Quantum Algorithms for Reinforcement Learning with a Generative Model

Daochen Wang∗ Aarthi Sundaram† Robin Kothari‡
Ashish Kapoor§ Martin Roetteler¶

Abstract

Reinforcement learning studies how an agent should interact with an environment to maximize its cumulative reward. A standard way to study this question abstractly is to ask how many samples an agent needs from the environment to learn an optimal policy for a $\gamma$-discounted Markov decision process (MDP). For such an MDP, we design quantum algorithms that approximate an optimal policy ($\pi^*$), the optimal value function ($v^*$), and the optimal Q-function ($q^*$), assuming the algorithms can access samples from the environment in quantum superposition. This assumption is justified whenever there exists a simulator for the environment; for example, if the environment is a video game or some other program. Our quantum algorithms, inspired by value iteration, achieve quadratic speedups over the best-possible classical sample complexities in the approximation accuracy ($\epsilon$) and two main parameters of the MDP: the effective time horizon ($\frac{1}{1-\gamma}$) and the size of the action space ($A$). Moreover, we show that our quantum algorithm for computing $q^*$ is optimal by proving a matching quantum lower bound.

1 Introduction

Markov Decision Processes (MDPs) are a fundamental mathematical abstraction in reinforcement learning, used to model problems where an agent should take actions in an environment to maximize its cumulative reward. The framework has been successfully applied to problems in healthcare, robotics, engineering, gaming, natural language processing, finance, and so on [Ber00, Ber13, Sze10, SB18, AJKW21].

Quantum computers are a model of computation based on the laws of quantum mechanics that promise substantially faster algorithms for certain tasks like search and factoring [Gro96, Sho97]. Recent experiments have achieved key milestones [AAB+19], bringing forward the tantalizing prospect of using quantum computers for real-world impact in the not-so-distant future.

In this paper, we construct quantum algorithms that more efficiently solve the main problems associated with MDPs: approximating an optimal policy, the optimal value function, and the optimal Q-value function. Our algorithms rely on the assumption that we have quantum access to the environment, which we will justify.

We intend this introduction to be accessible to those unfamiliar with quantum computing, and we have delayed technical discussions of quantum algorithms to Section 2.

∗University of Maryland. wdaochen@gmail.com
†Microsoft Quantum. aarthi.sundaram@microsoft.com
‡Microsoft Quantum. robin.kothari@microsoft.com
§Microsoft. akapoor@microsoft.com
¶Microsoft Quantum. martinro@microsoft.com

Note: a conference version of this work appears as [WSK+21] (ICML 2021).
1.1 Problem Setup

We study an infinite-horizon discounted MDP, $M$, with a finite set, $S$, of states, where at each state an agent can choose to take an action from a finite set, $A$, of actions. Upon taking an action $a \in A$ at state $s \in S$, the agent receives reward $r[s, a] \in [0, 1]$ and transitions to a state $s' \in S$ with some probability $p(s'|s, a)$. The last parameter needed to specify $M$ is the discount factor $\gamma \in [0, 1)$, which discounts the reward the agent receives at later time steps $t$ by a factor of $\gamma^t$. Hence $M$ is conveniently summarized by a 5-tuple, $M = (S, A, p, r, \gamma)$. For convenience, we define $S = |S|$ and $A = |A|$, the cardinalities of $S$ and $A$ respectively, and $\Gamma := (1 - \gamma)^{-1}$ for the effective time horizon of the MDP.

Given such an MDP, the agent’s goal is to choose actions to maximize its expected sum of $\gamma$-discounted rewards over infinitely many time steps. Following standard practice, we assume the agent has full knowledge of $S$, $A$, $r$, and $\gamma$, but not $p$ at the outset. A primary objective is to compute a deterministic policy $\pi : S \to A$ for the agent that specifies the action $a = \pi(s)$ it should take at $s \in S$ to best achieve its goal with high probability.

For a given policy $\pi : S \to A$, the value-function (or simply value) of $\pi$, $v^\pi : S \to [0, \Gamma]$, and the $Q$-function of $\pi$, $q^\pi : S \times A \to [0, \Gamma]$, are defined by

$$v^\pi[s] = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r[s_t, a_t] \mid s_0 = s, \forall i \geq 0 : a_i = \pi[s_i] \right],$$

$$q^\pi[s, a] = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r[s_t, a_t] \mid s_0 = s, a_0 = a, \forall i \geq 1 : a_i = \pi[s_i] \right],$$

where the expectations are over the probabilistic state transitions, i.e., for all $i \geq 0$, $s_{i+1}$ is sampled from the distribution $p(\cdot|s_i, a_i)$. Note that the maximum value that the sums in Eq. (1) can take is $\Gamma$, because the reward function is at most 1, and hence $v^\pi[s]$ and $q^\pi[s, a]$ are in $[0, \Gamma]$. It is known that any such MDP admits an optimal policy $\pi^* : S \to A$, in the strong sense that $v^{\pi^*}[s] \geq v^\pi[s]$ and $q^{\pi^*}[s, a] \geq q^\pi[s, a]$ for all $\pi \in \Pi$, $s \in S$, and $a \in A$, where $\Pi$ is the space of all policies (which could even contain randomized and non-stationary policies)$^2$ [AJKW21]. It is common to define $v^* := v^{\pi^*}$ and $q^* := q^{\pi^*}$.

We can now state our main computational tasks precisely. Using $\| \cdot \|$ for the infinity norm of a vector, for a given MDP $M$, $\epsilon \in (0, \Gamma)$, and $\delta \in (0, 1)$, our goal is to compute a policy $\hat{\pi}$ for $M$ such that with probability at least $1 - \delta$, it satisfies $\|v^* - v^{\hat{\pi}}\| \leq \epsilon$. In addition, we are interested in the related tasks of computing approximations $\hat{v}$ (resp. $\hat{q}$) to $v^*$ (resp. $q^*$) such that $\|v^* - \hat{v}\| \leq \epsilon$ (resp. $\|q^* - \hat{q}\| \leq \epsilon$) with probability at least $1 - \delta$.

The goal of this paper is to design algorithms that perform the above computational tasks using as few resources as possible. The resource use of an algorithm is normally quantified by either its time complexity or by the number of samples it draws from the unknown distribution $p(s'|s, a)$. In our paper, we study the latter and assume the generative model of sampling, as studied by [KS99, KMN02, Kak03], where we can choose an arbitrary $(s, a) \in S \times A$ and ask a simulator to draw samples $s' \sim p(\cdot|s, a)$. Our goal then translates to minimizing the number of uses of the

---

1We use square brackets to index into vectors and functions.

2In a randomized policy, the action taken at a given $s \in S$ may be probabilistic. A stationary policy is one where the action taken depends only on the current state $s$.

3It is common in the field of MDPs to have $\epsilon$ denote the additive approximation error of a number in $[0, \Gamma]$, which makes the valid range of $\epsilon$ be $(0, \Gamma)$. A more natural normalization may be to divide the $q$ and $v$ functions by $\Gamma$ to have $\epsilon \in (0, 1)$, but this changes what the sample complexity expressions look like, making it harder to visually compare our bounds with prior work.
Table 1: Quantum computing allows for speedups in terms of the parameters \( \epsilon, \Gamma := (1 - \gamma)^{-1} \), and \( A \), but not \( S \). All bounds are for maximum failure probability \( \delta \) constant. All upper bounds are \( \tilde{O}(\cdot) \), with unrestricted \( \epsilon \) except when [Theorem 5] appears, in which case we assume \( \epsilon \in O(1/\sqrt{\Gamma}) \). All lower bounds are \( \Omega(\cdot) \) and hold for any \( \epsilon \in (0, \Gamma/4) \). The classical upper bounds are shown in [LWC+20] for all \( \epsilon \); the classical lower bounds are shown in [AMK12] for \( q^*, v^* \) and [SWW+18] for \( \pi^* \). We also reprove all three classical lower bounds in Theorem 8.

**Fact 1.** Any classical circuit \( G \) with \( N \) logic gates can be converted to a quantum circuit consisting of \( O(N) \) logic gates that can compute on any quantum superposition of inputs; moreover, the conversion is efficient and based on simple conversion rules at the logic gate level.

We refer to \( \mathcal{G} \) as the (quantum) oracle or simulator and the ability to query it as the (quantum) generative model. The precise behavior of \( \mathcal{G} \) is formally defined in Section 2.3.

Under this setup, our goal is to design quantum algorithms approximating \( q^*, \pi^* \), and \( v^* \) that use the quantum simulator \( \mathcal{G} \) as few times as possible. We refer to the number of calls a quantum algorithm makes to \( \mathcal{G} \) as its (quantum) query or sample complexity. It is fair to compare the quantum sample complexity with the classical sample complexity because, as we have discussed above, \( \mathcal{G} \) and \( G \) have similar costs at the elementary gate-level.

Our paper constructs quantum algorithms having significantly less sample complexity than the best-possible classical algorithms. Moreover, we show that our quantum algorithms are either optimal, or optimal assuming \( \Gamma \) or \( A \) is constant, for certain ranges of \( \epsilon \).

### 1.2 Main Results

Table 1 summarizes our main results. The classical sample complexities have only recently been completely characterized for all three quantities [LWC+20] for the full range of \( \epsilon \in (0, \Gamma] \). As the table shows, for computing \( q^* \), we construct a quantum algorithm that offers a quadratic speedup in terms of \( \Gamma \) and \( \epsilon \) if \( \epsilon = O(1/\sqrt{\Gamma}) \). For computing \( v^* \) and \( \pi^* \), we construct a second quantum algorithm that offers an additional quadratic speedup in terms of \( A \) at the expense of \( \Gamma \). Moreover,
we prove quantum lower bounds for computing all three quantities. Our lower bounds show that our \( q^* \) algorithm is optimal, that we have optimal algorithms for \( v^* \) and \( \pi^* \) provided one of \( \Gamma \) or \( A \) is constant, but that there may still be a faster quantum algorithm for \( v^* \) and \( \pi^* \). We remark that we also reprove the classical lower bounds in a qualitatively stronger way than existing bounds as explained at the end of the next section.

We remark that the time complexities of our quantum algorithms are the same as their sample complexities up to log factors assuming that the classical generative model can be called in constant time and that we have access to quantum random access memory (QRAM) [GLM08]. This is because the classical algorithm of [SWW+18] that we quantize satisfies this property and the quantum subroutines we use to quantize it also satisfy this property.

1.3 Technical Overview

We now give an overview of the techniques we used in our two quantum algorithms, \texttt{SolveMdp1} and \texttt{SolveMdp2}. \texttt{SolveMdp1} and \texttt{SolveMdp2} correspond to the complexities next to [Theorem 5] and [Theorem 7] in Table 1 respectively. Our two quantum algorithms are essentially the product of infusing quantum subroutines into a modern variant of (approximate) value iteration by [SWW+18]. We first discuss the quantum subroutines: quantum mean estimation [BHMT02, Mon+15] and quantum maximum finding [DH96].

**Quantum subroutines.** Quantum mean estimation consists of two similar quantum algorithms \texttt{qEst1} and \texttt{qEst2} that we also refer to collectively as \texttt{qEst}. Here, \texttt{qEst} can compute the mean \( \mathbb{E}[X] \) of a random variable \( X \), suitably encoded quantumly, quadratically more efficiently than what is possible classically. \texttt{qEst1} roughly corresponds to a quadratically more sample-efficient Hoeffding’s inequality while \texttt{qEst2} roughly corresponds to a quadratically more sample-efficient Chebyshev’s (or Bernstein’s) inequality. That is, getting additive error \( \epsilon \) using these quantum algorithms takes quadratically fewer samples than what those classical inequalities imply. For example, Chebyshev’s inequality states that \( O(\text{Var}[X]/\epsilon^2) \) samples is required; \texttt{qEst2} roughly states that only \( O(\sqrt{\text{Var}[X]/\epsilon}) \) quantum samples is required. Using quantum mean estimation in both \texttt{SolveMdp1} and \texttt{SolveMdp2} yields the speedups in \( \Gamma \) and \( \epsilon \).

Quantum maximum finding, denoted \texttt{qArgmax}, is an algorithm that can find the maximum of a list of \( n \) numbers, again suitably encoded quantumly, using only \( O(\sqrt{n}) \) queries to that list. \texttt{qArgmax} is used in \texttt{SolveMdp2} and is the source of its speedup in \( A \).

**Quantum version of standard value iteration.** We will be discussing how the above subroutines can be used in the modern variant of value iteration by [SWW+18]. To warm up, consider how they can be applied to standard value iteration [KS99] to compute \( v^* \). In standard value iteration, we start with \( v_0 \) set to the zero vector in \( \mathbb{R}^S \) and repeatedly update it by the Bellman recursion \( v_i \leftarrow \mathcal{T}(v_{i-1}) \) where the Bellman operator \( \mathcal{T} : \mathbb{R}^S \to \mathbb{R}^S \) is defined by

\[
\mathcal{T}(v_{i+1})[s] := \max_{a} \left\{ r[s,a] + \gamma \mathbb{E}[v_i[s'] \mid s' \sim p(\cdot|s,a)] \right\},
\]  

for all \( s \in S \). For convenience, we denote the mean \( \mathbb{E}[v_i[s'] \mid s' \sim p(\cdot|s,a)] \) by \( \mu_i \). Hypothetically, if this mean were computed exactly at each iteration, then this is a contraction map with contraction factor \( \gamma \) and fixed point \( v^* \). So after \( t \) iterations, the error in the current iterate has dropped by a factor of \( \gamma^t \). Neglecting log factors, after about \( O(\Gamma) \) iterations, our iterate is \( \epsilon \)-close to \( v^* \). In reality, we cannot compute \( \mu_i \) exactly. But if we only require our final answer to be correct to error \( \epsilon \), then it is reasonable to assume that estimating \( \mu_i \) for each \( i \) to error \( O(\epsilon/\Gamma) \) suffices. If we make the reasonable assumption \( ||v_i|| \leq ||v^*|| \leq \Gamma \) (\( v_i \) is converging to \( v^* \) after all) then classically
Algorithm 1 SolveMdp1($M, \epsilon, \delta$)

1: **Input**: MDP $M = (S, A, p, r, \gamma)$, maximum error $\epsilon \in (0, \sqrt{\Gamma})$, and maximum failure probability $\delta \in (0, 1)$.
2: **Output**: $\hat{v} := v_{K,L} \in \mathbb{R}^S$, $\hat{\pi} := \pi_{K,L} \in \mathcal{A}^S$, and $\hat{q} := q_{K,L} \in \mathbb{R}^{SA}$.
3: **Initialize**: $K \leftarrow \lceil \log_2(\Gamma / \epsilon) \rceil$, $L \leftarrow \Gamma \lceil \ln(4\Gamma / \epsilon) \rceil + 1$, $f \leftarrow \delta / 4KLSA$, $b \leftarrow 1$, $c \leftarrow 0.01$
4: **Initialize**: $v_{1,0} \leftarrow 0$, $\pi_{1,0} \leftarrow$ arbitrary, $q_{1,0} \leftarrow 0$
5: **for** $k \in [K]$ **do**
6: $\epsilon_k \leftarrow \Gamma / 2^k$
7: $\forall (s, a) \in S \times A$
8: $y_k[s, a] \leftarrow \max \{q_{Est1_f}((Pv_{k,0})[s, a], b) - (q_{Est1_f}((Pv_{k,0})[s, a], (1 - \gamma)b))^2, 0 \}$
9: $x_k[s, a] \leftarrow q_{Est2_f}((Pv_{k,0})[s, a], c(1 - \gamma)1.5\sqrt{y_k[s, a] + b}) - c(1 - \gamma)1.5\sqrt{y_k[s, a] + b}$
10: **for** $l \in [L]$ **do**
11: $\forall s \in S$: if $v(q_{k,l-1})[s] \geq v_{k,l-1}[s]$ then $v_{k,l}[s] \leftarrow v(q_{k,l-1})[s]$, $\pi_{k,l}[s] \leftarrow \pi(q_{k,l-1})[s]$
12: else $v_{k,l}[s] \leftarrow v_{k,l-1}[s]$, $\pi_{k,l}[s] \leftarrow \pi_{k,l-1}[s]$ **end if**
13: $\forall (s, a) \in S \times A$: $\Delta_{k,l}[s, a] \leftarrow q_{Est1_f}((Pv_{k,l} - v_{k,l,0})[s, a], c(1 - \gamma)\epsilon_k) - c(1 - \gamma)\epsilon_k$
14: $q_{k,l} \leftarrow \max \{\gamma + (x_k + \Delta_{k,l}), 0 \}$
15: **end for**
16: $v_{k+1,0} \leftarrow v_{k,L}$, $\pi_{k+1,0} \leftarrow \pi_{k,L}$, $q_{k+1,0} \leftarrow q_{k,L}$
17: **end for**

Algorithm 2 SolveMdp2($M, \epsilon, \delta$)

1: **Input**: MDP $M = (S, A, p, r, \gamma)$, maximum error $\epsilon \in (0, \Gamma]$, and maximum failure probability $\delta \in (0, 1)$.
2: **Output**: $\hat{v} := v_L \in \mathbb{R}^S$ and $\hat{\pi} := \pi_L \in \mathcal{A}^S$.
3: **Initialize**: $L \leftarrow \Gamma \lceil \log(4\Gamma / \epsilon) \rceil + 1$, $f \leftarrow \delta / 4c_{max}LSA^{1.5} \log(1 / \delta)$
4: **Initialize**: $v_0 \leftarrow 0$, $\pi_0 \leftarrow$ arbitrary, $\forall s \in S$: $q_{0,s} \leftarrow 0 \in \mathbb{R}^A$
5: **for** $l \in [L]$ **do**
6: $\forall s \in S$: $a^*[s] \leftarrow \text{Argmax}_f \{q_{l-1,s}[a] : a \in A \}$
7: $\forall s \in S$: $\tilde{\pi}_{l}[s] \leftarrow a^*[s]$, $\tilde{v}_{l}[s] \leftarrow q_{l-1,s}[a^*[s]]$
8: $\forall s \in S$: if $\tilde{v}_{l}[s] \geq \tilde{v}_{l-1}[s]$ then $\tilde{v}_{l}[s] \leftarrow \tilde{v}_{l}[s]$, $\tilde{\pi}_{l}[s] \leftarrow \tilde{\pi}_{l}[s]$
9: else $\tilde{v}_{l}[s] \leftarrow \tilde{v}_{l-1}[s]$, $\tilde{\pi}_{l}[s] \leftarrow \tilde{\pi}_{l-1}[s]$ **end if**
10: $\forall s \in S$: create quantum oracle encoding, $U_{z_{l,s}}$, of $z_{l,s} \in \mathbb{R}^A$ defined by $z_{l,s}[a] \leftarrow q_{Est1_f}((Pv_{l})[s, a], (1 - \gamma)\epsilon/4) - (1 - \gamma)\epsilon/4$
11: $\forall s \in S$: create quantum oracle encoding, $U_{q_{l,s}}$, of $q_{l,s} \in \mathbb{R}^A$ defined by $q_{l,s}[a] \leftarrow \max \{r[s, a] + \gamma z_{l,s}[a], 0 \}$
12: **end for**
doing this estimation at each iteration uses \(O(SA\Gamma^2/(\epsilon/\Gamma)^2) = O(SA\Gamma^4/\epsilon^2)\) samples classically by the Hoeffding bound. The factor \(SA\) comes from the fact that an estimation is done for each \((s,a) \in S \times A\). Therefore, the overall classical sample complexity is of order \(O(SA\Gamma^5/\epsilon^2)\). Though the preceding argument is non-rigorous, it does give the right answer (up to log-factors) [SWWWY21].

How would our quantum subroutines speed up standard value iteration? By using quantum mean estimation, we can quadratically suppress the sample complexity at each iteration and for each \((s,a) \in S \times A\), meaning that the quantum sample complexity at each iteration becomes \(O(SA\sqrt{\Gamma^2}/(\epsilon/\Gamma)^2) = O(SA\Gamma^2/\epsilon)\). Accounting for the \(\Gamma\) iterations, gives an overall quantum sample complexity of \(O(SA\Gamma^3/\epsilon)\). In fact, observing that the Bellman recursion involves taking the maximum over the set of actions, we can use quantum maximum finding to reduce the complexity down further, to \(O(S\sqrt{\Gamma}^3/\epsilon^2)\), which matches the performance of \texttt{SolveMdp2} for \(v^*\). However, an \(\epsilon\)-optimal value function leads only to an \((2\gamma\Gamma)\)-optimal greedy policy [SY94, Ber13].

**Quantum version of modern value iteration.** To obtain an \(\epsilon\)-optimal policy, \texttt{SolveMdp1} and \texttt{SolveMdp2} directly employ the so-called monotonicity technique of [SWWW+18] which we observe does not interfere with our use of the two quantum subroutines. The monotonicity technique comprises the if-then-else statement and the subtractions in the lines involving \(q\texttt{Est}\). Note that the subtracted terms always equal the preceding estimation error which enforces one-sided error. Overall, the monotonicity technique ensures that the value function at each iteration is at most the value function of the policy at that iteration (which in turn is at most \(v^*\)). Hence we avoid the problem of an \(\epsilon\)-optimal \(\hat{v}\) not giving an \(\epsilon\)-optimal \(\hat{\pi}\).

We can get better dependence in \(\Gamma\) by leveraging two other techniques introduced in [SWWW+18, SWWWY21, Wai19]: “variance reduction” and “total variance”. We incorporate these techniques in \texttt{SolveMdp1} at the cost of re-inflating the \(A\) dependence back to linear. The reason we no longer get \(\sqrt{A}\) is because applying \(q\texttt{Argmax}\) is incompatible with the variance reduction technique.

Variance reduction essentially splits standard value iteration into \(K := \lceil \log_2(\Gamma/\epsilon) \rceil\) epochs where in each epoch we halve the error. Epochs in \texttt{SolveMdp1} are indexed by \(k\). At the \(l\)-th iteration of epoch \(k\), we need to estimate \(E[v_{k,l}[s']]\), where \(v_{k,l}\) is the current value function. The mean can be rewritten as

\[
E[v_{k,l}[s']] = E[(v_{k,l} - v_{k,0})[s']] + E[v_{k,0}],
\]

where \(v_{k,0}\) is the value function at the start of the epoch. There are \(SA\) of these equations, one corresponding to each \((s,a) \in S \times A\) such that \(s' \sim p(\cdot|s,a)\). We estimate the mean on the left-hand-side (LHS) by the sum of estimates of means on the right-hand-side (RHS). Since \(\|v_{k,l} - v_{k,0}\|\) decreases rapidly with \(k\), because \(v_{k,l}\) and \(v_{k,0}\) rapidly approach \(v^*\), we ignore the first term on the RHS in our overview. We remark that its estimation cost affects the \(\epsilon\) range for which \texttt{SolveMdp1} is optimal. Consider the second term, \(E[v_{k,0}]\). This again needs to be estimated to error \(\epsilon/\Gamma\) which classically costs \(O(SA\Gamma^2/(\epsilon/\Gamma)^2) = O(SA\Gamma^4/\epsilon^2)\) by the same argument before. Quantumly, this costs \(O(SA\Gamma^2/\epsilon)\), again as before. Now, the key point is that we only need to estimate \(E[v_{k,0}]\) once per epoch and reuse its value throughout the epoch. As there are only logarithmically many epochs, the overall cost becomes \(\tilde{O}(SA\Gamma^4/\epsilon^2)\) classically and \(\tilde{O}(SA\Gamma^2/\epsilon)\) quantumly.

The total variance technique is more subtle. It is based on the observation that the actual error accumulation from iteration to iteration is much less than what is implicit above. To be clear, in the above, we set the error in mean estimation at each iteration to be \(\epsilon/\Gamma\) so that over \(\Gamma\) iterations, the accumulated error is \(\epsilon\). However, the error at each iteration \(i\) can actually be set larger, to \(\epsilon/\Gamma\) (which could be as large as \(\epsilon/\sqrt{\Gamma}\)), and it can still be shown that the overall accumulated error is \(\epsilon\) using properties of the standard deviation. More specifically, let us write \(\sigma_i = \sqrt{\text{Var}[v_i[s']]}\). Then, the cumulative standard deviation, \(\sum_{i=1}^{\Gamma} \sigma_i\), is closely related to an
expression for which we can non-trivially upper bound by $\sqrt{2} \Gamma^{1.5}$ (Theorem 1). Classically, it is straightforward to estimate $\mu_i := E[v_i[s]]$ to an error of $\epsilon \sigma_i/\Gamma^{1.5}$, without needing to know the $\sigma_i$s. This can be done using about $O((\epsilon/\Gamma^{1.5})^{-2}) = O(\Gamma^3/\epsilon^2)$ samples for each state-action pair as guaranteed by Chebyshev’s (or Bernstein’s) inequality. Combined with variance reduction, that is, applying the above technique to estimate the $E[v_{i,0}]$ from before, we see that this yields an overall classical sample complexity of $O(SAT^3/\epsilon^2)$. This is one main result of [SWW+18]. Due to the first term on the RHS of Eq. (3), which we glossed over, this result only holds for $\epsilon = O(1)$.

Trying to do a quantum version of the total variance technique poses a significant technical challenge for the following reasons. The version of quantum mean estimation that should have corresponded to a more efficient Chebyshev’s inequality, namely $q\text{Est}2$, is deficient compared to its classical counterpart in two ways. The first is that $q\text{Est2}$ cannot estimate $\mu_i$ to an error proportional to $\sigma_i$ without knowing $\sigma_i$ a priori. To remedy this, we first estimate $\sigma_i$ using $q\text{Est1}$ to some additive error $b > 0$. Denote the estimate by $\hat{\sigma}_i$. Then we can use $q\text{Est2}$ to estimate $\mu_i$ to error proportional to $\sigma_{\text{low}} := \hat{\sigma}_i - b$ ($\leq \sigma_i$) which maintains correctness. Unfortunately, this approach does not work due to the second deficiency of $q\text{Est2}$. In fact, $q\text{Est2}$ also requires an upper bound $C$ on $\sigma_i$ to function and uses $O(C/\epsilon)$ samples to guarantee additive error $\epsilon$. For large $C$, the sample complexity can be highly redundant with respect to the error guaranteed. This problem is directly relevant for us if we try to use $\sigma_{\text{high}} := \hat{\sigma}_i + b$ as $C$. Then, the complexity becomes proportional to $C/\sigma_{\text{low}} = (\hat{\sigma}_i + b)/(\hat{\sigma}_i - b)$, which can be arbitrarily large depending on the value of $\hat{\sigma}_i$ that we cannot control. To remedy this second problem, we in fact estimate $\mu_i$ to error proportional to $\sigma_{\text{high}}$, so that $C/\sigma_{\text{high}} = 1$ becomes constant. Of course, this no longer maintains correctness as $\sigma_{\text{high}}$ is larger than $\sigma_i$. However, we can bound $\sigma_{\text{high}} \leq \hat{\sigma}_i + 2b$. We then find, by performing a full correctness analysis, that the extra error of $2b$ can be sufficiently suppressed if we set $b$ and the parameter $c$ on Line 3 of $\text{SolveMdp1}$ to be small enough constants. Doing so only increases the overall complexity by a constant factor. Setting $b$ constant also ensures that the complexity of estimating $\sigma_i$ to error $b$ by $q\text{Est1}$ is within our budget. With the technical challenges resolved, we see that the complexity of $\text{SolveMdp1}$ is $O(SAT(\epsilon/\Gamma^{1.5})^{-1}) = O(SAT^{1.5}/\epsilon)$. Again, due to the first term on the RHS of Eq. (3), this only holds for $\epsilon = O(1/\sqrt{\Gamma})$. The $\epsilon$ range is smaller than before, which was $\epsilon = O(1)$, because there is relatively less quantum speedup for estimating that first term. (Note added: subsequently to the conference version of this work appearing [WSK+21], Hamoudi [Ham21, Theorem 13] removed the deficiencies of quantum mean estimation, as described in this paragraph, in general.)

In summary, we have described $\text{SolveMdp1}$, which uses $q\text{Est}$ to “quantize” all three techniques in [SWW+18]: monotonicity, variance reduction, and total variance. Quantizing the first two is not difficult but quantizing the last one offers a technical challenge. We believe that our solution to that challenge could find uses in quantizing other classical algorithms as well. We have also described $\text{SolveMdp2}$, which offers a quadratic speedup in $A$ using $q\text{Argmax}$. But because $q\text{Argmax}$ conflicts with the variance reduction and total variance techniques, $\text{SolveMdp2}$ no longer has optimal $\Gamma$ dependence.

Lower bound techniques. Lastly, we discuss how we prove our lower bounds. Standard techniques for proving lower bounds on the number of uses of a quantum oracle generally work with Boolean oracles. In our case, we instead have an oracle $G$ that outputs a particular quantum state for a given state-action pair which can also be invoked in superposition over state-action pairs. To enable the use of standard lower bound techniques from quantum query complexity, we reduce the problems of computing certain Boolean functions $f$ to our problems of computing $q^*, v^*$, and $\pi^*$ by instantiating our oracle $G$ using standard Boolean oracles. For example, consider a quantum oracle $G_{\text{coin}}$ that produces a state which represents a quantum sample of a coin toss with probability $p$ of getting heads. $G_{\text{coin}}$ can be instantiated by a Boolean oracle encoding a $n$-bit string (for large $n$).
which has \( p \) fraction of its bits equal to 1. The reduction then allows us to translate known lower bounds on computing \( f \) using a Boolean oracle to lower bounds on computing \( q^*, v^*, \) and \( \pi^* \) using oracle \( G \).

This approach has some unexpected benefits. Because we reduce to standard problems in query complexity, our proof is very modular. It allows us to also show optimal classical lower bounds by simply invoking the best classical lower bounds for the Boolean functions \( f \) mentioned above. Moreover, we qualitatively improve on known classical lower bounds. The known lower bound of [AMK12] shows that for any \( S, A \), there exists a hard MDP which has a number of state–action pairs equal to \( SA \). However, it is not the case that their constructed MDP has \( S \) states and \( A \) actions, just that the total number of state–action pairs is \( SA \). Their constructed MDP actually has \( O(SA) \) states, but most states only have \( O(1) \) actions, so the total number of state–action pairs is \( SA \). In contrast, our hard MDP instance genuinely has \( S \) states and \( A \) actions.

### 1.4 Related Work

As we have discussed, our quantum algorithms can be viewed as “quantizations” of the classical algorithms and techniques in [SWW+18, SWWY21, Wai19] which represent the latest development of classical model-free MDP solvers, which also recently include [Wan17, Wan20, JS20] among others, that started with [KS99]. [SWW+18] give algorithms with complexity \( \tilde{O}(SAT^3/\epsilon^2) \) when \( \epsilon = O(1) \) for approximating all three of \( q^* \), \( v^* \), and \( \pi^* \). On the model-free side, there has been even more recent progress culminating in the work of [LWC+20] which achieves \( \tilde{O}(SAT^3/\epsilon^2) \) for the full range of \( \epsilon \in (0, \Gamma] \). That this bound is tight (up to log-factors) is established by [AMK12] which is closely related to our work. Indeed, to prove our lower bounds, we use an instance inspired by [AMK12]. However, our proof by reduction and composition theorems is technically quite different from theirs and extends their lower bound to apply to arbitrary \( S \) and \( A \). Arguably, model-based MDP solvers [AMK12, AKY20, LWC+20] have seen more successes than their model-based counterparts that we quantized. However, quantizing these techniques appears more difficult. As a first step, one might ask if the quantum sample complexity of learning a probability distribution supported on \( n \) points to error \( \epsilon \) in \( \ell_1 \)-norm can be \( O(n/\epsilon) \), which represents a quadratic speedup over classical in terms of \( \epsilon \). Recently, this question has been answered affirmatively [vA21, CJ21] which immediately implies an \( \tilde{O}(S^2A\Gamma^2/\epsilon) \) model-based quantum algorithm for \( q^* \) due to [AJKW21, Proposition 2.1]. However, this complexity is highly suboptimal and it remains to be seen whether we could eventually obtain an optimal model-based quantum algorithm for any one of \( q^* \), \( v^* \), or \( \pi^* \).

On the quantum side, the broader subject of reinforcement learning “remains relatively unaddressed by the quantum community” [JTPN+21]. The relatively few works on the subject include [DCLT08, DTB16, PDM+14, DTB17, JTPN+21]. However, these works are incomparable to ours as they focus either on problem formulation or lack rigorous results. None give rigorous complexity bounds on computing \( \pi^* \), \( v^* \), and \( q^* \). Some of these works do mention the possibility of quadratic speedups by using quantum maximum finding [DTB16]. However, they do not consider how this technique would work within an integrated algorithm. As we have mentioned, our work shows that to achieve optimal \( \Gamma \)-dependence overall, we may have to forgo the use of quantum maximum finding. We note that in the multi-armed bandits setting, where \( S = 1 \), an instance-optimal quadratic quantum speedup is shown in [WYLC21]. In synergy with our work, [DTB17] proposes methods to instantiate the quantum generative model in real physical environments as opposed to being given a classical simulator. If their methods can be realized, our work will have wider applicability.
2 Preliminaries

2.1 Notation

For a positive integer $n$, we write $[n]$ for the set $\{1, \ldots, n\}$. We use upper case letters for matrices and lower case letters for vectors. For vectors only, we use square bracket notation $v[i]$ to mean entry $i$ of vector $v$. Vectors $v$ appearing in this work often have indices $i = (i_1, i_2)$ described by two coordinates in which case we write $v[i_1, i_2]$ to mean $v[(i_1, i_2)]$. As a function $v : X \to Y$ can be identified with the corresponding vector $v \in Y^X$, we also use square bracket notation to index into functions. For any two real vectors $u, v$ of the same dimension, we write $\max\{u, v\}$ to mean the element-wise max of $u$ and $v$ and $u \leq v$ to mean the inequality holds element-wise. We write bold 1 (resp. 0) for a vector of all 1s (resp 0s) with dimension determined by context. A scalar $x \in \mathbb{R}$ appearing alone in an equation involving vectors is to be interpreted as $x \cdot 1$. For a function $f : A \to B$ and vector $v$ with entries in $A$, we write $f(v)$ for the vector with entries in $B$ resulting from applying $f$ to $v$ element-wise. For a set $X$, we often identify $X^S$ with $X^S, X^A$ with $X^A, X^{S \times A}$ with $X^{S \times A}$, and so on.

2.2 MDP Preliminaries

For a policy $\pi$, we define $P^\pi \in \mathbb{R}^{SA \times SA}$ to be the matrix with entries

$$
P^\pi_{(s,a),(s',a')} = \begin{cases} 
p(s'|s,a) & \text{if } a' = \pi(s'), \\ 0 & \text{otherwise.} \end{cases} \tag{4}
$$

We define $P \in \mathbb{R}^{SA \times S}$ to be the matrix with entries $P_{(s,a),s'} = p(s'|s,a)$ and, for fixed $(s,a) \in S \times A$, we define $p_{s,a} \in \mathbb{R}^S$ to be the vector with entries $p_{s,a}[s'] = p(s'|s,a)$. The preceding definitions mean that, for any $u \in \mathbb{R}^S$, we have $(Pu)[s,a] = p_{s,a} u$.

For $u \in \mathbb{R}^S$, we define $\sigma^2(u) \in \mathbb{R}^S$ to be a vector with entries $\sigma^2(u)[s,a] := \text{Var}[u[s'] | s' \sim p(\cdot | s,a)]$. Note that this means $\sigma^2(u) = Pu^2 - (Pu)^2$. Naturally, we write $\sigma(u) := \sqrt{\sigma^2(u)}$.

We define the value operator of policy $\pi$, $T^\pi : \mathbb{R}^S \to \mathbb{R}^S$, by its mapping of $u \in \mathbb{R}^S$, defined entry-wise by

$$
T^\pi(u)[s] := r(s, \pi[s]) + \gamma P^\pi_{s,\pi[s]} u. \tag{5}
$$

It can be readily verified that $T^\pi$ (for any $\pi$) is monotonically increasing with respect to the element-wise order ($\leq$) on $\mathbb{R}^S$, is a $\gamma$-contraction with respect to the $l_\infty$-norm on $\mathbb{R}^S$, and has unique fixed point $v^\pi$.

For a vector $q \in \mathbb{R}^SA$, we also define $v(q) \in \mathbb{R}^S$ and $\pi(q) \in \mathcal{A}^S$ by $v(q)[s] = \max_a \{q[s,a]\}$ and $\pi(q)[s] = \text{argmax}_a \{q[s,a]\}$ respectively. Note that this means $v(q)[s] = q[s, \pi(q)[s]]$.

Finally, for the total-variance technique, we will also need:

**Theorem 1.** [AJKW21, AMK12] For any policy $\pi$, we have

$$
\| (I - \gamma P^\pi)^{-1} \sigma(v^\pi) \| \leq \sqrt{2/\Gamma^{1.5}}. \tag{6}
$$

2.3 Quantum Preliminaries

We now describe quantum oracles in more detail using standard quantum notation (Dirac notation). We briefly review this notation so that the following definitions make sense and refer readers to [NC00] for more information.
In Dirac notation, vectors $v \in \mathbb{C}^n$ are written as as $|v\rangle$, and called “ket $v$”. The notation $|i\rangle$, with $i \in [n]$, is reserved for the $i$-th standard basis vector. $|0\rangle$ is also reserved for the 1st standard basis vector when there is no conflict. A ket $|i_1i_2\ldots i_M\rangle$ with $i_j \in \{0,1\}$ is interpreted as the vector $|i+1\rangle \in \mathbb{C}^{2^M}$, where $i$ is the integer that is represented by $i_1\ldots i_M$ in binary.

**Definition 1** (Quantum oracle encoding of functions and vectors). Let $\Omega$ be a finite set of size $n$ and $u \in \mathbb{R}^\Omega$ (equivalently, $u : \Omega \to \mathbb{R}$) where, for all $i \in \Omega$, $u_i$ is represented by an $M$-bit string $\bar{u}_i$. A quantum oracle encoding $u$ is a unitary matrix $U_u : \mathbb{C}^n \otimes \mathbb{C}^{2^M} \to \mathbb{C}^n \otimes \mathbb{C}^{2^M}$ such that $U_u : |i\rangle \otimes |0\rangle \mapsto |i\rangle \otimes |\bar{u}_i\rangle$ for all $i \in [n]$.

Like in the classical setting, we may always assume that $M$ is sufficiently large for our purposes.

**Definition 2** (Quantum oracle encoding of probability distributions). Let $\Omega$ be a finite set of size $n$ and $p = (p_x)_{x \in \Omega}$ a discrete probability distribution on $\Omega$. The quantum oracle encoding of $p$ is a unitary matrix $U_p : \mathbb{C}^n \otimes \mathbb{C}^J \to \mathbb{C}^n \otimes \mathbb{C}^J$ such that $U_p : |0\rangle \otimes |0\rangle = \sum_{x \in \Omega} \sqrt{p_x} |x\rangle \otimes |v_x\rangle$, where $0 \leq J \in \mathbb{Z}$ is arbitrary and $|v_x\rangle \in \mathbb{C}^J$ is arbitrary.

**Definition 3** (Quantum generative model of an MDP). The quantum generative model of an MDP, with transition probabilities $p(s'|s,a)$, is a unitary matrix $G : \mathbb{C}^S \otimes \mathbb{C}^A \otimes \mathbb{C}^S \otimes \mathbb{C}^J \to \mathbb{C}^S \otimes \mathbb{C}^A \otimes \mathbb{C}^S \otimes \mathbb{C}^J$ such that

$$G : |s\rangle \otimes |a\rangle \otimes |0\rangle \otimes |0\rangle$$

$$\mapsto |s\rangle \otimes |a\rangle \otimes \left( \sum_{s' \in S} \sqrt{p(s'|s,a)} |s'\rangle \otimes |\psi_{s',s,a}\rangle \right),$$

where $0 \leq J \in \mathbb{Z}$ is arbitrary and $|\psi_{s',s,a}\rangle \in \mathbb{C}^J$ is arbitrary.

We stress that the quantum state output by $G$ in Eq. (7) is analogous to a sample drawn from the classical probability distribution $\{p(s'|s,a)\}_{s' \in S}$ as opposed to that distribution fully written out on a piece of paper. In Appendix A, we describe how to systematically and efficiently construct the quantum generative model from a circuit for a classical generative model. This construction is already implicit in, for example, [Mon15, HM19, Bel19], but we provide a description for completeness.

### 3 Analysis of Quantum Algorithms

In this section, we formally analyze our two algorithms $\text{SolveMdp1}$ and $\text{SolveMdp2}$. These algorithms make essential use of two quantum subroutines: quantum mean estimation and quantum maximum finding. We begin by specifying the performance guarantees of these subroutines.

#### 3.1 Quantum Mean Estimation and Maximum Finding

**Theorem 2** (Quantum mean estimation [BHMT02, Mon15]). There are two quantum algorithms $\text{qEst1}$ and $\text{qEst2}$ with the following specifications. Let $\Omega$ be a finite set, $p = (p_x)_{x \in \Omega}$ a discrete probability distribution on $\Omega$, and function $v : \Omega \to \mathbb{R}$. Given quantum oracles $U_p$ and $U_v$ encoding $p$ and $v$ respectively. Then,

1. $\text{qEst1}$ requires $u, \epsilon > 0$ as additional inputs and a promise $0 \leq v \leq u$, in which case $\text{qEst1}$ uses $O(u/\epsilon + \sqrt{u/\epsilon})$ queries to $U_p$, alternatively

2. $\text{qEst2}$ requires $\sigma > 0$ and $\epsilon \in (0, 4\sigma)$ as additional inputs and a promise $\text{Var}[v(x) | x \sim p] \leq \sigma^2$, in which case $\text{qEst2}$ uses $O((\sigma/\epsilon) \log^2(\sigma/\epsilon))$ queries to $U_p$
to output an estimate $\hat{\mu}'$ of $\mu := \mathbb{E}[v[x] | x \sim p] = p^T v$ with $\Pr(|\hat{\mu}' - \mu| > \epsilon) < 1/3$. Moreover, by repeating one of $q\text{Est}1$ or $q\text{Est}2$ $O(\log(1/\delta))$ times and taking the median output yields an estimate $\hat{\mu}$ of $\mu$ with $\Pr(|\hat{\mu} - \mu| < \epsilon) > 1 - \delta$.

For $i \in \{1, 2\}$, we write $q\text{Est}_i \{\delta \} (p^T v, \epsilon)$ for an estimate of the mean of $v[x]$, with $x$ distributed as $p$, to error $< \epsilon$ with probability $> 1 - \delta$, using $q\text{Est}$. The median-of-means part of Theorem 2 is sometimes referred to as the “powering lemma” [JVV86].

**Theorem 3** (Quantum maximum finding [DH96]). There exists a universal constant $c_{\max} > 0$ such that the following holds. There is a quantum algorithm $q\text{Argmax}$ such that, given a quantum oracle $U_u$ encoding a vector $u \in \mathbb{R}^n$, $A_{\max}$ at most $c_{\max} \sqrt{n} \log(1/\delta)$ queries to $U_u$ and finds $\text{argmax}_i (u_i)$ with probability $> 1 - \delta$.

We write $q\text{Argmax}_\delta \{u[i] : i \in [n]\}$ for an estimate of the maximum of $u$, with probability $> 1 - \delta$, using $q\text{Argmax}$.

### 3.2 Analysis of SolveMdp1

We will use the following lemma which clearly follows from the if-then-else statement appearing in SolveMdp1.

**Lemma 1.** For all $k \in [K]$ and $l \in \{0\} \cup [L]$, the $v_{k,l}$s are monotone increasing with respect to $(k - 1)L + l$. Moreover, for all $k \in [K]$ and $l \in [L]$, we have $v_{k,l} \geq v_{(q_{k,l} - 1)}$.

Using Lemma 1 and the fact that our mean estimates are always shifted down to have one-sided error, we can prove the following proposition similarly to [SWW+18, Section E of arXiv version]; the key point is to show that $v_{k,l} \leq T_{v_{k,l}}(v_{k,l})$. We present the full details for completeness.

**Proposition 1.** For all $k \in [K]$ and $l \in [L]$, we have

\[
\begin{align*}
    v_{k,l} &\leq v^{\pi_{k,l}} \leq v^*, \\
    q_{k,l} &\leq q^{\pi_{k,l}} \leq q^*,
\end{align*}
\]

with probability at least $1 - \delta$.

**Proof.** We first consider the failure probability. As all estimations are carried out with maximum failure probability $f := \delta/4KLSA$ and there are $3KSA + KLSA < 4KLSA$ estimations (Lines 7, 8 and 12), the probability that there exists an incorrect estimate (up to the specified error) is at most $\delta$ by the union bound.

We henceforth assume the $q\text{Est}$ steps are all correct and proceed to prove Eq. (8) and Eq. (9).

The second inequalities in Eq. (8) and Eq. (9) are clear from the definitions of $v^*$ and $q^*$. We therefore only show the first inequalities below and refer to them when referring to Eq. (8) and Eq. (9). The main idea is to use Lemma 1 together with the inequalities

\[
\begin{align*}
    x_k &\leq P v_{k,0}, \\
    \Delta_{k,l} &\leq P v_{k,l} - P v_{k,0},
\end{align*}
\]

that are immediate from the definitions of $x_k$ and $\Delta_{k,l}$ on Lines 8 and 12 respectively because the subtracted terms equal the estimation errors.

To show Eq. (8), it suffices to show

\[
v_{k,l} \leq T_{v_{k,l}}(v_{k,l}).
\]
Equation (8) then follows from repeatedly applying $T^{π_k,t}$ on both sides of Eq. (12), and using the fact that $T^{π_k,t}$ is monotone increasing and is a contraction with unique fixed point $v^{π_k,t}$.

We proceed to show Eq. (12) by induction on $n := (k - 1)L + l$. The base case $n = 0$ is true because $v_{1,0} := 0 \leq T^{π_{1,0}}(v_{1,0}) = r$. The case $n = 1$ is also true because $v_{1,1} = v(q_{1,0}) = 0 \leq T^{π_{1,1}}(v_{1,1}) = r$, where we used $q_{1,0} := 0$. In addition, note that $v_{k,L} \leq T^{π_k,L}(v_{k,L})$ is the same as $v_{k+1,0} \leq T^{π_{k+1,0}}(v_{k+1,0})$ by definitions on Line 15. This means that once we have established the truth of Eq. (12) at $k = k', l = L$, we can assume its truth at $k = k' + 1, l = 0$.

Now consider $n > 1$. We prove Eq. (12) element-wise for each $s \in S$ by considering the following two cases that could happen at the if-clause on Line 10.

1. Case $v(q_{k,l-1})[s] \geq v_{k,l-1}[s]$. Then

$$v_{k,l}[s] := v(q_{k,l-1})[s]$$

$$= q_{k,l-1}[s, π_{k,l}[s]]$$

$$= \max\{r[s, π_{k,l}(s)] + ϵ(x_k[s, π_{k,l}[s]] + Δ_k,l-1[s, π_{k,l}[s]], 0)\}$$

$$\leq r[s, π_{k,l}(s)] + ϵ(Pv_{k,l-1}[s, π_{k,l}(s)])$$

$$= T^{π_k,l}(v_{k,l-1})[s]$$

$$\leq T^{π_k,l}(v_{k,l})[s],$$

where the second line uses $π_{k,l}[s] := π(q_{k,l-1})[s]$ in this case, the third line uses definition of $q_{k,l-1}$ (for $n > 1$), the fourth line uses Eq. (10) and Eq. (11) and $0 \leq v_{k,l-1}$ (Lemma 1) to remove the max, and the last line uses $v_{k,l-1} \leq v_{k,l}$ (Lemma 1).

2. Case $v(q_{k,l-1})[s] < v_{k,l-1}[s]$. Then

$$v_{k,l}[s] := v(q_{k,l-1})[s] \leq T^{π_{k,l-1}}(v_{k,l-1})[s] \leq T^{π_{k,l-1}}(v_{k,l})[s] = T^{π_k,l}(v_{k,l})[s],$$

where the first inequality is by the inductive hypothesis, the second inequality uses $v_{k,l-1} \leq v_{k,l}$ (Lemma 1), and the last equality uses $π_{k,l}[s] := π_{k,l-1}[s]$ in this case.

Therefore, we have established Eq. (12), and so Eq. (8).

Equation (9) then follows from

$$q_{k,l} \leq r + ϵPv_{k,l} \leq r + q^*Pv_{k,l} = q^*_{π_k,l},$$

where the first inequality again uses Eq. (10) and Eq. (11) and $0 \leq v_{k,l}$ (Lemma 1), and the second inequality uses Eq. (8) which we have just established. \hfill \Box

The above proposition shows that $v_{k,L}$ and $q_{k,L}$ are upper bounded by $v^*$ and $q^*$ respectively. Therefore, the following proposition shows that $v_{k,L}$ and $q_{k,L}$ are converging to $v^*$ and $q^*$ respectively.

**Proposition 2.** For all $k ∈ [K]$, we have

$$v^* - ε_k \leq v_{k,L},$$

$$q^* - ε_k \leq q_{k,L},$$

with probability at least $1 - δ$.

If there were no mean estimation errors, Proposition 2 follows from the contractive properties of the Bellman operator. The challenge for us is to analyze those errors carefully. As we mentioned in our Introduction, the errors involved here go beyond those analyzed in [SWW+18].

12
Proof. By reusing the first paragraph in the proof of Proposition 1, we can readily set aside consideration of the failure probability. We henceforth assume the qEst steps are all correct and proceed to prove Eq. (16) and Eq. (17).

We proceed by induction on \( k \geq 0 \) with the inductive hypothesis comprising both inequalities above for all indices strictly less than \( k \). The base case \( k = 0 \) can be established by defining \( \epsilon_0 := \Gamma \), \( v_{0,L} := 0 \), and \( q_{0,L} := 0 \). Note that these definitions are consistent with the induction steps below.

Now consider \( k > 0 \). The main idea is to use Theorem 1 and the inequalities

\[
\begin{align*}
    x_k &\geq P v_{k,0} - 2c(1 - \gamma)^{1.5} \epsilon \sqrt{y_k + b}, \\
    \Delta_{k,l} &\geq P v_{k,l} - P v_{k,0} - 2c(1 - \gamma) \epsilon_k,
\end{align*}
\]

that are immediate from the definitions of \( x_k \) and \( \Delta_{k,l} \) on Lines 8 and 12 respectively.

We first show Eq. (17). Define vector \( \xi_k \in \mathbb{R}^{S_A} \) by

\[
\xi_k := 2c(1 - \gamma)^{1.5} \epsilon \sqrt{y_k + b} + 2c(1 - \gamma) \epsilon_k,
\]

then we have

\[
q^* - q_{k,l} = r + \gamma P^{\pi^*} q^* - \max\{r + \gamma(x_k + \Delta_{k,l}), 0\} \\
\leq \gamma P^{\pi^*} q^* - \gamma(x_k + \Delta_{k,l}) \\
\leq \gamma P^{\pi^*} q^* - \gamma(P v_{k,0} + P v_{k,l} - P v_{k,0} - 2c(1 - \gamma)^{1.5} \epsilon \sqrt{y_k + b} - 2c(1 - \gamma) \epsilon_k) \\
\leq \gamma P^{\pi^*} q^* - \gamma P v_{k,l} + 2c(1 - \gamma)^{1.5} \epsilon \sqrt{y_k + b} + 2c(1 - \gamma) \epsilon_k
\]

(21)

where the fourth line uses \( \gamma \leq 1 \), the sixth line uses \( v(q_{k,l-1}) \leq v_{k,l} \) (Lemma 1), and the last line uses \( P^{\pi^*} q_{k,l-1} \leq P v(q_{k,l-1}) \) which follows from definitions.

Recursing Eq. (21) with respect to \( l \geq 1 \) gives

\[
q^* - q_{k,l} \leq \gamma^l (P^{\pi^*})^l (q^* - q_{k,0}) + \sum_{i=0}^{l-1} \gamma^i (P^{\pi^*})^i \xi_k
\]

(22)

\[
\leq \gamma^l \Gamma + (I - \gamma P^{\pi^*})^{-1} \xi_k,
\]

where the last line uses \( q^* - q_{k,0} \leq q^* \leq \Gamma \) as \( q_{k,0} \geq 0 \) by definitions on Line 4 and Line 13. The first term, \( \gamma^l \Gamma \), can be bounded when \( l = L - 1, L \):

\[
\gamma^l \Gamma \leq \gamma^{L-1} \Gamma \leq \exp(-(L - 1)(1 - \gamma)) \Gamma \leq \epsilon/4 \leq \epsilon_k/2,
\]

(23)

where the second inequality uses \( x \leq \exp(-(1 - x)) \) for all \( x \in \mathbb{R} \), the third inequality uses the definition \( L := \Gamma \lceil \log(4\Gamma/\epsilon) \rceil + 1 \), and the last inequality uses \( \epsilon \leq 2\epsilon_k \leq 2\epsilon_k \) for all \( k \in [K] \) which follows from \( K \leq \log_2(\Gamma/\epsilon) + 1 \).

We now bound the second term, \( (I - \gamma P^{\pi^*})^{-1} \xi_k \). To this end, we first bound the term \( \sqrt{y_k + b} \).
appearing in $\xi_k$. From the definition of $y_k$, there exists a $b'$ with $|b'| \leq b$ such that

$$\sqrt{y_k + b} \leq \max \{(Pv_{k,0}^2 + b - (Pv_{k,0} + (1 - \gamma)b')^2)^{1/2}, \sqrt{b}\} \leq (\sigma^2(v_{k,0}) + b + 2(1 - \gamma)|b'|Pv_{k,0})^{1/2} \leq \sqrt{\sigma^2(v_{k,0}) + 3b} \leq \sigma(v_{k,0}) + \sqrt{3b} \leq \sigma(v^*) + \sigma(v^* - v_{k,0}) + \sqrt{3b}, \quad (24)$$

where the second line uses $0 \leq v_{k,0}$ (Lemma 1) to remove the max, the third line uses the fact that, for any random variables $X$ and $Y$, we have $\text{Var}[X + Y] = \text{Var}[X] + \text{Var}[Y] + 2\text{Cov}[X, Y] \leq (\sqrt{\text{Var}[X]} + \sqrt{\text{Var}[Y]})^2$.

But we have $v_{k,0} - v^* \leq 0$ from Eq. (8) of Proposition 1 and $v^* - v_{k,0} = v^* - v_{k-1,L} \leq \epsilon_{k-1}$ by the inductive hypothesis. Therefore, $\sigma(v_{k,0} - v^*) \leq \|v_{k,0} - v^*\| \leq \epsilon_{k-1} = 2\epsilon_k$, and therefore

$$\sqrt{y_k + b} \leq \sigma(v^*) + 2\epsilon_k + \sqrt{3b}. \quad (25)$$

Therefore, recalling $\xi_k := 2c(1 - \gamma)^{1.5}e\sqrt{y_k + b} + 2c(1 - \gamma)\epsilon_k$ from Eq. (20), we have

$$(I - \gamma P^{\pi^*})^{-1}\xi_k = 2c(1 - \gamma)^{1.5}e(I - \gamma P^{\pi^*})^{-1}\sqrt{y_k + b} + 2c(1 - \gamma)\epsilon_k(I - \gamma P^{\pi^*})^{-1}1$$

$$\leq 2c(1 - \gamma)^{1.5}e(I - \gamma P^{\pi^*})^{-1}(\sigma(v^*) + 2\epsilon_k + \sqrt{3b}) + 2c(1 - \gamma)\epsilon_k(I - \gamma P^{\pi^*})^{-1}1 \leq 2c(1 - \gamma)^{1.5}e(I - \gamma P^{\pi^*})^{-1}\epsilon_k + 2c\sqrt{1 - \gamma} \leq 2c(\sqrt{2} + 2\sqrt{1 - \gamma}) \epsilon_k \leq 2c(\sqrt{2} + 2\sqrt{3b} + 1) \epsilon_k < \epsilon_k/2, \quad (26)$$

where the third line uses $(I - \gamma P^{\pi^*})^{-1}1 \leq (1 - \gamma)^{-1}$, the fourth line crucially uses Theorem 1 with $\pi$ set to $\pi^*$, the fifth line uses $e \leq \epsilon_k$ for all $k \in [K]$ and the input assumption $\sqrt{1 - \gamma} \leq 1$, i.e., $\epsilon \leq \sqrt{1}$, and the last line uses definitions $b := 1$ and $c := 0.01$.

Using Eq. (23) and Eq. (26) to bound the first and second terms in Eq. (22) respectively, we find

$$q^* - q_{k,L} \leq \epsilon_k, \quad (27)$$

$$q^* - q_{k,L-1} \leq \epsilon_k. \quad (28)$$

The top equation is one inequality we wish to show in our induction. The bottom equation can be used to establish the other inequality as follows. For all $s \in S$, we have

$$v_{k,L}[s] \geq v(q_{k,L-1}[s]) = \max_a\{q_{k,L-1}[s, a]\} \geq \max_a\{q^*[s, a] - \epsilon_k\} = v^*[s] - \epsilon_k, \quad (29)$$

where the first inequality is by Lemma 1. Hence $v_{k,L} \geq v^* - \epsilon_k$, as desired.

The correctness of Algorithm 1 then follows from combining Proposition 1 and Proposition 2 with $k = K$ and $l = L$ and recalling the definitions of $(\hat{v}, \hat{\pi}, \hat{q})$ and $K$. Formally:

**Theorem 4** (Correctness of SolveMdp1). The outputs $\hat{v}$, $\hat{\pi}$, and $\hat{q}$ of SolveMdp1 satisfy

$$v^* - \epsilon \leq \hat{v} \leq v^*, \quad (30)$$

$$q^* - \epsilon \leq \hat{q} \leq q^*, \quad (31)$$

with probability at least $1 - \delta$. 

---

14
Having shown correctness, we turn to complexity:

**Theorem 5 (Complexity of SolveMdp1).** The quantum query complexity of SolveMdp1 is

\[ O(SA(\Gamma^{1.5} \epsilon^{-1} + \Gamma^2) \log^4(\Gamma/\epsilon) \log(SA\Gamma/\delta)). \]  

(32)

The proof of Theorem 5 involves showing Theorem 2 is applicable and applying it.

**Proof.** As in the correctness analysis, we assume that all estimations are correct, up to the specified error, because the probability that this does not hold is at most \( \delta \). This means we can assume all results obtained during the correctness analysis. In the following, we will use \( K = O(\log(\Gamma/\epsilon)) \) and \( L = O(\Gamma \log(\Gamma/\epsilon)) \) without further remarks.

Let \( C \) be the complexity of SolveMdp1 as if all estimations were carried out with maximum failure probabilities set to constant. Then, since the actual maximum failure probabilities are set to \( f := \delta/4KLSA \), the actual complexity of SolveMdp1 is

\[ O(C \log(KLSA/\delta)) = O(C \log(SA\Gamma \log(\Gamma/\epsilon)/\delta)). \]  

(33)

Now we bound \( C \) by examining each line involving \( q\text{Est} \) in turn and using Theorem 2.

On Line 7, we can bound \( 0 \leq v_{k,0} \leq v^* \leq \Gamma \). Therefore, we can use quantum mean estimation algorithm \( q\text{Est}1 \) in Theorem 2, which results in an overall query cost of order

\[ SAK(\Gamma^2 b^{-1} + \sqrt{\Gamma^2 b^{-1}} + \Gamma(1-\gamma)^{-1}b^{-1} + \sqrt{\Gamma(1-\gamma)^{-1}b^{-1}}) = O(SA\Gamma^2 \log(\Gamma/\epsilon)). \]  

(34)

On Line 8, we see that \( \sigma^2 (v_{k,0})[s,a] \leq \sqrt{y_k[s,a] + b} \). We also note that \( 0 < (1-\gamma)^{1.5} \sqrt{y_k[s,a] + b} < 4\sqrt{y_k[s,a] + b} \). Therefore, we can use quantum mean estimation algorithm \( q\text{Est}2 \) in Theorem 2, with error set to \( (1-\gamma)^{1.5} \sqrt{y_k[s,a] + b} \) and variance upper bound set to \( y_k[s,a] + b \), which results in an overall query cost of order

\[ K \sum_{(s,a) \in S \times A} w[s,a] \log^2(w[s,a]) = O(SA\Gamma^{1.5} \epsilon^{-1} \log^3(\Gamma/\epsilon)), \]  

(35)

where, importantly, \( w[s,a] := (\sqrt{y_k[s,a] + b})(1-\gamma)^{1.5} \sqrt{y_k[s,a] + b}^{-1} = \Gamma^{1.5}/\epsilon. \)

On Line 12, we can bound \( 0 \leq v_{k,l} - v_{k,0} \leq v^* - v_{k,0} \leq \epsilon_{k-1} = 2\epsilon_k \). Therefore, we can use quantum mean estimation algorithm \( q\text{Est}1 \) in Theorem 2, which results in an overall cost of order

\[ LSA \left( \frac{2\epsilon_k}{c(1-\gamma)\epsilon_k} + \sqrt{\frac{2\epsilon_k}{c(1-\gamma)\epsilon_k}} \right) = O(SA\Gamma^2 \log(\Gamma/\epsilon)). \]  

(36)

Adding together Eq. (34), Eq. (35), and Eq. (36), and noting that all logarithmic terms are at most \( \log^3(\Gamma/\epsilon) \), shows that

\[ C = O(SA(\Gamma^{1.5} \epsilon^{-1} + \Gamma^2) \log^3(\Gamma/\epsilon)). \]  

(37)

Combining the above equation with Eq. (33) shows that the overall quantum query complexity of SolveMdp1 is

\[ O(SA(\Gamma^{1.5} \epsilon^{-1} + \Gamma^2) \log^3(\Gamma/\epsilon) \log(SA\Gamma \log(\Gamma/\epsilon)/\delta)) = O(SA(\Gamma^{1.5} \epsilon^{-1} + \Gamma^2) \log^4(\Gamma/\epsilon) \log(SA\Gamma/\delta)), \]  

(38)

as desired. \( \square \)
3.3 Analysis of SolveMdp2

If we only require $v^*$ and $\pi^*$ but not $q^*$, then we present an alternative quantum algorithm we call SolveMdp2 that is quadratically faster than SolveMdp1 in terms of $A$. The source of the speedup in $A$ is our use of quantum maximum finding to find the maximum at each iteration $l$, $\text{qArgmax}_f\{q_{l-1,s}[a] : a \in A\}$ on Line 6 which uses the quantum oracle encoding $U_{q_{l-1,s}}$ created in the previous iteration $l - 1$.

Failure probability aside, our strategy for proving the correctness of SolveMdp2 (Theorem 6) is to observe the similarity between SolveMdp2 and SolveMdp1 and then reuse the arguments used to prove the correctness of SolveMdp1.

SolveMdp2 is similar to SolveMdp1 with $k$ set to 1. In particular, the vectors $z_l$, $q_l \in \mathbb{R}^{SA}$, defined entry-wise by

$$z_l[s,a] := z_{l,s}[a], \quad (39)$$

$$q_l[s,a] := q_{l,s}[a], \quad (40)$$

are analogous to the vectors $x_1 + \Delta_{1,t}$ and $q_{1,t}$ appearing in SolveMdp1 respectively. Moreover, the $\tilde{v}_l[s] := \max_a\{q_{l-1,s}[a]\} = \max_a\{q_{l-1}[s,a]\}$ appearing in SolveMdp2 corresponds exactly to the $v(q_{l-1})[s] := \max_a\{q_{l-1}[s,a]\}$ appearing in SolveMdp1.

Having observed the similarity between SolveMdp2 and SolveMdp1, the following analogue of Lemma 1 due to the if-then-else statement is clear.

**Lemma 2.** For all $l \in [L]$, the $v_l$s are monotone increasing, that is $v_{l-1} \leq v_l$, and moreover we have $v_l \geq v(q_{l-1})$.

We now proceed to establish the correctness and complexity of SolveMdp2. In the proof of correctness, we will reuse the proofs of Proposition 1 and Proposition 2 unchanged except that they now invoke Lemma 2 instead of Lemma 1.

**Theorem 6 (Correctness of SolveMdp2.)** The outputs $\hat{v}$ and $\hat{\pi}$ of SolveMdp2 satisfy

$$v^* - \epsilon \leq \hat{v} \leq v^* \hat{\pi},$$

with probability at least $1 - \delta$.

**Proof.** We first consider the failure probability. The analysis is similar to that used to prove Proposition 1 except that we now need to analyze quantum oracles that may fail. To do this, we appeal to basic facts about unitary matrices, in particular, a quantum version of the union bound stating that the failure probabilities of quantum operators, i.e., unitary matrices, add linearly. On Line 10, because $U_{z_{l,s}}$ is created using qEst with failure probability $f$, it is $2Af$-close to its “ideal version”. More precisely, we mean that there exists a quantum oracle $U^{\text{ideal}}_{z_{l,s}}$ encoding $(\hat{P}v_l)[s,a] - (1 - \gamma)\epsilon/4$, where $(\hat{P}v_l)[s,a]$ satisfies $|(\hat{P}v_l)[s,a] - (Pv_l)[s,a]| \leq (1 - \gamma)\epsilon/4$, such that $\|U^{\text{ideal}}_{z_{l,s}} - U_{z_{l,s}}\|_{\text{op}} \leq 2Af$. Since $U_{q_l,s}$ can be created using one call to $U_{z_{l,s}}$ and one call to $U_{z_{l,s}}^{-1}$, it is $4Af$-close to its ideal version (defined similarly). Then, on Line 6, qArgmax uses the oracle $U_{q_l,s}$ at most $c_{\text{max}}\sqrt{A}\log(1/\delta)$ times. By the quantum union bound and substituting in the definition of $f$, this means the quantum operation implemented by qArgmax is $(c_{\text{max}}\sqrt{A}\log(1/\delta) \cdot 4Af = \delta/LS)$-close to its ideal version. This means that the output of qArgmax is incorrect with probability at most $\delta/LS$. Since qArgmax is invoked a total of $LS$ times, we see that the overall probability of failure is at most $\delta$ by the (usual) union bound.

We henceforth assume the qEst and qArgmax steps are all correct and proceed to prove Eq. (41).
The last inequality, \( v^* \leq v^* \), is clear.

To prove the middle inequality, \( \hat{v} \leq v^* \), we can directly reuse the proof of Proposition 1 provided we have \( z_l \leq Pv_l \). But this is clear because \( x_l \) is equal to an estimate of \( Pv_l \) with the estimation error subtracted off.

To prove the first inequality, \( v^* - \epsilon \leq \hat{v} \), we can reuse the proof of Proposition 2, provided we have \( z_l \geq Pv_l - (1 - \gamma)\epsilon/2 \), which is true. Defining \( \xi := (1 - \gamma)\epsilon/2 \cdot 1 \in \mathbb{R}^{SA} \), we see from the proof of Proposition 2 that

\[
q^* - q_{L-1} \leq \gamma^{L-1}\Gamma + (1 - \gamma P_{q^*})^{-1}\xi \leq \epsilon,
\]

since \( L := \Gamma[\log(4\Gamma/\epsilon)] + 1 \). Therefore, for all \( s \in S \), we have

\[
v_L[s] \geq v(q_{L-1})[s] = \max_a\{q_{L-1}[s, a] \geq \max_a\{q^*[s, a] - \epsilon\} = v^*[s] - \epsilon.
\]

\( \blacksquare \)

**Theorem 7 (Complexity of SolveMdp2).** The quantum query complexity of \( \text{SolveMdp2} \) is

\[
O(S\sqrt{A}\Gamma^3 \epsilon^{-1} \log^2(\Gamma/\epsilon) \log(SA\Gamma/\delta)).
\]

**Proof.** We can assume all results obtained during the correctness proof of \( \text{SolveMdp2} \).

We let \( C \) be the complexity of \( \text{SolveMdp2} \) as if all estimations and maximum finding were carried out with maximum failure probabilities set to constant. Then the actual complexity of our algorithm is

\[
O(C \log(LSA/\delta)) = O(C \log(SA\Gamma \log(\Gamma/\epsilon)/\delta)),
\]

since the actual maximum failure probabilities are set to \( f := \delta/4c_{\max}LSA^{1.5}\log(1/\delta) \) and \( L = O(\Gamma \log(\Gamma/\epsilon)) \).

Now we bound \( C \). Note that, for all \( l \in [L] \), we have

\[
0 \leq v_l \leq v^* \leq \Gamma.
\]

By using \( \text{qEst1} \) of Theorem 2 to do the \( \text{qEst} \) on Line 10, the query complexity of \( U_{z_l,s} \) is

\[
\frac{\Gamma}{(1 - \gamma)\epsilon/4} + \sqrt{\frac{\Gamma}{(1 - \gamma)^2/4}} = O(\Gamma^2/\epsilon),
\]

provided \( \epsilon = O(\Gamma^2) \). But we have (trivially) assumed \( \epsilon \leq \Gamma \) on the input \( \epsilon \), so this holds.

As \( U_{q_l,s} \) uses one call to \( U_{z_l,s} \) and one call to its inverse \( U_{z_l,s}^{-1} \), the query complexity of \( U_{q_l,s} \) is twice that of \( U_{z_l,s} \).

By means of the quantum maximum finding algorithm (Theorem 3) we only incur a multiplicative factor of \( O(\sqrt{A}) \) when we invoke \( \text{qArgmax} \) over an action space of size \( A \). That is, for each \( l \in [L] \) and \( s \in S \), \( \text{qArgmax} \) makes \( O(\sqrt{A}) \) queries to \( U_{q_l,s} \) to find \( \text{argmax}_a \{q_{l-1,s}[a]\} \). There are also \( L \) iterations, so

\[
C = O(LS\sqrt{A}\Gamma^2 \epsilon^{-1}) = O(S\sqrt{A}\Gamma^3 \epsilon^{-1} \log(\Gamma/\epsilon)),
\]

because \( L = O(\Gamma \log(\Gamma/\epsilon)) \). Combining the above equation with Eq. (45) shows that the overall quantum query complexity of \( \text{SolveMdp2} \) is

\[
O(S\sqrt{A}\Gamma^3 \epsilon^{-1} \log(\Gamma/\epsilon) \log(SA\Gamma \log(\Gamma/\epsilon)/\delta)) = O(S\sqrt{A}\Gamma^3 \epsilon^{-1} \log^2(\Gamma/\epsilon) \log(SA\Gamma/\delta)),
\]

as desired. \( \blacksquare \)
4 Lower Bounds

We now state our lower bounds on the number of samples needed to compute $q^*$, $v^*$, and $\pi^*$. Since our proof technique is very modular, we can prove lower bounds for both classical and quantum algorithms with only minor changes.

Our classical lower bounds match known results [AMK12, SWW+18] and use a similar hard MDP instance, but they are qualitatively stronger as explained in the Introduction (end of Section 1.3).

These lower bounds are interesting when the parameters $S$, $A$, and $\Gamma$ are large since the algorithms scale polynomially in these parameters. To avoid edge cases that make the analysis tedious, we only prove the lower bound for $S, A \geq 2$, and $\Gamma \geq 10$ (equivalently $\gamma \in [0.9, 1)$).

**Theorem 8** (Classical and quantum lower bounds). Fix any integers $S,A \geq 2$ and $\gamma \in [0.9, 1)$. Let $\Gamma := (1 - \gamma)^{-1} \geq 10$ and fix any $\epsilon \in (0, \Gamma/4)$. There exists an MDP with $S$ states, $A$ actions, and discount parameter $\gamma$ such that the following lower bounds hold:

1. Given access to a classical generative oracle, any algorithm that computes an $\epsilon$-approximation to $q^*$, $v^*$, or $\pi^*$ must make $\Omega(SA\Gamma^3/\epsilon^2)$ queries.

2. Given access to a quantum generative oracle, any algorithm that computes an $\epsilon$-approximation to $q^*$ must make $\Omega(SA\Gamma^{1.5}/\epsilon)$ queries and any algorithm that computes an $\epsilon$-approximation to $v^*$ or $\pi^*$ must make $\Omega(SA^{1.5}/\epsilon)$ queries.

We first establish the lower bound for an MDP with $S = 2$ and $A = 1$. Note that when $A = 1$, there is only one action per state, so it is trivial to compute the optimal policy. So we can only show hardness for computing $q^*$ or $v^*$, which will be the same because there is only one action.

![MDP diagram](image)

**Figure 1**: The MDP we use for the lower bound with $S = 2$ and $A = 1$. Distinguishing between $p \leq p_0$ and $p \geq p_0 + \alpha$ is hard.

**Lemma 3**. Fix any $\gamma \in [0.9, 1)$. Let $\Gamma := (1 - \gamma)^{-1} \geq 10$ and fix any $\epsilon \in (0, \Gamma/4)$. There exists an MDP shown in Figure 1 with 2 states and 1 action, for which computing $v^*$ (or equivalently, $q^*$) to error $\epsilon$ requires $\Omega(\Gamma^3/\epsilon^2)$ queries to a classical generative oracle or $\Omega(\Gamma^{1.5}/\epsilon)$ queries to a quantum generative oracle.

**Proof.** The MDP shown in Figure 1 has two states we call $s$ and $t$. State $t$ is a sink and the only transition from there is back to $t$ with no reward. Hence $v^*(t) = 0$. State $s$ is a source, and on taking action $a$, there is a reward $r(s,a) = 1$. The transition is probabilistic and controlled by an unknown probability $p \in (0, 1)$. With probability $p$ we come back to $s$, and with probability $1 - p$ we move to $t$. We can compute $v^*(s)$ using the equation $v^*(s) = 1 + \gamma(pv^*(s) + (1 - p)v^*(t))$, which yields

$$v^*(s) = \frac{1}{1 - \gamma p}.$$  (50)
Now further assume that we are promised that $p \leq p_0$ or $p \geq p_0 + \alpha$, where

$$p_0 = 1 - \frac{1}{\Gamma} \quad \text{and} \quad \alpha = \frac{3\epsilon}{\Gamma^2}. \quad (51)$$

Note that $p_0 + \alpha < 1$ because of the way we have chosen the range of $\epsilon$.

We claim that computing $v^*(s)$ to additive error $\epsilon$ will allow us to distinguish these two cases. To see this, note that the difference between the two values of $v^*(s)$ is at least

$$\frac{1}{1 - \gamma(p_0 + \alpha)} - \frac{1}{1 - \gamma p_0} \geq (1 - \gamma(p_0 + \alpha))(1 - \gamma p_0) 0.9\alpha \geq (1 - \gamma p_0)^2 \geq (1.1/\Gamma - 1/10\Gamma^2)^2 \geq 0.9\alpha\Gamma^2/1.21 \geq \alpha\Gamma^2/1.35 \geq 2\epsilon. \quad (52)$$

Thus computing $v^*$ to additive error $\epsilon$ will allow us to distinguish these two possibilities.

Now we just have to show that distinguishing a coin with probability of heads at most $p_0$ or at least $p_0 + \alpha$ given samples from this coin is as hard as claimed in the lower bound. We prove this via query complexity.

Suppose that instead of having sample access to a coin, we have query access to an $n$-bit string $x$ with the promise that either at most $p_0$ fraction of its bits is equal to 1 or at least $p_0 + \alpha$ fraction of its bits is equal to 1. Both quantumly and classically, we can query any bit $x_i$ of $x$ using 1 query.

It is easy to see that we can generate a sample from our coin with probability of heads equal to $|x|/n$ (the fraction of 1s in $x$) with only 1 query to $x$. This works both classically and quantumly.

So we have shown a reduction from the problem of computing $v^*$ to error $\epsilon$ to the problem of deciding whether $|x|/n \leq p_0$ or $|x|/n \geq p_0 + \alpha$ given query access to an $n$-bit string $x$. This is the approximate counting problem. If we count the number of 0s, we want to distinguish $1/\Gamma$ 0s from $(1/\Gamma - 3\epsilon/\Gamma^2)$ 0s. We need to approximate the count to multiplicative precision $O(\epsilon/\Gamma)$. Finally, we can invoke the known lower bounds for approximate counting summarized in Lemma 4. These give a classical lower bound of $\Omega(\Gamma^3/\epsilon^2)$ and a quantum lower bound of $\Omega(\Gamma^{1.5}/\epsilon)$ as claimed. \qed

We formally state the approximate counting lemma used in the previous proof. The quantum bounds are due to [NW99] and [BHMT02].

**Lemma 4** (Approximate counting). Let $x \in \{0,1\}^n$ be a string to which we have standard classical or quantum query access (i.e., we can query the $i$th bit and receive $x_i$). Then deciding whether $|x| \leq k$ or $|x| \geq k(1 + \epsilon)$ for $1 \leq k \leq n/2$ and $\epsilon \geq 1/k$, requires $\Theta(\min\{\frac{n}{\epsilon\sqrt{\Gamma}}, n\})$ classical queries or $\Theta(\frac{1}{\epsilon^2 \sqrt{\Gamma}})$ quantum queries.

We can now extend the lower bound to larger $S$ and $A$. Before doing so, we will need some structural theorems about quantum query complexity and randomized query complexity. For a function $f$, let $R(f)$ and $Q(f)$ denote their randomized and quantum query complexities. The first result shows that computing the logical OR of $k$ copies of a problem scales with $k$. The classical result is due to [GJPW17] and the quantum result follows from a general composition theorem for quantum query complexity in [Rei11]. The second result, known as a direct sum result, can also be found in [Rei11].

**Lemma 5.** Let $\text{OR}_k$ be the logical OR function on $k$ bits and $f$ be an arbitrary Boolean function. Then the complexity of the composed function $\text{OR}_k \circ f$, which is defined as the logical OR of the $k$ outputs
of \( k \) independent instances of \( f \) is related to the complexity of \( f \) as follows: \( Q(\text{OR}_k \circ f) = \Omega(\sqrt{k}Q(f)) \) and \( R(\text{OR}_k \circ f) = \Omega(kR(f)) \). In addition, computing all \( k \) outputs of \( k \) independent instances of \( f \) requires \( \Omega(kR(f)) \) queries classically and \( \Omega(kQ(f)) \) queries quantumly.

Note that the “in addition” result can be viewed as a result about the query complexity of \( f \) composed with the function \( \text{Identity}_k : \{0, 1\}^k \to \{0, 1\}^k \); \( x \mapsto x \).

We are now ready to prove the main lower bound theorem.

**Proof of Theorem 8.** We start by keeping \( S = 2 \) and allowing arbitrarily large \( A \geq 2 \). For notational convenience, we identify \( A \) with \( \{1, \ldots, A\} \).

We will use essentially the same instance as in Fig. 1 but now with \( A \) outgoing actions from state \( s \), each with transition probability \( p_a \) for \( a \in A \). The modified instance is illustrated in Fig. 2. We again consider the case where all the \( p_a \) satisfy the promise that they are either small (\( \leq p_0 \)) or large (\( \geq p_0 + \alpha \)). As argued in the previous proof, deciding if a given \( p_a \) is small or large has a classical lower bound of \( \Omega(\Gamma^3/\epsilon^2) \) and a quantum lower bound of \( \Omega(\Gamma^{1.5}/\epsilon) \).

![Figure 2: The MDP we use for the lower bound with \( S = 2 \) and arbitrary \( A \). For each \( i, p_i \) is promised to be either \( \leq p_0 \) or \( \geq p_0 + \alpha \). Any action \( i \in A \) taken from state \( t \) always returns to \( t \) with zero reward.](image)

Now consider the problem of deciding whether any of the \( p_a \) is small or large. This is the logical \( \text{OR} \) of \( A \) independent problems, each of which we have already shown a lower bound for. If we could compute \( v^* \) to error \( \epsilon \), then we would be able to solve this problem. Hence using Lemma 5, we get a classical lower bound of \( \Omega(A\Gamma^3/\epsilon^2) \) and a quantum lower bound of \( \Omega(\sqrt{A}\Gamma^{1.5}/\epsilon) \) for the problem of computing \( v^* \).

Similarly, consider the problem of deciding which of the \( p_a \) is large, promised that exactly one of them is large and the rest are small. This is similar to logical \( \text{OR} \), except the goal is to identify the location of a 1 promised that it exists. This problem is as hard as logical \( \text{OR} \), and we get the same lower bounds. For such an instance, computing \( \pi^* \) to error \( \epsilon \) will allow us to distinguish the two cases, since \( \pi^*(s) \) should equal the unique action for which \( p_a \) is large. This gives us the claimed lower bounds for \( \pi^* \).

Similarly, consider the problem of learning which \( p_a \)s are large and which are small for all \( a \) (without any promise on the number of each type). This is the problem of solving \( A \) independent instances of a problem for which we have already proved a lower bound. For quantum and classical algorithms, this increases the complexity by a factor of \( A \) as stated in the second part of Lemma 5.
Thus we get a classical lower bound of $\Omega(A\Gamma^3/\epsilon^2)$ and a quantum lower bound of $\Omega(A\Gamma^{1.5}/\epsilon)$ for this problem. But if we could compute $q^*$ to error $\epsilon$, then we would be able to solve this problem since such an estimate encodes whether each $p_a$ is large or small. This gives us the claimed lower bounds for $q^*$.

Thus we have established all the lower bounds for $S = 2$ and arbitrary $A$. Finally, to extend the lower bounds to arbitrarily large $S$, we can just use $S/2$ copies of the MDP in Fig. 2. Computing any one of the quantities $q^*$, $v^*$, or $\pi^*$ on this MDP instance means solving $S/2$ independent copies of the problems discussed above. As stated in the second part of Lemma 5, for both classical and quantum algorithms, this increases the complexity by a factor of $\Omega(S)$. This yields the claimed lower bounds for general $S$ and $A$.

5 Conclusion

To the best of our knowledge, ours is the first work to rigorously study quantum algorithms for solving MDPs. We show that quantum computers can offer quadratic speedups in terms of $\Gamma$, $\epsilon$, and $A$ in calculating $q^*$, $v^*$, and $\pi^*$. We show our algorithms are either optimal, or optimal assuming $\Gamma$ or $A$ is constant, for certain ranges of $\epsilon$. We discuss some open problems left from our work:

1. Can we give optimal algorithms in all parameters $(S, A, \Gamma, \epsilon)$ for an unrestricted range of $\epsilon$?

A first step towards answering this question may be to try to interpolate between SolveMdp1 and SolveMdp2 by adjusting the number of epochs and the length of each epoch. This question partly reduces to the purely classical question of finding a sample-optimal algorithm for $v^*$ and $\pi^*$ that has space complexity $\Theta(S)$ instead of $\Theta(SA)$.

2. Can we circumvent our quantum lower bounds? In our work, we made few assumptions on the MDP. For special classes of MDPs, there may be greater quantum speedups that break our current quantum lower bounds. Such speedups may also be available in the function approximation setting or if we only ask for a few entries of the vectors $q^*$, $v^*$, and $\pi^*$. For example, see [ABI+19].

3. Can we quantize model-based classical algorithms? Our quantum algorithms are all model-free. But classically, the current best MDP solver is model-based [LWC+20]. Therefore it is natural to try to construct a quantum model-based algorithm.

Acknowledgements

We especially thank Wen Sun for suggesting the tabular MDP setting as the first place to search for quantum speedups and for referring us to [AJKW21]. We also thank Aaron Sidford, Mengdi Wang, and Xian Wu for helpful discussions on [SWWY21]. DW acknowledges funding by the Army Research Office (grant W911NF-20-1-0015) and NSF award DMR-1747426. Part of this work was performed while DW was an intern at Microsoft.

A Construction of the Quantum Generative Model

In this appendix, we describe how to systematically and efficiently construct the quantum generative model (Definition 3) from a circuit $C$ for a classical generative model.

Recall the definition of a classical generative model: for a given state-action pair $(s, a) \in (S, A)$, $C$ generates $s'$ with probability $p(s'|s, a)$. Since $C$ is the circuit of a randomized algorithm, it can be represented as a deterministic circuit that takes in two inputs $(s, a) \in (S, A)$ and $x \in \{0, 1\}^m$, and
outputs \( s' \in \mathcal{S} \) with
\[
\Pr_{x \sim \{0,1\}^m}(\mathcal{C}(s, a, x) = s') = p(s'|s, a),
\]
(53)
where \( x \sim \{0,1\}^m \) means \( x \) is uniformly selected from \( \{0,1\}^m \). That is
\[
|\{x \in \{0,1\}^m | \mathcal{C}(s, a, x) = s'\}| = 2^m \cdot p(s'|s, a).
\]
(54)

Pictorially, \( \mathcal{C} \) is of the form:
\[
\begin{array}{c}
\mathcal{S} \times \mathcal{A} \ni (s, a) \\
\{0,1\}^m \ni x \\
\mathcal{S} \ni 0_S \\
0^n
\end{array}
\]
\[
\mathcal{C}(s, a, x)
\]
(55)

Now, as \( \mathcal{C} \) is a deterministic circuit, we can systematically make it a reversible classical circuit by [Ben73] (see [NC00, Sec. 1.4.1] for a textbook exposition). This gives another circuit, \( \mathcal{C}' \), consisting of \( O(\text{size}(\mathcal{C})) \) Toffoli and \text{NOT} gates that uses an additional \( O(\text{size}(\mathcal{C})) \) ancillary bits\(^4\), of the form:
\[
\begin{array}{c}
\mathcal{S} \times \mathcal{A} \ni (s, a) \\
\{0,1\}^m \ni x \\
\mathcal{S} \ni 0_S \\
0^n
\end{array}
\]
\[
\mathcal{C}'(s, a, x)
\]
(56)
where the \( 0_S \) input of the third wire represents some fixed state in \( \mathcal{S} \), the wires at the same height (to the left and right of \( \mathcal{C}' \)) use registers of the same size, and \( 0^n \) is an ancillary bitstring with \( n = O(\text{size}(\mathcal{C})) \).

Now, we can change all the classical Toffoli and \text{NOT} gates in \( \mathcal{C}' \) into quantum Toffoli and Pauli-X gates (note that this changes the physical implementation of \( \mathcal{C}' \) to produce a quantum circuit \( \mathcal{Q}' \). \( \mathcal{Q}' \) behaves the same as \( \mathcal{C}' \) on classical inputs but is now also able to accept quantum superpositions of these classical inputs. Pictorially, \( \mathcal{Q}' \) is of the form:
\[
\begin{array}{c}
\mathcal{C}^S \ni |s,a\rangle \\
(C^2)^\otimes m \ni |x\rangle \\
\mathcal{C}^S \ni |0_S\rangle \\
|0\rangle^{\otimes n}
\end{array}
\]
\[
|s,a\rangle \\
|x\rangle \\
|\mathcal{C}(s, a, x)\rangle \\
|0\rangle^{\otimes n}
\]
(57)

Now, we append a Hadamard gate to each of the \( m \) qubits of the second register of \( \mathcal{Q}' \) at the start of the computation to give \( \mathcal{Q} \). Pictorially, \( \mathcal{Q} \) is of the form:
\[
\begin{array}{c}
\mathcal{C}^S \ni |s,a\rangle \\
(C^2)^\otimes m \ni |0\rangle^{\otimes m} \\
\mathcal{C}^S \ni |0_S\rangle \\
|0\rangle^{\otimes n}
\end{array}
\]
\[
\begin{array}{c}
|H^\otimes m|
\end{array}
\]
\[
\mathcal{Q}'
\]
(58)

We can compute the output of \( \mathcal{Q} \) on the input \( |s,a\rangle |0\rangle^{\otimes m} |0_S\rangle |0\rangle^{\otimes n} \equiv |s,a,0^m,0_S,0^n\rangle \) as follows.

\(^4\)We can be more precise if \( \mathcal{C} \) consists entirely of \text{NAND} gates (\text{NAND} is universal for classical computation). In this case, \( \mathcal{C}' \) would, at most, consist of \( 2 \times \text{size}(\mathcal{C}) \) Toffoli gates and \( 2 \times \text{size}(\mathcal{C}) \) \text{NOT} gates, and use an additional \( \text{size}(\mathcal{C}) \) ancillary bits.
\[ \mathcal{Q} |s, a, 0^m, 0_S, 0^n\rangle = Q' \frac{1}{\sqrt{2^m}} \sum_{x \in \{0, 1\}^m} |s, a, x, 0_S, 0^n\rangle \] (apply \( H^\otimes m \))

\[ = \frac{1}{\sqrt{2^m}} \sum_{x \in \{0, 1\}^m} |s, a, x, C(s, a, x), 0^n\rangle \] (Eq. (57))

\[ = |s, a\rangle \sum_{s' \in S} \left( \frac{1}{\sqrt{2^m}} \sum_{x \in \{0, 1\}^m} |x\rangle \right) |s'\rangle |0^n\rangle \] (rearrange sum).

Now, the \( m \)-qubit state in the brackets has norm \( |\{x \in \{0, 1\}^m \mid C(s, a, x) = s'\}|/2^m = p(s'|s, a) \) due to Eq. (54). Therefore, it can be rewritten as \( \sqrt{p(s'|s, a)} |\psi_{s, a, s'}\rangle \) for some normalized state \( |\psi_{s, a, s'}\rangle \). Therefore

\[ \mathcal{Q} |s, a, 0^m, 0_S, 0^n\rangle = |s, a\rangle \sum_{s' \in S} \sqrt{p(s'|s, a)} |\psi_{s, a, s'}\rangle |s'\rangle |0^n\rangle \] (59)

Finally, swapping the \( |\psi_{s, a, s'}\rangle \) and \( |s'\rangle \) registers, dropping the \( |0^n\rangle \) register for convenience, and renaming \( 0_S \) to just \( 0 \), we see that \( \mathcal{Q} \) is precisely of the form in Definition 3.

References

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, ..., Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. Nature, 574(7779):505–510, 2019. doi:10.1038/s41586-019-1666-5. [p. 1]

[ABI+19] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Kristjání Prúsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1783–1793. Society for Industrial and Applied Mathematics, 2019. doi:10.1137/1.9781611975482.107. [p. 21]

[AJKW21] Alekh Agarwal, Nan Jiang, Sham M. Kakade, and Sun Wen. Reinforcement Learning: Theory and Algorithms. Book in preparation, draft available at rltheorybook.github.io. Date accessed: 2nd November, 2021. [pp. 1, 2, 8, 9, 21]

[AKY20] Alekh Agarwal, Sham Kakade, and Lin F. Yang. Model-Based Reinforcement Learning with a Generative Model is Minimax Optimal. In Proceedings of the 33rd Conference On Learning Theory (COLT), volume 125 of Proceedings of Machine Learning Research, pages 67–83. PMLR, 2020. [p. 8]

[AMK12] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J. Kappen. On the Sample Complexity of Reinforcement Learning with a Generative Model. In Proceedings of the 29th International Conference on Machine Learning (ICML), pages 1707–1714, 2012. [pp. 3, 8, 9, 18]

[Bel19] Aleksandrs Belovs. Quantum Algorithms for Classical Probability Distributions, 2019. arXiv:1904.02192 [p. 10]
[Ben73] C. H. Bennett. Logical Reversibility of Computation. *IBM J. Res. Dev.*, 17(6):525–532, 1973. doi:10.1147/rd.176.0525. [pp. 3, 22]

[Ber00] D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Number v. 1 in Athena Scientific optimization and computation series. Athena Scientific, 2000. [p. 1]

[Ber13] D.P. Bertsekas. *Abstract Dynamic Programming*. Athena Scientific, 2013. [pp. 1, 6]

[BHMT02] Gilles Brassard, Peter Höyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Information*, page 53–74, 2002. doi:10.1090/conm/305/05215. [pp. 4, 10, 19]

[CJ21] Arjan Cornelissen and Sofiene Jerbi. Quantum algorithms for multivariate Monte Carlo estimation, 2021. arXiv:2107.03410 [p. 8]

[DCLT08] D. Dong, C. Chen, H. Li, and T. Tarn. Quantum Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1207–1220, 2008. doi:10.1109/TSMCB.2008.925743. [p. 8]

[DH96] Christoph Dürr and Peter Höyer. A Quantum Algorithm for Finding the Minimum. 1996. arXiv:quant-ph/9607014 [pp. 4, 11]

[DTB16] Vedran Dunjko, Jacob M. Taylor, and Hans J. Briegel. Quantum-Enhanced Machine Learning. *Physical Review Letters*, 117(13), 2016. doi:10.1103/PhysRevLett.117.130501. [p. 8]

[DTB17] V. Dunjko, J. M. Taylor, and H. J. Briegel. Advances in quantum reinforcement learning. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 282–287, 2017. doi:10.1109/SMC.2017.8122616. [p. 8]

[GJPW17] Mika Göös, T. S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized Communication vs. Partition Number. In *44th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.52. [p. 19]

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum Random Access Memory. *Physical Review Letters*, 100:160501, 2008. doi:10.1103/PhysRevLett.100.160501. [p. 4]

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM Symposium on the Theory of Computing (STOC)*, pages 212–219, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/237814.237866. [p. 1]

[Ham21] Yassine Hamoudi. Quantum Sub-Gaussian Mean Estimator. In *29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2021.50. [p. 7]

[HM19] Yassine Hamoudi and Frédéric Magniez. Quantum Chebyshev’s Inequality and Applications. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132 of *Leibniz International Proceedings in Informatics*
[JS20] Yujia Jin and Aaron Sidford. Efficiently Solving MDPs with Stochastic Mirror Descent. In Proceedings of the 37th International Conference on Machine Learning (ICML), volume 119 of Proceedings of Machine Learning Research, pages 4890–4900. PMLR, 2020. [p. 8]

[JTPN+21] Sofiene Jerbi, Lea M. Trenkwalder, Hendrik Poulsen Nautrup, Hans J. Briegel, and Vedran Dunjko. Quantum Enhancements for Deep Reinforcement Learning in Large Spaces. PRX Quantum, 2:010328, 2021. doi:10.1103/PRXQuantum.2.010328. [p. 8]

[JVV86] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. Theoretical Computer Science, 43:169–188, 1986. doi:10.1016/0304-3975(86)90174-X. [p. 11]

[Kak03] Sham M. Kakade. On the Sample Complexity of Reinforcement Learning. PhD thesis, 2003. [p. 2]

[KMN02] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. Machine Learning, 49(2):193–208, 2002. doi:10.1023/A:1017932429737. [p. 2]

[KS99] Michael Kearns and Satinder Singh. Finite-Sample Convergence Rates for Q-Learning and Indirect Algorithms. In Advances in Neural Information Processing Systems 11 (NeurIPS), pages 996–1002, Cambridge, MA, USA, 1999. MIT Press. [pp. 2, 4, 8]

[LWC+20] Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. Breaking the Sample Size Barrier in Model-Based Reinforcement Learning with a Generative Model. In Advances in Neural Information Processing Systems 33 (NeurIPS), volume 33, pages 12861–12872. Curran Associates, Inc., 2020. [pp. 3, 8, 21]

[Mon15] Ashley Montanaro. Quantum speedup of Monte Carlo methods. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 471(2181):20150301, 2015. doi:10.1098/rspa.2015.0301. [pp. 4, 10]

[NC00] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2000. [pp. 3, 9, 22]

[NW99] Ashwin Nayak and Felix Wu. The quantum query complexity of approximating the median and related statistics. In Proceedings of the 31st ACM Symposium on the Theory of Computing (STOC), pages 384–393, New York, NY, USA, 1999. Association for Computing Machinery. doi:10.1145/301250.301349. [p. 19]

[PDM+14] Giuseppe Davide Paparo, Vedran Dunjko, Adi Makmal, Miguel Angel Martin-Delgado, and Hans J. Briegel. Quantum Speedup for Active Learning Agents. Physical Review X, 4(3):031002, 2014. doi:10.1103/PhysRevX.4.031002. [p. 8]

[Rei11] Ben W. Reichardt. Reflections for Quantum Query Algorithms. In Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 560–569, USA, 2011. Society for Industrial and Applied Mathematics. [p. 19]
[SB18] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA, USA, 2018. [p. 1]

[Sho97] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing, 26(5):1484–1509, 1997. doi:10.1137/S0097539795293172. [p. 1]

[SWW+18] Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-Optimal Time and Sample Complexities for Solving Markov Decision Processes with a Generative Model. In Advances in Neural Information Processing Systems 31 (NeurIPS), pages 5186–5196. Curran Associates, Inc., 2018. [pp. 3, 4, 6, 7, 8, 11, 12, 18]

[SWWY21] Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving Markov decision processes. Naval Research Logistics (NRL), 2021. doi:10.1002/nav.21992. [pp. 6, 8, 21]

[SY94] Satinder P. Singh and Richard C. Yee. An upper bound on the loss from approximate optimal-value functions. Machine Learning, 16(3):227–233, 1994. doi:10.1007/BF00993308. [p. 6]

[Sze10] Csaba Szepesvári. Algorithms for Reinforcement Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 4(1):1–103, 2010. doi:10.2200/S00268ED1V01Y201005AIM009. [p. 1]

[vA21] Joran van Apeldoorn. Quantum Probability Oracles & Multidimensional Amplitude Estimation. In 16th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC), volume 197 of Leibniz International Proceedings in Informatics (LIPIcs), pages 9:1–9:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.TQC.2021.9. [p. 8]

[Wai19] Martin J. Wainwright. Variance-reduced Q-learning is minimax optimal, 2019. arXiv:1906.04697 [pp. 6, 8]

[Wan17] Mengdi Wang. Primal-Dual π Learning: Sample Complexity and Sublinear Run Time for Ergodic Markov Decision Problems, 2017. arXiv:1710.06100 [p. 8]

[Wan20] Mengdi Wang. Randomized Linear Programming Solves the Markov Decision Problem in Nearly Linear (Sometimes Sublinear) Time. Mathematics of Operations Research, 45(2):517–546, 2020. doi:10.1287/moor.2019.1000. [p. 8]

[WSK+21] Daochen Wang, Aarthi Sundaram, Robin Kothari, Ashish Kapoor, and Martin Roetteler. Quantum algorithms for reinforcement learning with a generative model. In Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 10916–10926. PMLR, 18–24 Jul 2021. [pp. 1, 7]

[WYLC21] Daochen Wang, Xuchen You, Tongyang Li, and Andrew M. Childs. Quantum Exploration Algorithms for Multi-Armed Bandits. Proceedings of the AAAI Conference on Artificial Intelligence, 35(11):10102–10110, 2021. [p. 8]