frac{1}{\sqrt{c_P}} \Big[\sum_{sa} \sqrt{p(s, a, s')} \, |s, a\rangle \, |s'\rangle \Big] + |G_{s'}^\perp\rangle \, |s'\rangle$$

   where $\{|G_{s'}^\perp\rangle\}_{s' \in \mathcal{S}}$ are unnormalized garbage quantum states such that $\langle G_{s'}^\perp | s, a \rangle = 0$ for all transitions $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$.

3. *An oracle for the reward vector with cost T_R such that:*

$$O_R : |0_s, 0_a\rangle \longrightarrow |R\rangle = \frac{1}{\|R\|} \sum_{sa} r(s, a) \, |s, a\rangle$$

Let us look at the above oracles in more detail. The oracle O_R gives quantum access to the reward vector $|R\rangle$ whilst the oracles O_P^r and O_P^c correspond to the unitaries used to construct block-encodings from Lemma 2.1 applied on the transition matrix P where we have set the factor p to be $1/2$ in our case. This choice is based on the observation that the rows of P form probability distributions over $\mathcal{S}$ and can be seen as valid quantum states. The unitary O_P^r encodes the rows of P and maps every state-action pair $|s, a\rangle$ to the quantum state $\sum_{s'} \sqrt{p(s, a, s')} \, |s'\rangle$ with amplitudes forming a probability distribution, over next states $|s'\rangle$, that match the dynamics of the MDP $\mathcal{M}$. On the other hand, O_P^c encodes the columns of P and maps every next-state $|s'\rangle$ to the neighboring state-action pairs $|s, a\rangle$ such that $p(s, a, s') \neq 0$. The quantity c_P depends on the dynamics of $\mathcal{M}$ and is an upper-bound on the portion of the state-action space $\mathcal{S} \times \mathcal{A}$ covered by every s'.

The definitions of the above oracles are made so that Lemma 2.1 with parameters $\alpha = 1$ and $\beta = \sqrt{c_P}$ implies that we can implement a $\sqrt{c_P}$-block-encoding of P using the oracles O_P^r and O_P^c and with cost T_P.

We have shown that efficient quantum access to $\mathcal{M}$ as defined in Definition 3.2 suffices to implement efficiently a block-encoding for the transition matrix P. In the next definition, we introduce quantum access to the policy π where we map each state to a distribution over the set of actions.

Definition 3.3 (Quantum access to π). *Let π be a finite policy. We say that we have quantum access to π with cost T_π if we can implement the following oracle with cost T_π:*

$$O_\pi : |s, 0_a\rangle \longrightarrow |s, \pi(s)\rangle = \sum_a \sqrt{\pi(s, a)} \, |s, a\rangle$$

Recall that we need a block-encoding for the matrix P^π to perform quantum policy evaluation as in Theorem 3.1. We will build its block-encoding by combining the block-encoding of P with the oracle O_π as stated in the following lemma:

Lemma 3.1 (Block-encoding of P^π). *Given quantum access to an MDP $\mathcal{M}$ with cost (T_P, T_R) and to a policy π with cost T_π, we can implement a μ_{P^π}-block-encoding of P^π with cost $\mathcal{O}(T_P + T_\pi)$, where the factor $\mu_{P^\pi} = \sqrt{c_P}$ does not depend on the policy π.*

Proof. We will use O_π to construct a 1-block-encoding of the matrix $\Pi \in \mathbb{R}^{SA \times S}$ defined as:

$$\Pi := \left[\mathbb{1}[s = s'] \pi(s, a) \right]_{sa, s'}$$

and then use Lemma 2.1 to get a block-encoding of P^π which can be rewritten as $P^\pi = P\Pi^\top$. First, we build the oracle acting on the three registers $|0_s\rangle |0_a\rangle |s'\rangle$ that gives access to the columns of Π. Let U_C be the unitary that uses CNOT gates to copy the third register to the first register. If we apply U_C followed by O_π on the first register, we get:

$$\sum_a \sqrt{\pi(s', a)} |s', a\rangle |s'\rangle = \sum_{sa} \sqrt{\mathbb{1}[s = s'] \pi(s, a)} |s, a\rangle |s'\rangle$$

Next, we build the oracles acting on four registers $|s\rangle |a\rangle |0_s\rangle |0_a\rangle$ that gives quantum access to the rows of Π. Let U_C be the unitary that copies the first state register to the third register. If we apply U_C followed by O_π on the last two registers, we get:

$$\sum_{a'} \sqrt{\pi(s, a')} |s, a\rangle |s\rangle |a'\rangle$$

Using again U_C to copy the second register to the fourth register we get

$$\sqrt{\pi(s, a)} |s, a\rangle |s\rangle |0\rangle + |G_{sa}^\perp\rangle$$

where $|G_{sa}^\perp\rangle$ is a garbage quantum state such that $\langle G_{sa}^\perp | s, a, s, 0\rangle = 0$. The procedures above for accessing the rows and columns of Π give us a 1-block-encoding of Π. Finally, we apply the product of the block-encodings of Π and P to get a μ_{P^π}-block-encoding of $P\Pi^\top = P^\pi$. $\qquad\square$

3.5 Running time analysis

From Theorem 3.1, we see that the cost of the quantum policy evaluation step is:

$$\mathcal{O}((\mu_{P^\pi}(T_P + T_\pi) + T_R)\Gamma \operatorname{polylog}(\Gamma/\epsilon))$$

Let us make some comments now of how this cost can behave in practice. We will see in following sections that for many MDP of interest quantum access can be implemented using quantum circuits of $\widetilde{\mathcal{O}}(SA)$ qubits and with only $\operatorname{polylog}(SA)$ depth, and thus the running time for the quantum policy evaluation, where here time refers to the depth of the quantum circuit will be of the form:

$$\mathcal{O}(\mu_{P^\pi}\Gamma \operatorname{polylog}(SA\Gamma/\epsilon))$$

The overall running time of the quantum policy iteration will then be $\widetilde{\mathcal{O}}(SA + \mu_{P^\pi} M\Gamma)$, where M is the number of measurements during the quantum policy improvement step. If we are using the ℓ_∞-tomography algorithm from Theorem 2.5 then M is taken to be $\widetilde{\mathcal{O}}(1/\epsilon^2)$ and the running time becomes $\widetilde{\mathcal{O}}(SA + \mu_{P^\pi}\Gamma/\epsilon^2)$, while in the experiments the value was actually smaller.

In comparison, the running time of classical policy iteration is $\mathcal{O}((SA)^\omega)$ when using a classical linear system solver[4]. Whether our quantum algorithm provides an advantage for a specific environment depends on the environment parameters, i.e. the values of Γ and μ_{P^π}, what is the actual cost of constructing the block encodings of the transition matrix P and the policy π, as well as how many samples M are needed for a good policy improvement method.

Let us also remark on the value μ_{P^π}. This value can be shown in the worst case to be $\sqrt{SA}$ but we expect it to be much smaller when we have efficient access to the transition matrix P as detailed in Subsection 3.4 where we have shown that $\mu_{P^\pi} = \sqrt{c_P}$. The idea of the bound c_P is that in some of the most studied environments in reinforcement learning, any next state s' arises as a result of taking an action a from a small number of neighboring states s. In other terms, the transition matrix is very sparse and usually has $\mathcal{O}(A)$ non-zero elements in each column. We use our approach to build a $\sqrt{c_P}$-block-encoding of P^π for a total running of $\widetilde{\mathcal{O}}(\sqrt{c_P} \times \Gamma/\epsilon^2)$. We expect this bound $\sqrt{c_P}$ to be very small and have poly-logarithmic dependence on the size of the state space $\mathcal{S}$. For example, in the case of d-dimensional mazes, the factor c_P can be chosen such that $c_P = 2d = \mathcal{O}(\log S)$. For two-player board games, such as chess or go, again the number of different states that could result to a particular state of the board through a single action are small and we can again think of it as $c_P = \mathcal{O}(A)$, much smaller than the number of states that grows exponentially with the size of the board game. A concrete example is given in Subsection 5.1 where we show, for FROZENLAKE, that $c_P = \mathcal{O}(1)$ is constant and does not depend on the environment size.

3.6 Convergence guarantees

We are going now to prove the theoretical convergence of our algorithm with precision ϵ and $M = \widetilde{\mathcal{O}}(1/\epsilon^2)$ measurements using a similar approach to [23]. Our bound is similar to the result given in Theorem 2.1 but using the norm $\|.\|_\rho \leq \|.\|_\infty$ which is the ℓ_2-norm weighted by the uniform distribution ρ over $\mathcal{S} \times \mathcal{A}$. The $\|.\|_\rho$ norm is equal to the expected norm of a coordinate, instead of the maximum one as in the ℓ_∞ norm. Weighted quadratic norms are also used in classical reinforcement learning as in [30] to prove the convergence of approximation algorithms. Next, we state the error bound on the policies generated by our algorithm when performing $M = \widetilde{\mathcal{O}}(1/\epsilon^2)$ measurements followed by a greedy update on the reconstructed normalized value function $\widehat{q}^{\pi_t}$ as described in Algorithm 1.

Theorem 3.4 (Error bound of QPI). *Let $\{\pi_t\}_{t\in\mathbb{N}}$ be the sequence of policies generated by the quantum policy iteration algorithm with a greedy update and let $\{\widehat{q}^{\pi_t}\}_{t\in\mathbb{N}}$ be the corresponding approximate normalized value functions. Then, this sequence satisfies the following suboptimality bound:*

$$\limsup_{t\to+\infty} \| |Q^*\rangle - |Q^{\pi_t}\rangle \|_\rho \leq 2\sqrt{2}\gamma\Gamma^2 \limsup_{t\to+\infty} \|\widehat{q}^{\pi_t} - q^{\pi_t}\|_\infty$$

---

[4]ω is the matrix multiplication exponent, with best known theoretical value 2.37 and in practice close to 3.

Proof. First, we use Theorem 2.1 on our quantum policy iteration algorithm which can be seen as a classical approximate policy iteration algorithm with a greedy update applied on $\widetilde{Q}^{\pi_t} = \|Q^{\pi_t}\|.\widehat{q}^{\pi_t}$ where $\widehat{q}^{\pi_t}(s,a) := \sqrt{M(s,a)/M}$. Moreover, note that the update in Algorithm 1 is equivalent to $\pi_{t+1}(s) = \arg\max \widehat{q}^{\pi_t}(s,a)$. In this case, the approximation errors $\|\widetilde{Q}^{\pi_t} - Q^{\pi_t}\|_\infty$ are bounded by $\|Q^*\|.\|\widehat{q}^{\pi_t} - q^{\pi_t}\|_\infty$ since $\|Q^{\pi_t}\| \le \|Q^*\|$ for every policy π_t. Using Theorem 2.1, we have:

$$\limsup_{t\to+\infty} \|Q^* - Q^{\pi_t}\|_\infty \le 2\gamma\Gamma^2 \limsup_{t\to+\infty} \|\widetilde{Q}^{\pi_t} - Q^{\pi_t}\|_\infty \le 2\gamma\Gamma^2 \|Q^*\| \limsup_{t\to+\infty} \|\widehat{q}^{\pi} - q^{\pi}\|_\infty$$

Since we define our rewards to be greater than 0, the angle between the vectors Q^{π_t} and Q^* will be no greater than $\pi/2$. Using Claim 2.6, we have:

$$\| \,|Q^*\rangle - |Q^{\pi_t}\rangle \,\| \le \frac{\sqrt{2}}{\|Q^*\|} \|Q^* - Q^{\pi_t}\|$$

By taking the limit superior in the inequality above and observing that $\|.\|_\rho = \|.\|/\sqrt{SA}$, we conclude the proof by:

$$\limsup_{t\to+\infty} \| \,|Q^*\rangle - |Q^{\pi_t}\rangle \,\|_\rho \le \frac{\sqrt{2}}{\|Q^*\|} \limsup_{t\to+\infty} \|Q^* - Q^{\pi_t}\|_\rho$$
$$\le \frac{\sqrt{2}}{\|Q^*\|} \limsup_{t\to+\infty} \|Q^* - Q^{\pi_t}\|_\infty$$
$$\le 2\sqrt{2}\gamma\Gamma^2 \limsup_{t\to+\infty} \|\widehat{q}^{\pi_t} - q^{\pi_t}\|_\infty$$

$\square$

Using weighted quadratic norms instead of the ℓ_∞-norm appears in many approximate algorithms in reinforcement learning when the subroutines minimize the ℓ_2-norm [30]. In our case, the bound shows that our quantum approach is a stable algorithm. When the inequality $\|\widehat{q}^{\pi_t} - q^{\pi_t}\|_\infty \le \epsilon$ holds for every iteration, the bound in Theorem 3.4 becomes:

$$\limsup_{t\to+\infty} \| \,|Q^*\rangle - |Q^{\pi_t}\rangle \,\|_\rho \le 2\sqrt{2}\gamma\Gamma^2\epsilon$$

which shows that quantum policy iteration oscillates between sub-optimal policies with value functions ϵ-close to the optimal policy.

4 Quantum approximate policy iteration

4.1 General framework for quantum approximate policy iteration

We continue the description of our general framework for quantum reinforcement learning by looking at the common case where we may not be able to compute directly the value function Q^π, for example when the dynamics of the environment are unknown or the state-action space is too large. Instead, one approximates it with linear or non-linear value functions. Linear value functions correspond to the use of a linear combinations of features whilst the non-linear case corresponds to the use of non-linear approximation schemes such as neural networks for example.
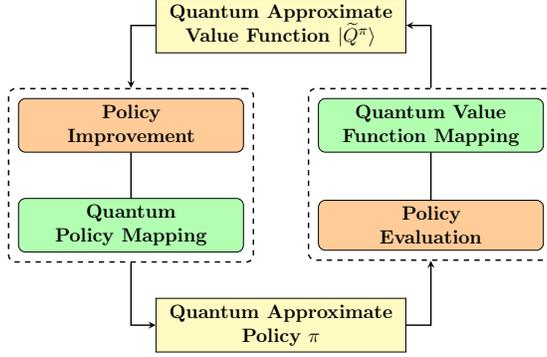
14

Figure 4: Quantum Approximate Policy Iteration

In our case, we are going to provide quantum algorithms for approximate policy iteration with linear value function approximation. We will start with the model-based case when having access to the parameters of the MDP $\mathcal{M}$ and provide a model-free implementation in Subsection 4.6 for the case when the dynamics of $\mathcal{M}$ are unknown. We refer to our framework as quantum approximate policy iteration (QAPI) and we summarize the general procedure in Figure 4.

4.2 Quantum approximate policy evaluation

In many cases, linear architectures are used for value function approximation where the Q^π values are approximated by a linear combination $\widetilde{Q}^\pi$ of K basis functions of the form $\phi_k : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and the policy parameters $w^\pi \in \mathbb{R}^K$:

$$Q^\pi(s,a) \approx \widetilde{Q}^\pi(s,a) = \sum_k \phi_k(s,a) w_k^\pi$$

Note that the linearly independent features $\Phi := [\phi_k(s,a)]_{sa,k} \in \mathbb{R}^{SA \times K}$ are usually hand-crafted and common between all policies. On the other hand, one should compute $w^\pi \in \mathbb{R}^K$ to get the estimated value function $\widetilde{Q}^\pi = \Phi w^\pi$. To do so, we will base our approach on the least-squares policy iteration algorithm by Lagoudakis and Parr [23] for finding the parameters w^π. In the model-based case when we have access to P and R, we can compute P^π and we retrieve w^π as the solution of the linear system $\mathbf{A}^\pi w^\pi = \mathbf{b}$ with $\mathbf{A}^\pi = \Phi^\top (\Phi - \gamma P^\pi \Phi)$ and $\mathbf{b} = \Phi^\top R$.

We now define for any policy π the corresponding quantum state:

$$|w^\pi\rangle = \frac{1}{\|w^\pi\|} \sum_k w_k^\pi |k\rangle$$

which encodes the classical weight vector w^π in the amplitudes of a normalized quantum state.

Using the same arguments provided in Section 4, a copy of this quantum state cannot recreate w^π. However, it can still be used to provide useful information for quantum policy improvement as we will show in the next subsection. Using quantum linear algebra techniques, there exist efficient quantum procedures for producing approximations to $|w^\pi\rangle$ that we describe below.

First, we assume we have quantum access to the model parameters P and R as in the quantum policy iteration algorithm. We also assume that we have quantum access to the features matrix Φ and we discuss how to do so in Subsection 4.4. Denoting $\mathbf{A}^\pi = \Phi^\top(\Phi - \gamma P^\pi \Phi)$ and $\mathbf{b} = \Phi^\top R$, the weight vector w^π is the solution to the linear system $\mathbf{A}^\pi w^\pi = \mathbf{b}$. Using the quantum linear system solver from Theorem 2.4, we can build an ϵ-approximation (in ℓ_2-norm) state $|\widehat{w}^\pi\rangle$ to the quantum weight vector $|w^\pi\rangle$:

Theorem 4.1 (Model-based policy evaluation). *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ a finite Markov decision process, Φ a features matrix with $\|\Phi\| = 1$, π a policy and $\epsilon > 0$ the precision parameter. Suppose there exists a μ_{P^π}-block-encoding of the policy transition matrix P^π with cost T_{P^π} and a μ_Φ-block-encoding of the features matrix Φ with cost T_Φ. Also suppose that we can prepare the reward vector $|R\rangle$ with cost T_R. Then there exists a quantum algorithm that returns a quantum state $|\widehat{w}^\pi\rangle$ such that $\|\,|\widehat{w}^\pi\rangle - |w^\pi\rangle\,\| \le \epsilon$ with cost:*

$$\mathcal{O}\big(\kappa_\Phi^2\big((\mu_\Phi^2 \mu_{P^\pi} + \mu_\Phi \kappa_\Phi)\mathrm{T}_\Phi + \mu_\Phi^2 \mu_{P^\pi}\mathrm{T}_{P^\pi} + \kappa_\Phi \mathrm{T}_R\big)\Gamma\,\mathrm{polylog}(\kappa_\Phi\Gamma/\epsilon)\big)$$

Proof. As we said, denoting $\mathbf{A}^\pi = \Phi^\top(\Phi - \gamma P^\pi \Phi)$ and $\mathbf{b} = \Phi^\top R$, the weight vector w^π is the solution to the linear system $\mathbf{A}^\pi w^\pi = \mathbf{b}$. First, we implement the procedure that prepares $|\mathbf{b}\rangle = |\Phi^\top R\rangle$. Using the μ_Φ-block-encoding of Φ, and picking $\epsilon' = (1 + \gamma)\epsilon/(2\sqrt{2}\kappa_\Phi^2 \Gamma)$, we can efficiently generate a state $|\widehat{\mathbf{b}}\rangle$ such that $\|\,|\widehat{\mathbf{b}}\rangle - |\mathbf{b}\rangle\,\| \le \epsilon'$ with cost:

$$\mathrm{T}_\mathbf{b} = \mathcal{O}(\kappa_\Phi(\mu_\Phi \mathrm{T}_\Phi + \mathrm{T}_R)\,\mathrm{polylog}(\kappa_\Phi\Gamma/\epsilon))$$

Next, we build the block encoding of $\mathbf{A}^\pi = \Phi^\top(I - \gamma P^\pi)\Phi$ using the block-encodings of P^π, Φ and $\Phi^\top$. Using the same approach as in Theorem 3.1, we can build a $(1 + \gamma\mu_{P^\pi})$-block-encoding of $I - \gamma P^\pi$ with cost $\mathcal{O}(\mathrm{T}_{P^\pi})$. Using Theorem 2.3 to compute the block-encoding of the product $\Phi^\top(I - \gamma P^\pi)\Phi$, we get a $\mu_\mathbf{A}$-block-encoding of $\mathbf{A}^\pi$ with cost $\mathrm{T}_\mathbf{A} = \mathcal{O}(\mathrm{T}_\Phi + \mathrm{T}_{P^\pi})$ such that $\mu_\mathbf{A} = \mu_\Phi^2(1 + \gamma\mu_{P^\pi}) = \mathcal{O}(\mu_\Phi^2 \mu_{P^\pi})$.

Then, we apply the quantum linear solver with precision $\epsilon/2$ to generate a state $|\widehat{w}^\pi\rangle$ such that $\|\,|\widehat{w}^\pi\rangle - |(\mathbf{A}^\pi)^{-1}\widehat{\mathbf{b}}\rangle\,\| \le \epsilon/2$ with cost:

$$\mathcal{O}\big(\kappa(\mathbf{A}^\pi)\big(\|\mathbf{A}^\pi\|^{-1}\mu_\mathbf{A}\mathrm{T}_\mathbf{A} + \mathrm{T}_\mathbf{b}\big)\,\mathrm{polylog}(\kappa(\mathbf{A}^\pi)/\epsilon)\big)$$

Since P^π is a row-stochastic matrix, we know that $\|P^\pi\| = 1$ and that the singular values of $I - \gamma P^\pi$ range in $[1 - \gamma, 1 + \gamma]$. Similarly, we have $\|\Phi\| = 1$ and the singular-values of Φ range in $[1/\kappa_\Phi, 1]$. We conclude that $\mathbf{A}^\pi$ has its singular values in $[(1 - \gamma)/\kappa_\Phi^2, 1 + \gamma]$. It follows that $\kappa(\mathbf{A}^\pi) = \mathcal{O}(\kappa_\Phi^2 \Gamma)$ and $\|\mathbf{A}^\pi\|^{-1}\kappa(\mathbf{A}^\pi) = \mathcal{O}(\kappa_\Phi^2 \Gamma)$. We plug these bounds into the total cost to get the total cost of our algorithm:

$$\mathcal{O}\big(\kappa_\Phi^2\big(\mu_\Phi^2 \mu_{P^\pi}(\mathrm{T}_\Phi + \mathrm{T}_{P^\pi}) + \kappa_\Phi(\mu_\Phi \mathrm{T}_\Phi + \mathrm{T}_R)\big)\Gamma\,\mathrm{polylog}(\kappa_\Phi\Gamma/\epsilon)\big)$$

We conclude the proof by showing that $|\widehat{w}^\pi\rangle$ is ϵ-close to $|w^\pi\rangle = |(\mathbf{A}^\pi)^{-1}\mathbf{b}\rangle$. We use Claim 2.6 to show that, for a small value of ϵ, $\|\,|(\mathbf{A}^\pi)^{-1}\widehat{\mathbf{b}}\rangle - |(\mathbf{A}^\pi)^{-1}\mathbf{b}\rangle\,\| \le \sqrt{2}\kappa(\mathbf{A}^\pi)\epsilon'$. Then, we have:

$$\|\,|\widehat{w}^\pi\rangle - |w^\pi\rangle\,\| \le \|\,|\widehat{w}^\pi\rangle - |(\mathbf{A}^\pi)^{-1}\widehat{\mathbf{b}}\rangle\,\| + \|\,|(\mathbf{A}^\pi)^{-1}\widehat{\mathbf{b}}\rangle - |(\mathbf{A}^\pi)^{-1}\mathbf{b}\rangle\,\|$$
$$\le \epsilon/2 + \epsilon'\kappa_\Phi^2\Gamma = \epsilon$$

$\square$

Algorithm 2 Quantum Approximate Policy Iteration

---

input MDP $\mathcal{M}$, features Φ, number of measurements M, number of iterations T, precision ϵ.
initialize policy π_0.
for $t = 1$ **to** T **do**
  for $s \in \mathcal{S}$ **do**
    initialize measurement histogram. $M_t(a) = 0$ for every action a.
    for $m = 0$ **to** $M - 1$ **do**
      use quantum linear solver with precision ϵ to obtain $|\widehat{w}^{\pi_t}\rangle \approx |(\Phi^\intercal \Phi - \gamma \Phi^\intercal P^{\pi_t} \Phi)^{-1} \Phi^\intercal R\rangle$.
      use quantum linear algebra with precision ϵ to obtain $|\widehat{Q}^{\pi_t}(s, \cdot)\rangle \approx |\Phi(s)\widehat{w}^{\pi_t}\rangle$.
      measure $|\widehat{Q}^{\pi_t}(s, \cdot)\rangle$ to get action a with probability $|\langle a|\widehat{Q}^{\pi_t}(s, .)\rangle|^2$.
      update measurement histogram $M_t(a) = M_t(a) + 1$
    end for
    improve policy as $\pi_{t+1}(s) = \arg\max M_t(a)$.
  end for
end for
output policy π_T

4.3   Quantum approximate policy improvement

We will now describe several quantum policy improvements methods that work together with the approximate quantum policy evaluation method we described above, where for each policy π we estimate a weight vector $|\widehat{w}^\pi\rangle$. Let us assume that these states can be produced in time T_{w^π}. Again, we can assume a very small ϵ in the approximation guarantee of the states $|\widehat{w}^\pi\rangle$ and the states $|w^\pi\rangle$ (since it appears only inside a logarithm in the running time) and thus the approximation to the state $|\widehat{w}^\pi\rangle$ we will achieve through measurements will provide the same guarantees for the state $|w^\pi\rangle$ as well.

Our goal is to be able to compute a greedy policy with respect to the approximate value function $\widehat{Q}^\pi = \Phi\widehat{w}^\pi$. Since our quantum procedure produces the normalized state $|\widehat{w}^\pi\rangle$, we are going to perform measurements in order to compute the actions corresponding to the improved policy π' defined as $\pi'(s) = \arg\max_a \Phi(s)\widehat{w}^\pi$ where $\Phi(s) \in \mathbb{R}^{A \times K}$ is the matrix with rows $\Phi(s, a)^\top$ containing the features associated to the state s such that $\Phi(s)\widehat{w}^\pi$ is an approximation to $Q^\pi(s, .)$. Next, we will describe three different improvement strategies.

The first approach is similar to the one detailed in Subsection 3.3 but requires an additional step. Since we have quantum access to Φ, we use the quantum matrix multiplication procedure from Theorem 2.4 with the μ_Φ-block-encoding of Φ and the output $|\widehat{w}^\pi\rangle$ of the approximate quantum policy evaluation procedure to compute the quantum state $|\Phi\widehat{w}^\pi\rangle$ which is an approximate to the quantum value function $|Q^\pi\rangle$ with cost $\widetilde{\mathcal{O}}(\kappa_\Phi(\mu_\Phi \mathrm{T}_\Phi + \mathrm{T}_{w^\pi}))$. We then perform measurements on this quantum state and update the policy according to the rule:

$$\pi'(s) = \arg\max_a M(s, a) \approx \arg\max_a \langle \Phi w^\pi | s, a\rangle$$

where M is the histogram of the measured state-action pairs $|s, a\rangle$ sampled from $|\Phi\widehat{w}^\pi\rangle$. The total cost of this policy update rule is $\widetilde{\mathcal{O}}(M\kappa_\Phi(\mu_\Phi \mathrm{T}_\Phi + \mathrm{T}_{w^\pi}))$ where the number of measurements can be adjusted in practice according to the arguments provided in Subsection 3.3.

The second approach reconstructs classically an approximation to the output $|\widehat{w}^\pi\rangle$ in order to improve the actual policy. First, we will perform a number of M measurements on $|\widehat{w}^\pi\rangle$ such that we will sample for each measurement some feature index $|k\rangle$ with probability $|\langle\widehat{w}^\pi|k\rangle|^2$. Since the components of $|\widehat{w}^\pi\rangle$ are not necessarily positive, we also need to perform sign estimation of the components of $|w^\pi\rangle$ by performing an additional number of M measurements that query $|\widehat{w}^\pi\rangle$ [9]. Denoting by $M(k)$ the number of times the feature index $|k\rangle$ was sampled and by $\sigma(k)$ the estimated sign of $\langle\widehat{w}^\pi|k\rangle$, the normalized vector with coordinates $\sigma(k)\sqrt{M(k)/M}$ is an approximation to the quantum state $|w^\pi\rangle$. Hence, we can use the following policy improvement rule:

$$\pi'(s) = \arg\max_a \sum_k \sigma(k)\sqrt{M(k)}\phi_k(s,a) \approx \arg\max_a \langle\Phi(s,a)|w^\pi\rangle$$

The total cost of this policy improvement strategy is $\widetilde{\mathcal{O}}(SK + M\mathrm{T}_{w^\pi})$ since we need to perform M measurements in order to reconstruct classically $|\widehat{w}^\pi\rangle$ before performing $\mathcal{O}(K)$ operations to compute $\pi'(s)$ for each $s \in \mathcal{S}$ or $\mathcal{D}$.

The third approach consists of building approximations to the quantum states $|\Phi(s)\widehat{w}^\pi\rangle$ for every state s using quantum matrix-vector multiplication and performing measurements on these quantum states. First, for every state s, we construct a $\mu_{\Phi(s)}$-block-encoding of $\Phi(s)$ that we apply to $|\widehat{w}^\pi\rangle$ in order to compute $|\Phi(s)\widehat{w}^\pi\rangle$ which is as an approximation to $|Q^\pi(s,.)\rangle$ defined as:

$$|Q^\pi(s,.)\rangle = \frac{1}{\|Q^\pi(s,.)\|} \sum_a Q^\pi(s,a)\,|a\rangle$$

Second, we measure the quantum states $|\Phi(s)\widehat{w}^\pi\rangle$ to get an action a with probability $|\langle a|\widehat{Q}^\pi(s,.)\rangle|^2 \approx Q^\pi(s,a)^2/\|Q^\pi(s,\cdot)\|^2$. Similarly to the approach in Subsection 3.3, we construct for every s a histogram of measurements denoted by M such that $M(a)$ is the number of times we measured action when applying the block-encoding of $\Phi(s)$ to $|w^\pi\rangle$. Then, we update the policy according to the rule:

$$\pi'(s) = \arg\max_a M(a) \approx \arg\max_a \langle a|\Phi(s)w^\pi\rangle$$

Let $\kappa_{\Phi|\mathcal{S}}$ and $\mu_{\Phi|\mathcal{S}}$ be upper bounds on the quantities $\kappa_{\Phi(s)}$ and $\mu_{\Phi(s)}$ of $\Phi(s)$ over all states s, the total cost for updating the policy is then $\mathcal{O}(MS\kappa_{\Phi|\mathcal{S}}(\mu_{\Phi|\mathcal{S}}\mathrm{T}_{\Phi|\mathcal{S}} + \mathrm{T}_{w^\pi}))$ since the cost for producing a single quantum state $|\Phi(s)\widehat{w}^\pi\rangle$ is $\mathcal{O}(\kappa_{\Phi(s)}(\mu_{\Phi(s)}\mathrm{T}_{\Phi(s)} + \mathrm{T}_{w^\pi}))$.

We have defined different quantum approximate policy improvement methods that can be used together with the quantum policy evaluation described in previous sections. The quantum approximate policy iteration method with the third improvement strategy, which provides a good method for near term implementations, is used in Algorithm 2. In the next subsections, we are going to discuss how to construct the block-encodings of the features matrix Φ, analyze the running time of our approach and provide a model-free implementation.

4.4 Constructing block-encodings

We are going to show how to build quantum access to the parameters required by Theorem 4.1. We need to construct the block-encodings of the transition matrix P^π and the features matrix Φ and build quantum access to the reward vector $|R\rangle$.

We assume quantum access to $\mathcal{M}$ as in Definition 3.2 for the parameters of the MDP and we have already discussed in Subsection 3.4 how to get a block-encoding for P^π and the procedure that prepares $|R\rangle$. In the following, we apply a similar approach to get a block-encoding for Φ. We extend Definition 3.2 to the approximate case it by assuming access to an additional oracle that encodes the features:

Definition 4.2 (Model-based quantum access). *Let $\Phi \in \mathbb{R}^{SA \times K}$ be a features matrix such that $\|\Phi(s,a)\| = 1$ for every state-action pair (s,a). We say that we have quantum access in the model-based case with cost $(\mathrm{T}_P, \mathrm{T}_R, \mathrm{T}_\Phi)$ if, additionally to the oracles in Definitions 3.2 and 3.3 with cost $(\mathrm{T}_P, \mathrm{T}_R)$, we can implement with cost T_Φ the following oracle and its controlled version for the features matrix Φ:*

$$\mathrm{O}_\Phi : |s,a\rangle |0_k\rangle \longrightarrow |s,a\rangle |\Phi(s,a)\rangle = \sum_{sa} \phi_k(s,a) |s,a\rangle |k\rangle$$

Then, we use the oracle O_Φ to build the block-encoding of Φ as shown in the following lemma:

Lemma 4.1 (Block-encoding of Φ). *Given quantum access to Φ with cost T_Φ, we can implement a $\sqrt{K}$-block-encoding of Φ with cost $\mathcal{O}(\mathrm{T}_\Phi)$.*

Proof. Since the rows of Φ are normalized, we get from O_Φ a $\sqrt{K}$-block-encoding of Φ using Lemma 2.1 with $p = 1$. $\square$

4.5 Running time analysis

We have formally defined the oracles that we need for the implementation of quantum approximate policy iteration and we are going to analyze its running time. From Theorem 4.1, we see that the cost of quantum approximate policy evaluation in the model-based case is:

$$\mathcal{O}\big(\kappa_\Phi^2 \big((\mu_\Phi^2 \mu_{P^\pi} + \mu_\Phi \kappa_\Phi) \mathrm{T}_\Phi + \mu_\Phi^2 \mu_{P^\pi} (\mathrm{T}_P + \mathrm{T}_\pi) + \kappa_\Phi \mathrm{T}_R \big) \Gamma \, \mathrm{polylog}(\kappa_\Phi \Gamma / \epsilon)\big)$$

As we said, we could make the assumption that oracles for the transition matrix P and the policy π can be built in poly-logarithmic depth, and the same for the feature matrix Φ that is hand-picked by us. We also have that the normalizing factor $\mu_\Phi = \sqrt{K}$. We then have the following simplification of the running time of the model-based approximate quantum policy evaluation:

$$\mathrm{T}_{w^\pi} = \mathcal{O}\Big(\kappa_\Phi^2 \Big(K \mu_{P^\pi} + \sqrt{K}\kappa_\Phi\Big)\Gamma \, \mathrm{polylog}(\kappa_\Phi SA\Gamma K/\epsilon)\Big)$$

The overall running time of our algorithm where we apply the greedy update rule as in Algorithm 2 will be $\widetilde{\mathcal{O}}(\kappa_{\Phi|\mathcal{S}} MS \mathrm{T}_{w^\pi})$. Classically, the running time of approximate policy iteration is $\mathcal{O}(SAK^2)$ for the approximate policy evaluation step and $\mathcal{O}(SAK)$ for the policy improvement step. Whether our algorithm provides an advantage over the classical one depends on the number of measurements M required for policy improvement and the properties of the features function Φ, namely the dimension K and the condition numbers κ_Φ and $\kappa_{\Phi|\mathcal{S}}$, which given that we pick the matrix Φ ourselves, we can easily control. Moreover, we do not expect the number of measurements to grow with the size of the state space S since we measure quantum states $|Q^\pi(s,.)\rangle$ of size A which was not the case with $|Q^\pi\rangle$ of size SA.

19

4.6 Model-free implementation

We have defined a quantum algorithm for performing model-based approximate policy iteration where we have access to a model for the MDP $\mathcal{M}$. Next, we are going to show that we can also implement a model-free approach that does not require such access. When P and R are unknown, we assume having access to a source $\mathcal{D}$ containing transition samples of the form $(\tilde{s}, \tilde{a}, \tilde{s}', \tilde{r})$ and we compute an estimate $\widetilde{w}^\pi$ of w^π as a solution to $\mathbf{A}^\pi \widetilde{w}^\pi = \mathbf{b}$ with $\mathbf{A}^\pi = \widetilde{\Phi}^\top (\widetilde{\Phi} - \gamma \widetilde{P^\pi \Phi})$ and $\mathbf{b} = \widetilde{\Phi}^\top \widetilde{R}$ such that $\widetilde{\Phi} \in \mathbb{R}^{D \times K}$, $\widetilde{P^\pi \Phi} \in \mathbb{R}^{D \times K}$ and $\widetilde{R} \in \mathbb{R}^D$ are estimated using the samples from the source $\mathcal{D}$. Denoting by D the number of samples in $\mathcal{D}$, $\widetilde{\Phi} \in \mathbb{R}^{D \times K}$ is the matrix with rows $\phi(\tilde{s}_i, \tilde{a}_i)^\top$ where i denotes the i-th sample of $\mathcal{D}$, $\widetilde{P^\pi \Phi} \in \mathbb{R}^{D \times K}$ is the matrix with rows $\phi(\tilde{s}'_i, \pi(\tilde{s}'_i))^\top$ and $\widetilde{R} \in \mathbb{R}^D$ is the vector with elements $\widetilde{R}_i = \tilde{r}_i$. In the quantum case, we will assume having quantum access to these three quantities and use the quantum linear algebra techniques to build an ϵ-approximation (in ℓ_2-norm) state $|\widehat{w}^\pi\rangle$ to $|\widetilde{w}^\pi\rangle$:

Theorem 4.3 (Model-free evaluation). *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ be a finite or non-finite Markov decision process with unknown model P and R, Φ a features function such that $\|\Phi(s, a)\| = 1$ for every state-action pair (s, a), π a deterministic policy and $\epsilon > 0$ the precision parameter. Suppose there exists a $\mu_{\widetilde{\Phi}}$-block-encoding of the estimated $\widetilde{\Phi}$ and $\widetilde{P^\pi \Phi}$ with cost $\mathrm{T}_{\widetilde{\Phi}}$. Both matrices having singular values ranging in $[1/\kappa_{\widetilde{\Phi}}, 1]$. Also suppose that we can prepare the estimated reward vector $|\widetilde{R}\rangle$ with cost $\mathrm{T}_{\widetilde{R}}$. Then there exists a quantum algorithm that returns a quantum state $|\widehat{w}^\pi\rangle$ such that $\| |\widehat{w}^\pi\rangle - |\widetilde{w}^\pi\rangle \| \leq \epsilon$ with cost:*

$$\mathcal{O}\left(\kappa_{\widetilde{\Phi}}^2 \left(\mu_{\widetilde{\Phi}}^2 \mathrm{T}_{\widetilde{\Phi}} + \kappa_{\widetilde{\Phi}} \mu_{\widetilde{\Phi}} \mathrm{T}_{\widetilde{\Phi}} + \kappa_{\widetilde{\Phi}} \mathrm{T}_R \right) \Gamma \, \mathrm{polylog}\left(\kappa_{\widetilde{\Phi}} \Gamma / \epsilon \right) \right)$$

Proof. Using a similar approach to Theorem 4.1, we can implement a $\mu_{\widetilde{\Phi}}$-block-encoding of $\widetilde{\Phi}^\top$ and generate an $\epsilon' = (1 + \gamma)\epsilon / (2\sqrt{2}\kappa_{\widetilde{\Phi}}^2 \Gamma)$ approximation state $|\widehat{\mathbf{b}}\rangle$ to the state $|\mathbf{b}\rangle = |\widetilde{\Phi}^\top \widetilde{R}\rangle$ with cost:

$$\mathrm{T}_{\mathbf{b}} = \mathcal{O}\left(\kappa_{\widetilde{\Phi}} \left(\mu_{\widetilde{\Phi}} \mathrm{T}_{\widetilde{\Phi}} + \mathrm{T}_{\widetilde{R}} \right) \mathrm{polylog}\left(\kappa_{\widetilde{\Phi}} \Gamma / \epsilon \right) \right)$$

Next, we build the block encoding of $\mathbf{A}^\pi = \widetilde{\Phi}^\top (\widetilde{\Phi} - \gamma \widetilde{P^\pi \Phi})$ using the block-encodings of $\widetilde{\Phi}$, $\widetilde{\Phi}^\top$ and $\widetilde{P^\pi \Phi}$. First, note that we can construct $\mu_{\widetilde{\Phi}}^2$-block-encodings of $\widetilde{\Phi}^\top \widetilde{\Phi}$ and $(\widetilde{\Phi}^\top \widetilde{P^\pi \Phi})$ with cost $\mathcal{O}(\mathrm{T}_{\widetilde{\Phi}})$. Using Theorem 2.3, we can implement a $\mu_{\widetilde{\Phi}}^2(1 + \gamma)$-block-encoding of $(\widetilde{\Phi}^\top \widetilde{\Phi} - \gamma \widetilde{\Phi}^\top \widetilde{P^\pi \Phi})$ with cost $\mathrm{T}_{\mathbf{A}} = \mathcal{O}(\mathrm{T}_{\widetilde{\Phi}})$. Then, we apply the quantum linear solver with precision $\epsilon/2$ to generate a state $|\widehat{w}^\pi\rangle$ such that $\| |\widehat{w}^\pi\rangle - |(\mathbf{A}^\pi)^{-1}\widehat{\mathbf{b}}\rangle \| \leq \epsilon/2$ with cost:

$$\mathcal{O}\left(\kappa(\mathbf{A}^\pi) \left(\|\mathbf{A}^\pi\|^{-1} \mu_{\widetilde{\Phi}}^2 (1 + \gamma) \mathrm{T}_{\widetilde{\Phi}} + \mathrm{T}_{\mathbf{b}} \right) \mathrm{polylog}(\kappa(\mathbf{A}^\pi)/\epsilon) \right)$$

$\square$

The model-free quantum policy evaluation approach above can work together with any of the improvement strategies described in Subsection 4.3. The only difference is that we need to update the policy for all states $s' \in \mathcal{D}$, i.e. we iterate over all next-states s' and update the estimated $\widetilde{P^\pi \Phi}$. The cost analysis is sill valid by replacing the space state size S by the source size D. Next, we describe how to construct the necessary block-encodings in the model-free case.

We are provided with a source $\mathcal{D}$ of transition samples of the form $(\tilde{s}, \tilde{a}, \tilde{s}', \tilde{r})$ and their corresponding features. Similarly to the model-based case, we will assume that the features are normalized for each state-action pair $(\tilde{s}, \tilde{a})$. The following definition gives the list of oracles that we need to implement in order to perform quantum approximate policy evaluation needed for Theorem 4.3:

Definition 4.4 (Model-free quantum access). *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ be a finite or non-finite Markov decision process with unknown model P and R, $\mathcal{D}$ a finite source of transition samples from $\mathcal{M}$ of the form $(\tilde{s}, \tilde{a}, \tilde{s}', \tilde{r})$, Φ a feature function such that $\|\Phi(\tilde{s}, \tilde{a})\| = 1$ for all $(\tilde{s}, \tilde{a})$ and π a deterministic policy. We say that we have quantum access in the model-free case with costs $(T_{\tilde{P}}, T_{\tilde{R}}, T_{\tilde{\Phi}}, T_{\tilde{\pi}})$ if we can implement:*

1. *Two oracles for the transition samples with cost $T_{\tilde{P}}$ such that:*

$$O_{\tilde{P}}^{s,a} : |i\rangle |0_s, 0_a\rangle \longrightarrow |i\rangle |\tilde{s}_i, \tilde{a}_i\rangle$$

$$O_{\tilde{P}}^{s'} : |i\rangle |0_s\rangle \longrightarrow |i\rangle |\tilde{s}_i'\rangle$$

2. *An oracle for the reward samples with cost $T_{\tilde{R}}$ such that:*

$$O_{\tilde{R}} : |0_i\rangle \longrightarrow |\tilde{R}\rangle = \frac{1}{\|\tilde{R}\|} \sum_i \tilde{r}_i |i\rangle$$

3. *An oracle for the features function Φ with cost T_Φ such that:*

$$O_\Phi : |\tilde{s}, \tilde{a}\rangle |0_k\rangle \longrightarrow |\tilde{s}, \tilde{a}\rangle |\Phi(\tilde{s}, \tilde{a})\rangle = \sum_k \phi_k(\tilde{s}, \tilde{a}) |\tilde{s}, \tilde{a}\rangle |k\rangle$$

4. *An oracle for the deterministic policy π with cost $T_{\tilde{\pi}}$ such that:*

$$O_{\tilde{\pi}} : |\tilde{s}, 0_a\rangle \longrightarrow |\tilde{s}, \pi(\tilde{s})\rangle$$

Our quantum policy evaluation algorithm requires a procedure for the estimated vector $|\tilde{R}\rangle$, which is given by the oracle $O_{\tilde{R}}$, and the block-encodings of $\tilde{\Phi}$ and $\widetilde{P^\pi\Phi}$ given by the following lemma:

Lemma 4.2 (Block-encodings of $\tilde{\Phi}$ and $\widetilde{P^\pi\Phi}$). *Given quantum access in the model-free case to $\mathcal{D}$, Φ and π as in Definition 4.4, we can implement a $\sqrt{K}$-block-encoding of $\tilde{\Phi}$ and $\widetilde{P^\pi\Phi}$ with cost $T_{\tilde{\Phi}} = \mathcal{O}(T_{\tilde{P}} + T_\Phi + T_{\tilde{\pi}})$.*

Proof. If we start from the state $|i\rangle |0_k\rangle |0_s, 0_a\rangle$ and apply $O_{\tilde{P}}^{s,a}$ on the first and third registers followed by O_Φ on the third and second register, we get the following mapping after uncomputing the third register using the adjoint operation $(O_{\tilde{P}}^{s,a})^\dagger$:

$$|i\rangle |0_k\rangle \longrightarrow \sum_k \phi_k(\tilde{s}_i, \tilde{a}_i) |i\rangle |k\rangle$$

Similarly, if we start from the state $|i\rangle |0_k\rangle |0_s, 0_a\rangle$ and apply $O_{\widetilde{P}}^{s'}$ followed by $O_{\widetilde{\pi}}$ and O_Φ, we get the following mapping after uncomputing the last register using $(O_{\widetilde{P}}^{s'})^\dagger$:

$$|i\rangle |0_k\rangle \longrightarrow \sum_k \phi_k(\tilde{s}'_i, \pi(\tilde{s}'_i)) |i\rangle |k\rangle$$

Then, we use Lemma 2.1 to construct respectively the block-encodings for $\widetilde{\Phi}$ and $\widetilde{P^\pi \Phi}$ by setting the factor p to be 1. $\qquad\square$

Assuming that the circuits for model-free quantum access can be implemented in poly-logarithmic depth as in Subsection 4.5, the running time of approximate quantum policy iteration simplifies to:

$$\mathcal{O}\left(\kappa_{\widetilde{\Phi}}^3 \sqrt{K} \Gamma \operatorname{polylog}\left(\kappa_{\widetilde{\Phi}} DAK\Gamma/\epsilon\right)\right)$$

5 Applications

In the previous sections, we formulated our quantum policy iteration algorithms using the block-encoding framework and we have explicitly described what quantum oracles we need in order to construct these block-encodings. In this section, we will describe how to implement in practice quantum access to those oracles for the FROZENLAKE and INVERTEDPENDULUM which are two environments listed in OpenAI's Gym [31] and widely used in reinforcement learning.

5.1 Application to FROZENLAKE

Description of the environment: FROZENLAKE is an environment that consists of a two-dimensional grid of size $X \times Y$ where the agent moves around the grid in four directions to reach the goal state without falling into holes. The episode terminates if the agent steps into a hole or reaches the goal state where a reward of $+1$ is perceived. Its state space $\mathcal{S} = \{(x,y)|x \in [X], y \in [Y]\}$ is the set of all grid positions and its action space $\mathcal{A} = \{(0,1),(0,-1),(1,0),(-1,0)\}$ contains the four possible actions: *up*, *down*, *left* and *right*. For example, taking action $a = (0,-1)$ when in state $s = (x,y)$ moves the agent to the next state $s' = s + a = (x, y-1)$.

Quantum access: We want to build quantum access to the MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ associated to this environment by constructing the oracles as in Definition 3.2. In the classical case, we can recover the environment dynamics (P, R) by specifying the goal state s_G and the subset of walkable positions $\mathcal{F} \subset \mathcal{S}$, i.e. positions that are neither holes nor the goal state. Similarly, we show in the following claim that we can build quantum access to $\mathcal{M}$ if we have access to appropriate oracles that encode the subset $\mathcal{F}$ and the state s_G:

Claim 5.1. *Let $\mathbb{1}_\mathcal{F}$ be the indicator function for the subset of walkable positions in the grid. Given quantum access to an oracle $O_\mathcal{F} : |s\rangle |0\rangle \to |s\rangle |\mathbb{1}_\mathcal{F}(s)\rangle$ with cost $\mathrm{T}_\mathcal{F}$ and to an oracle $O_G : |0_s\rangle \to |s_G\rangle$ with cost T_G, we can build quantum access to $\mathcal{M}$ with costs $(\mathrm{T}_P, \mathrm{T}_R) = (\mathcal{O}(\mathrm{T}_\mathcal{F}), \mathcal{O}(\mathrm{T}_G))$.*
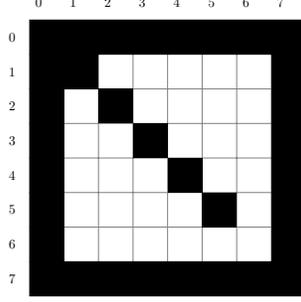
Figure 5: An example of the FROZENLAKE environment with holes located in the diagonal.

Proof. We will build quantum access to $\mathcal{M}$ by constructing the oracles O_P^r, O_P^c and O_R from Definition 3.2. Classically, the transition matrix P and the reward vector R can both be recovered from $\mathcal{F}$ and s_G since:

$$p(s, a, s') = \begin{cases} 1, & \text{if } (s \in \mathcal{F} \text{ and } s' = s + a) \text{ or } (s \notin \mathcal{F} \text{ and } s' = s) \\ 0, & \text{otherwise} \end{cases}$$

$$r(s, a) = \begin{cases} 1, & \text{if } (s \in \mathcal{F} \text{ and } s + a = s_G) \\ 0, & \text{otherwise} \end{cases}$$

First, we will construct the oracle O_P^r. If we assume without loss of generality that all positions located in the borders of the grid are non-walkable, then the oracle O_P^r that encodes the rows of P corresponds to the mapping:

$$O_P^r : |s, a\rangle |0_s\rangle \longrightarrow \begin{cases} |s, a\rangle |s + a\rangle, & \text{if } s \in \mathcal{F} \\ |s, a\rangle |s\rangle, & \text{if } s \notin \mathcal{F} \end{cases}$$

Both mappings $|s, a\rangle |0_s\rangle \to |s, a\rangle |s + a\rangle$ and $|s, a\rangle |0_s\rangle \to |s, a\rangle |s\rangle$ can be implemented in linear cost on the number of qubits used to represent state-action pairs $|s, a\rangle$. If we combine both mappings with $O_{\mathcal{F}}$ applied on one ancilla qubit, we can construct the oracle O_P^r with cost $\mathcal{O}(T_{\mathcal{F}})$.

Next, we will construct the oracle O_P^c that encodes the columns of P. Note that for this particular environment, any next state s' arises as a transition from at most four state-action pairs corresponding to taking the action $a = s' - s$ from adjacent positions s. In other words, the value c_P can be chosen to be 4 and we have:

$$O_P^c : |0_s, 0_a\rangle |s'\rangle \longrightarrow \frac{1}{2} \sum_{s \in \mathcal{F} | s + a = s'} |s, a\rangle |s'\rangle + |G_{s'}^\perp\rangle$$

The mapping $|0_s, 0_a\rangle |s'\rangle \to \frac{1}{2} \sum_{s, a | s + a = s'} |s, a\rangle |s'\rangle$, that builds the superposition of the four possible ancestors of s', can be implemented in linear cost. It can be combined with $O_{\mathcal{F}}$ to construct O_P^c with total cost $\mathcal{O}(T_{\mathcal{F}})$.

Finally, we also need to build the oracle O_R that gives access to the reward vector R. For example, assuming that all four adjacent positions to the goal state s_G are walkable, we can rewrite O_R as:

$$O_R : |0_s, 0_a\rangle \longrightarrow |R\rangle = \frac{1}{2} \sum_a |s_G - a, a\rangle.$$

23

Similar constructions are easy to design when there are some holes around the goal state s_G. The sum on the above oracles contain at most four elements and can be implemented with cost $\mathcal{O}(T_G)$ by first mapping $|0_s\rangle$ to $|s_G\rangle$ using O_G and then mapping $|s_G, 0_a\rangle$ to $|R\rangle$. $\quad\square$

We have shown how to implement the necessary oracles that give quantum access to $\mathcal{M}$. It is important to note that we have set $c_P = 4$ which implies that the normalizing factor $\mu_P = 2$ of the block-encoding of the transition matrix P is fixed and does not depend on the grid size $S = X \times Y$. Given some policy π, the total cost of our quantum policy evaluation procedure from Theorem 3.1 is then $\mathcal{O}(\Gamma(T_{\mathcal{F}} + T_G + T_\pi))$ where T_π is the cost for the oracle encoding π as in Definition 3.3. In the general case, we can implement $O_{\mathcal{F}}$, O_G and O_π with $\mathcal{O}(XY)$ qubits and $\mathcal{O}(\text{polylog}(XY))$ depth. However, there exists specific cases where the implementation of the oracle $O_{\mathcal{F}}$ requires only $\mathcal{O}(\text{polylog}(XY))$ qubits and depth. If for example all the non-walkable positions are located in the diagonal of the grid as in Figure 5, we can implement $O_{\mathcal{F}}$ using elementary mappings that require only $\mathcal{O}(\text{polylog}(XY))$ qubits.

Running time: The running time of quantum policy evaluation is $\mathcal{O}(\Gamma \, \text{polylog}(XY))$ and the total running time of quantum policy iteration is $\mathcal{O}(M\Gamma \, \text{polylog}(XY))$ with M being the number of measurements. Executing the same classical algorithm yields a running time of $\mathcal{O}((XY)^\omega)$ since $A = 4$ and does not depend on XY. Whether or not we have a quantum advantage depends on what is the required value of M. Setting $M = \mathcal{O}(\log(XY)/\epsilon^2)$ may not suffice when XY is very large because the value function concentrates around the goal state and the ℓ_∞-tomography only guarantees ϵ-approximation in average. In this case, we may use ℓ_2-tomography with $M = \widetilde{\mathcal{O}}(XY/\epsilon^2)$ to guarantee that each grid position is sampled enough and the running time becomes $\widetilde{\mathcal{O}}(XY\Gamma/\epsilon^2)$ where the value of ϵ does not depend on the grid size, which still gives us a polynomial speedup over the classical in the worst case.

Experimental results: We simulated the quantum policy iteration on a classical computer by introducing the appropriate noise and randomness within the linear algebraic procedures of the algorithm. More precisely, two types of noise were added to the normalized state-value function evaluated with a classical procedure. Given a precision parameter ϵ, the first noise corresponds to the matrix inversion error (Theorem 2.4) in the quantum policy evaluation method, whereas the second noise corresponds to the sampling error due to the finite number of quantum measurements (Theorem 2.5) where the number of measurements is chosen to be $M = 36\log(SA)/\epsilon^2 = 36\log(4XY)/\epsilon^2$ as in [9]. We used 5 different random seeds to run our experiments on the 4×4 and 8×8 maps for the FROZENLAKE environment [31] and we saw that the quantum policy iteration converges to the optimal policy after at most five iterations for a precision parameter $\epsilon = 10^{-2}$.

5.2 Application to INVERTEDPENDULUM

Description of the environment: INVERTEDPENDULUM is an environment that requires maintaining a pendulum in a stable position by moving the cart it is attached to [32]. The space state $\mathcal{S} \subset \mathbb{R}^2$ is continuous and consists of tuples of the form $s = (\theta, \dot{\theta})$ where $\theta \in [-\pi/2, \pi/2]$ is the vertical angle and $\dot{\theta} \in \mathbb{R}$ the velocity. The action space consists of three Newtonian forces $\mathcal{A} = \{-50N, 0N, +50N\}$ that can be applied to the cart to balance the pendulum. A uniform noise in $[-10, 10]$ is added to any action. The game stops when the angle is greater than $\pi/2$ in absolute value.

The dynamics of the environment are governed by the following equation:

$$\ddot{\theta} = \frac{g\sin(\theta) - \alpha ml\dot{\theta}^2 \sin(2\theta)/2 - \alpha\cos(\theta)a}{4l/3 - \alpha ml\cos^2(\theta)}$$

where g is the gravity constant, m is the mass of the pendulum, M is the mass of the cart, l is the length of the pendulum and $\alpha = 1/(m+M)$.

Quantum access: Since the state space $\mathcal{S}$ is continuous, we will apply the model-free implementation of quantum policy iteration. We want to build quantum access to a source $\mathcal{D}$ of transition samples classically collected from the INVERTEDPENDULUM environment and to some features function Φ by constructing the oracles as in Definition 4.4.

First, let us consider quantum access to $\mathcal{D}$ by constructing $O_{\widetilde{P}}^{s,a}$, $O_{\widetilde{P}}^{s'}$. Assuming that we have a B-bit binary description for the states and actions, we can implement both oracles with cost $T_{\widetilde{P}} = \mathcal{O}(D \times B)$ where D is the number of samples in $\mathcal{D}$. Moreover, we can also implement the oracle $O_{\widetilde{R}}$ with cost $T_{\widetilde{R}} = \mathcal{O}(\text{polylog}(D))$ since all rewards have value $+1$ and the approximated reward vector $|\widetilde{R}\rangle = \sum_i |i\rangle/\sqrt{D}$ can be implemented by applying a Hadamard transform to $|0_i\rangle$.

Next, we will construct an oracle that implements the features function Φ. In particular, we will use the Fourier features [33]. Given some policy π, we will approximate the value function Q^π using a multivariate Fourier series expansion of $Q^\pi(\cdot, a)$ on $[-1,1]^2$:

$$Q^\pi(s,a) = \sum_{\mathbf{c}} \alpha_{\mathbf{c}} \cos(\pi\mathbf{c}\cdot s) + \beta_{\mathbf{c}}\sin(\pi\mathbf{c}\cdot s) \quad \text{with} \quad \mathbf{c} \in \mathbb{N}^{\dim(\mathcal{S})} = \mathbb{N}^2$$

To do so, we rescale the state parameters to range in $[0,1]^2$. We limit the expansion to some degree $k \in \mathbb{N}$ by considering coefficients $\mathbf{c} \in [k]^2 = \{0,\ldots,k-1\}^2$ which results in $2k^2$ features per action and a total number of $K = 2Ak^{\dim(\mathcal{S})} = 6k^2$ features. The features function $\Phi : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}^K$ maps every state-action pair (s,a) to $\Phi(s,a) = \{\phi_{\cos}^{\mathbf{c},\mathbf{a}}(s,a), \phi_{\sin}^{\mathbf{c},\mathbf{a}}(s,a) | \mathbf{c} \in [k]^2, \mathbf{a} \in \mathcal{A}\}$ where:

$$\phi_{\cos}^{\mathbf{c},\mathbf{a}}(s,a) = \mathbb{1}[\mathbf{a} = a]\cos(\pi\mathbf{c}\cdot s) \quad \text{and} \quad \phi_{\sin}^{\mathbf{c},\mathbf{a}}(s,a) = \mathbb{1}[\mathbf{a} = a]\sin(\pi\mathbf{c}\cdot s)$$

In the next claim, we show how to efficiently implement the oracle O_Φ associated to the features function Φ:

Claim 5.2. *Let $k \in \mathbb{N}$ be the Fourier degree expansion and B the number of bits used to describe s. We can implement the oracle O_Φ with cost $T_\Phi = \mathcal{O}(B\,\text{polylog}(K))$.*

Proof. We need to build quantum access to the oracle $O_\Phi : |s,a\rangle|0_k\rangle \to |s,a\rangle|\Phi(s,a)\rangle$. It is important to note that $\|\Phi(s,a)\|$ is constant for all state-action pairs and that the corresponding quantum state $|\Phi(s,a)\rangle$ can be written as:

$$|\Phi(s,a)\rangle = \frac{1}{k}\sum_{\mathbf{c}\in[k]^2} (\cos(\pi\mathbf{c}\cdot s)|0\rangle + \sin(\pi\mathbf{c}\cdot s)|1\rangle)|\mathbf{c},a\rangle$$

Note that the features register $|0_k\rangle$ can be decomposed into three registers $|0\rangle|0_{\mathbf{c}}, 0_a\rangle$ that index the $K = 6k^2$ features. Moreover, the state representation $|s\rangle$ can also be decomposed into $|\theta, \dot{\theta}\rangle$ and we assume that every observation is encoded as a B-bit binary description.

Figure 6: INVERTEDPENDULUM environment.

Starting from $|s,a\rangle |0_k\rangle = |s,a\rangle |0\rangle |0_{\mathbf{c}}, 0_a\rangle$, we can map $|0_{\mathbf{c}}\rangle$ to $\sum_{\mathbf{c}\in[k]^2} |\mathbf{c}\rangle /k$ with cost $\mathcal{O}(\mathrm{polylog}(K))$. Then we use B-ancilla qubits to compute and store the results of the mapping $|s\rangle |\mathbf{c}\rangle |0_B\rangle \longrightarrow |s\rangle |\mathbf{c}\rangle |\mathbf{c}\cdot s\rangle$ with cost $\mathcal{O}(B\,\mathrm{polylog}(K))$ using quantum circuits for addition and multiplication. Next, we control on $|\mathbf{c}\cdot s\rangle$ to map the first qubit of the features register to $(\cos(\pi\mathbf{c}\cdot s)|0\rangle + \sin(\pi\mathbf{c}\cdot s)|1\rangle)$ with cost B. Finally, we finish by uncomputing $|\mathbf{c}\cdot s\rangle$ and copy the action register of $|s,a\rangle$ to $|0_a\rangle$. $\qquad\square$

We have shown how to implement the necessary oracles that give quantum access to the parameters of our model-free quantum approximate policy evaluation. Using Lemma 4.2, we can use these oracles to construct $\sqrt{K}$-block-encodings of the matrices $\widetilde{\Phi}$ and $\widetilde{P^\pi\Phi}$ with cost $\mathrm{T}_{\widetilde{\Phi}} = \mathcal{O}(BD + B\,\mathrm{polylog}(K))$ where we assumed that the implementation of $\widetilde{\pi}$ has the same cost as the one of $\widetilde{P}$. The total cost of our evaluation procedure from Theorem 4.3 simplifies to $\mathcal{O}(\kappa_{\widetilde{\Phi}}^2 BD\Gamma\,\mathrm{polylog}(K\kappa_{\widetilde{\Phi}}/\epsilon))$.

Running time: As demonstrated in the analysis above, the implementation of the oracles that give access to the memory D require at most $\mathcal{O}(BD)$ qubits and can be performed with constant depth. However, the implementation of the features function uses quantum circuits with a linear dependency on B since we need to control on the B qubits used to store the values $\mathbf{c}\cdot s$. The running time of quantum approximate policy evaluation simplifies then to $\mathcal{O}(\kappa_{\widetilde{\Phi}}^2 B\,\mathrm{polylog}(K\kappa_{\widetilde{\Phi}}/\epsilon))$. Since we need to improve the policy for every transition state s in $\mathcal{D}$, the total running time of quantum approximate policy iteration is then $\widetilde{\mathcal{O}}(\kappa_{\widetilde{\Phi}|\mathcal{S}}\kappa_{\widetilde{\Phi}}^2 MDB\,\mathrm{polylog}(K\kappa_{\widetilde{\Phi}}/\epsilon))$ where M is the total number of measurements performed to update one state s. In comparison, executing classically this algorithm takes $\widetilde{\mathcal{O}}(BDK^2 + BK^w)$ for the approximate policy evaluation that computes $\widehat{w}^\pi =$ and $\widetilde{\mathcal{O}}(BDK)$ for the approximate policy improvement step where we use the bit time complexity for matrix multiplication and inversion. Our analysis show that both approaches have linear dependency on B and D, however our quantum algorithm provides a polynomial speedup in the total number of features $K = 2Ak^{\dim(\mathcal{S})}$ that grows exponentially with the dimension of the state space if we apply this approach to other environments.

Experimental results: We simulated the model-free implementation of quantum approximate policy iteration on a classical computer with a Fourier expansion of degree $k = 4$ for a total of $K = 96$ features per state-action pair. Similarly to the experiments in Subsection 5.1, we added a noise of magnitude ϵ to all algebraic procedures and we performed $M = 100$ measurements for every sample in the memory. We preprocessed the state to range in $[0,1]^2$ by normalizing the angle and clipping the angle velocity $\dot{\theta}$ between $[-1,1]$ before rescaling to $[0,1]$. All other simulation parameters are identical to those in [23]. Moreover, we also clipped the singular values of $\mathbf{A}^\pi = \widetilde{\Phi}^\top(\widetilde{\Phi} - \gamma\widetilde{P^\pi\Phi})$ so that its corresponding condition number is constant and has value $\kappa = 1/10^{-3}$. We repeated the experiment over 5 different random seeds with a precision $\epsilon = 10^{-2}$ and saw that our algorithm converges to the optimal policy within the first 8 iterations.

26

6 Conclusion and discussions

In this work, we provided a general framework for performing quantum reinforcement learning via exact and approximate policy iteration. We validated our framework by designing and analyzing quantum policy evaluation methods for infinite horizon discounted problems by building quantum states that approximately encode the value function of a policy π, and quantum policy improvement methods by post-processing measurement outcomes on these quantum states. In all cases, we provided details about constructing block encodings for all matrices needed in the quantum linear algebra computations. Last, we studied the theoretical and experimental performance of our quantum algorithms on the FROZENLAKE and INVERTEDPENDULUM environments.

Our framework can be adapted and generalized to encompass many different policy iteration algorithms, including ones using deep learning techniques, and, of course, further theoretical work is needed in order to fully understand the strengths and limits of this approach. We conclude by providing several directions for possible future work.

First, the cost and running time of our quantum policy evaluation algorithms have linear dependency on the quantities μ and κ of the different matrices appearing in the linear systems used to compute or estimate the value function. The condition number κ is a property of the matrix and cannot be optimized, but one can use a much smaller threshold κ_{th}, thus disregarding smaller eigenvalues, a method that works well when there is a good low rank approximation of the matrix. The quantity μ depends on the procedure used to build quantum access as in the block-encoding framework and different methods will provide different μ parameters. We have provided examples where both these parameters are small, but it remains open to understand the families of environments for which quantum linear algebra can be faster than classical methods.

Second, we provided several quantum policy improvements strategies that consist of performing a series of measurements on the outputs of quantum policy evaluation to update the actual policy. Again, the running time of one iteration of our algorithm is linearly dependent on the number of measurements which also affects the overall performance of our policy. Moreover, there is also inherent noise induced from measurements that is specific to the quantum procedures. We have set this number, for most of our results, to be $\widetilde{\mathcal{O}}(1/\epsilon^2)$ for the theoretical guarantees provided by ℓ_∞-tomography but this number may be far from optimal. Possible research directions include adaptively controlling this number or/and making it state-dependent to appropriately balance between exploration-exploitation. If no exploration is needed, we can instead focus on finding the correct argmax using the quantum maximum finding algorithm by Dürr and Høyer [34] similarly to the approach in [22]. Understanding better how the number of measurements affects the convergence and performance of the quantum reinforcement learning methods needs to be more thoroughly explored.

Last, one may also study the different variants of classical policy iteration that exist and try to provide similar theoretical guarantees of convergence for some appropriate norm for the quantum case.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[2] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, L. Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[3] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandão, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Benjamin Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin G. Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan K. Ho, Markus Hoffmann, Trent Huang, T. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul Klimov, Sergey Knysh, Alexander N. Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry I. Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew J. McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles J. Neill, Murphy Yuezhen Niu, Eric P. Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor Gilbert Rieffel, Pedram Roushan, Nicholas C Rubin, Daniel Thomas Sank, Kevin J Satzinger, Vadim N. Smelyanskiy, Kevin J. Sung, Matthew D Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, P. Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019.

[5] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10:631–633, 2014.

[6] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *ArXiv*, abs/1603.08675, 2017.

[7] Jacob D. Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549:195–202, 2017.

[8] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121 4:040502, 2018.

[9] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. In *International Conference on Learning Representations*, 2019.

[10] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101:022316, 2020.

[11] Iordanis Kerenidis, Jonas Landman, and Natansh Mathur. Classical and quantum algorithms for orthogonal neural networks. *ArXiv*, abs/2106.07198, 2021.

[12] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16:285–286, 2005.

[13] Daoyi Dong, Chunlin Chen, Hanxiong Li, and Tzyh-Jong Tarn. Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1207–1220, 2008.

[14] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[15] Arjan Cornelissen. Quantum gradient estimation and its application to quantum reinforcement learning. Master's thesis, Delft University of Technology, 2018.

[16] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. SIAM, 2019.

[17] Pooya Ronagh. Quantum algorithms for solving dynamic programming problems. *ArXiv*, abs/1906.02229, 2019.

[18] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational quantum circuits for deep reinforcement learning. *IEEE Access*, 8:141007–141024, 2020.

[19] Owen Lockwood and M. Si. Reinforcement learning with quantum variational circuits. *ArXiv*, abs/2008.07524, 2020.

[20] Andrea Skolik, Sofiène Jerbi, and Vedran Dunjko. Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *ArXiv*, abs/2103.15084, 2021.

[21] Sofiène Jerbi, Casper Gyurik, Simon Marshall, Hans J. Briegel, and Vedran Dunjko. Variational quantum policies for reinforcement learning. *ArXiv*, abs/2103.05577, 2021.

[22] Daochen Wang, Aarthi Sundaram, Robin Kothari, Ashish Kapoor, and Martin Rötteler. Quantum algorithms for reinforcement learning with a generative model. In *ICML*, 2021.

[23] Michail G. Lagoudakis and Ronald E. Parr. Least-squares policy iteration. *J. Mach. Learn. Res.*, 4:1107–1149, 2003.

[24] Dimitri P. Bertsekas. Approximate policy iteration: a survey and some new methods. *Journal of Control Theory and Applications*, 9:310–335, 2011.

[25] Bruno Scherrer, Victor Gabillon, Mohammad Ghavamzadeh, and Matthieu Geist. Approximate modified policy iteration. *ArXiv*, abs/1205.3054, 2012.

[26] Dimitri P. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[27] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[28] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster hamiltonian simulation. *ArXiv*, abs/1804.01973, 2018.

[29] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019.

[30] Rémi Munos. Error bounds for approximate policy iteration. In *ICML*, volume 3, pages 560–567, 2003.

[31] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *ArXiv*, abs/1606.01540, 2016.

[32] Hua O. Wang, Kazuo Tanaka, and Michael F. Griffin. An approach to fuzzy control of nonlinear systems: stability and design issues. *IEEE Trans. Fuzzy Syst.*, 4:14–23, 1996.

[33] George Konidaris, Sarah Osentoski, and Philip Thomas. Value function approximation in reinforcement learning using the fourier basis. In *Twenty-fifth AAAI conference on artificial intelligence*, 2011.

[34] Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.