Quantum Algorithms for Reinforcement Learning with a Generative Model

Daochen Wang [1] **Aarthi Sundaram** [2] **Robin Kothari** [2] **Ashish Kapoor** [3] **Martin Roetteler** [2]

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 γ-discounted Markov decision process (MDP). For such an MDP, we design quantum algorithms that approximate an optimal policy (π^*), 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 (ϵ) 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 (Bertsekas, 2000; 2013; Szepesvári, 2010; Sutton & Barto, 2018; Agarwal et al., 2021). The framework has been successfully applied to problems in healthcare, robotics, engineering, gaming, natural language processing, finance, and so on.

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 (Grover, 1996; Shor, 1997). Recent experiments have achieved key milestones (Arute et al., 2019), 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. We assume that we have quantum access to the environment, which we will justify.

1.1. Problem Setup

We study infinite-horizon discounted MDPs M with a finite set $\mathcal{S}$ of states where at each state an agent can choose to take an action from a finite set $\mathcal{A}$ of actions. Upon taking an action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$, the agent receives reward[1] $r[s,a] \in [0,1]$ and transitions to a state $s' \in \mathcal{S}$ with some probability $p(s'|s,a)$. The last parameter needed to specify M is a number $\gamma \in [0,1)$ which discounts the reward the agent receives at later time steps t by a factor of γ^t. Hence, M is conveniently summarized by a 5-tuple, $M = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$. For convenience, we define $S = |\mathcal{S}|$ and $A = |\mathcal{A}|$, the cardinalities of $\mathcal{S}$ and $\mathcal{A}$ 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 γ-discounted rewards over infinitely many time steps. Following standard practice, we assume the agent has full knowledge of $\mathcal{S}$, $\mathcal{A}$, r, and γ but not p at the outset. A primary objective is to compute a policy $\pi : \mathcal{S} \to \mathcal{A}$ for the agent that specifies the action $a = \pi(s)$ it should take at $s \in \mathcal{S}$ to best achieve its goal with high probability.

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

$$\begin{aligned} v^\pi[s] &= \mathbb{E}\big[\textstyle\sum_{t=0}^{\infty} \gamma^t r[s_t, a_t] \mid {}^{s_0 = s}_{\forall i \geq 0 : a_i = \pi[s_i]}\big], \\ q^\pi[s,a] &= \mathbb{E}\big[\textstyle\sum_{t=0}^{\infty} \gamma^t r[s_t, a_t] \mid {}^{s_0 = s,\ a_0 = a}_{\forall i \geq 1 : a_i = \pi[s_i]}\big], \end{aligned} \tag{1}$$

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

[1]University of Maryland [2]Microsoft Quantum [3]Microsoft. Correspondence to: Daochen Wang <wdaochen@gmail.com>, Aarthi Sundaram <aarthi.sundaram@microsoft.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

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

Table 1. Quantum computing allows for speedups in terms of the parameters ϵ, $\Gamma := (1-\gamma)^{-1}$, and A, but not S. All bounds are for maximum failure probability δ being constant. All upper bounds are $\widetilde{O}(\cdot)$, with unrestricted ϵ except when [Theorem 5] appears, in which case we assume $\epsilon \in O(1/\sqrt{\Gamma})$. All lower bounds are $\Omega(\cdot)$ and stated for $\epsilon \in O(\Gamma)$. The classical upper bounds are shown in (Li et al., 2020) for all ϵ; the classical lower bounds are shown in (Azar et al., 2012) for q^*, v^* and (Sidford et al., 2018a) for π^*.

Goal: output an ϵ-accurate estimate of	Classical sample complexity	Quantum sample complexity	
	Upper and lower bound	Upper bound	Lower bound
q^*	$\frac{SA\Gamma^3}{\epsilon^2}$	$\frac{SA\Gamma^{1.5}}{\epsilon}$ [Theorem 5]	$\frac{SA\Gamma^{1.5}}{\epsilon}$ [Theorem 8]
v^*, π^*	$\frac{SA\Gamma^3}{\epsilon^2}$	$\frac{SA\Gamma^{1.5}}{\epsilon}$ [Theorem 5] $\frac{S\sqrt{A}\Gamma}{\epsilon}$ [Theorem 7]	$\frac{S\sqrt{A}\Gamma^{1.5}}{\epsilon}$ [Theorem 8]

sums in Eq. (1) can take is Γ, and hence $v^\pi[s]$ and $q^\pi[s,a]$ are in $[0,\Gamma]$. It is known that any discounted MDP admits an optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$, in the strong sense that $v^{\pi^*}[s] \geq v^\pi[s]$ for all $\pi \in \Pi, s \in \mathcal{S}, a \in \mathcal{A}$, where Π is the space of all policies (which could even contain random and non-stationary policies). It is common to denote $v^* := v^{\pi^*}$ and $q^* := q^{\pi^*}$.

We can now state our main computational goals precisely. Using $\|\cdot\|$ for the infinity norm, 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 $\|v^* - v^{\hat{\pi}}\| \leq \epsilon$ with probability at least $1-\delta$. 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 its time complexity or the number of samples it draws from the unknown distribution $p(s'|s,a)$. Our paper is concerned with the latter and assumes the *generative model* of sampling, as studied by Kearns & Singh (1999), Kearns et al. (2002), and Kakade (2003), meaning that we can choose *arbitrary* $(s,a) \in \mathcal{S} \times \mathcal{A}$ 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 simulator. The generative model makes particular sense when the environment is a computer program, whence the simulator is that program.

We now let quantum computing enter the picture. If the simulator is itself a computer program and we have its source code, then we can produce a Boolean circuit G that acts as the simulator, i.e., draws samples from the distribution $p(\cdot|s,a)$. We can use the following basic fact in quantum computation to efficiently convert G to a quantum circuit $\mathcal{G}$.

Fact 1 (Nielsen and Chuang (Sec. 1.5.1), 2000 ; Bennett, 1973; 1989)**.** *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. $\mathcal{G}$ is formally defined in Section 2.3.

Under this setup, our goal is to design quantum algorithms approximating q^*, π^*, 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 T or A is constant, for certain ranges of ϵ.

1.2. Main Results

Table 1 summarizes our main results. The classical sample complexities have only recently been completely characterized for all three quantities (Li et al., 2020) 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 Γ and ϵ if $\epsilon = O(1/\sqrt{\Gamma})$. For computing v^* and π^*, we construct a second quantum algorithm that offers an additional quadratic speedup in terms of A at the expense of Γ. Moreover, we prove quantum lower bounds

Algorithm 1 SolveMdp1(M, ϵ, δ)

Input: MDP $M = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, maximum error $\epsilon \in (0, \sqrt{\Gamma}]$, and maximum failure probability $\delta \in (0, 1)$.
Output: $\hat{v} \coloneqq v_{K,L} \in \mathbb{R}^S$, $\hat{\pi} \coloneqq \pi_{K,L} \in \mathcal{A}^S$, and $\hat{q} \coloneqq q_{K,L} \in \mathbb{R}^{SA}$.
Initialize: $K \leftarrow \lceil \log_2(\Gamma/\epsilon) \rceil$, $L \leftarrow \Gamma \lceil \ln(4\Gamma/\epsilon) \rceil + 1$, $f \leftarrow \delta/4KLSA$, $b \leftarrow 1$, $c \leftarrow 0.01$
Initialize: $v_{1,0} \leftarrow \mathbf{0}$, $\pi_{1,0} \leftarrow$ arbitrary, $q_{1,0} \leftarrow \mathbf{0}$
for $k \in [K]$ **do**
  $\epsilon_k \leftarrow \Gamma/2^k$
  $\forall (s,a) \in \mathcal{S} \times \mathcal{A} : y_k[s,a] \leftarrow \max\{\mathsf{qEst1}_f((Pv_{k,0}^2)[s,a],\, b) - (\mathsf{qEst1}_f((Pv_{k,0})[s,a],\, (1-\gamma)b))^2, 0\}$
  $\forall (s,a) \in \mathcal{S} \times \mathcal{A} : x_k[s,a] \leftarrow \mathsf{qEst2}_f((Pv_{k,0})[s,a],\, c(1-\gamma)^{1.5}\epsilon\sqrt{y_k[s,a]+b}) - c(1-\gamma)^{1.5}\epsilon\sqrt{y_k[s,a]+b}$
  for $l \in [L]$ **do**
    $\forall s \in \mathcal{S}$: **if** $v(q_{k,l-1})[s] \geq v_{k,l-1}[s]$ **then** $v_{k,l}[s] \leftarrow v(q_{k,l-1})[s]$, $\pi_{k,l}[s] \leftarrow \pi(q_{k,l-1})[s]$
    else $v_{k,l}[s] \leftarrow v_{k,l-1}[s]$, $\pi_{k,l}[s] \leftarrow \pi_{k,l-1}[s]$ **end if**
    $\forall (s,a) \in \mathcal{S} \times \mathcal{A} : \Delta_{k,l}[s,a] \leftarrow \mathsf{qEst1}_f((P(v_{k,l} - v_{k,0}))[s,a],\, c(1-\gamma)\epsilon_k) - c(1-\gamma)\epsilon_k$
    $q_{k,l} \leftarrow \max\{r + \gamma(x_k + \Delta_{k,l}), \mathbf{0}\}$
  end for
  $v_{k+1,0} \leftarrow v_{k,L}$, $\pi_{k+1,0} \leftarrow \pi_{k,L}$, $q_{k+1,0} \leftarrow q_{k,L}$
end for

for computing all three quantities. Our lower bounds show that our q^* algorithm can be optimal, that we have optimal algorithms for v^* and π^* provided one of Γ or A is constant, but that there may still be a faster quantum algorithm for v^* and π^*. We remark that we also re-prove a version of the *classical* lower bounds that is qualitatively stronger than those existing as explained at the end of the next section.

We remark that the time complexities of our 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) (Giovannetti et al., 2008). This is because the classical algorithm of Sidford et al. (2018a) that we quantize satisfies this property and the quantum subroutines we use to quantize it also satisfies this property.

1.3. Technical Overview

We now give an overview of the techniques we used in our two quantum algorithms, SolveMdp1 and SolveMdp2. SolveMdp1 and 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 (Sidford et al., 2018a). We first discuss the quantum subroutines: quantum mean estimation (Brassard et al., 2000; Montanaro, 2015) and quantum maximum finding (Dürr & Høyer, 1996).

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

Quantum maximum finding, denoted 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. qArgmax is used in 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 Sidford et al. (2018a). To warm up, consider how they can be applied to standard value iteration (Kearns & Singh, 1999) to compute π^*. In standard value iteration, we start with v_0 set to the zero vector in $\mathbb{R}^{\mathcal{S}}$ and repeatedly update it by the Bellman recursion $v_i \leftarrow \mathcal{T}(v_{i-1})$ where the Bellman operator $\mathcal{T} : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ is defined by

$$\mathcal{T}(v_{i+1})[s] \coloneqq \max_a \{r[s,a] + \gamma \mathbb{E}[v_i[s'] \mid s' \sim p(\cdot|s,a)]\}, \tag{2}$$

for all $s \in \mathcal{S}$. For convenience, we abbreviate the mean $\mathbb{E}[v_i[s'] \mid s' \sim p(\cdot|s,a)]$ as μ_i. If this mean is computed exactly at each iteration, then the recursion takes $O(\Gamma)$ iterations to converge to π^*, which follows from basic contractive properties of $\mathcal{T}$. In reality, we cannot compute μ_i exactly. But if we only require our final answer to be

Algorithm 2 SolveMdp2(M, ϵ, δ)

Input: MDP $M = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, maximum error $\epsilon \in (0, \Gamma]$, and maximum failure probability $\delta \in (0, 1)$.
Output: $\hat{v} := v_L \in \mathbb{R}^S$ and $\hat{\pi} := \pi_L \in \mathcal{A}^S$.
Initialize: $L \leftarrow \Gamma\lceil \log(4\Gamma/\epsilon) \rceil + 1$, $f \leftarrow \delta / 4c_{\max} LSA^{1.5} \log(1/\delta)$
Initialize: $v_0 \leftarrow \mathbf{0}$, $\pi_0 \leftarrow$ arbitrary, $\forall s \in \mathcal{S} : q_{0,s} \leftarrow \mathbf{0} \in \mathbb{R}^A$
for $l \in [L]$ **do**
  $\forall s \in \mathcal{S} : a^*[s] \leftarrow \mathsf{qArgmax}_f\{q_{l-1,s}[a] \ : \ a \in \mathcal{A}\}$
  $\forall s \in \mathcal{S} : \tilde{\pi}_l[s] \leftarrow a^*[s]$, $\tilde{v}_l[s] \leftarrow q_{l-1,s}[a^*[s]]$
  $\forall s \in \mathcal{S}$: **if** $\tilde{v}_l[s] \geq v_{l-1}[s]$ **then** $v_l[s] \leftarrow \tilde{v}_l[s]$, $\pi_l[s] \leftarrow \tilde{\pi}_l[s]$
  else $v_l[s] \leftarrow v_{l-1}[s]$, $\pi_l[s] \leftarrow \pi_{l-1}[s]$ **end if**
  $\forall s \in \mathcal{S}$: create quantum oracle encoding, $U_{z_{l,s}}$, of $z_{l,s} \in \mathbb{R}^A$ defined by
    $z_{l,s}[a] \leftarrow \mathsf{qEst1}_f((Pv_l)[s,a],\ (1-\gamma)\epsilon/4) - (1-\gamma)\epsilon/4$
  $\forall s \in \mathcal{S}$: create quantum oracle encoding, $U_{q_{l,s}}$, of $q_{l,s} \in \mathbb{R}^A$ defined by
    $q_{l,s}[a] \leftarrow \max\{r[s,a] + \gamma z_{l,s}[a], 0\}$
end for

correct to error ϵ, then it is reasonable to assume that estimating μ_i for each i to error ϵ/Γ suffices. If we make the reasonable assumption $\|v_i\| \leq \|v^*\| \leq \Gamma$ (v_i is converging to v^* after all) then classically 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 $(s,a) \in \mathcal{S} \times \mathcal{A}$. Therefore, the overall classical sample complexity is of order $O(SA\Gamma^5/\epsilon^2)$. Though the preceding argument is non-rigorous, it does in fact give the right answer (up to log-factors) (Sidford et al., 2018b).

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 $(s,a) \in \mathcal{S} \times \mathcal{A}$, 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 Γ iterations, gives an overall quantum sample complexity of $O(SA\Gamma^3/\epsilon^2)$. 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 $O(S\sqrt{A}\Gamma^3/\epsilon^2)$, which matches the performance of SolveMdp2 for v^*. However, an ϵ-optimal value function leads only to an $(2\gamma\Gamma\epsilon)$-optimal greedy policy (Singh & Yee, 1994; Bertsekas, 2013).

Quantum version of modern value iteration. To obtain an ϵ-optimal policy, SolveMdp1 and SolveMdp2 directly employ the so-called monotonicity technique of (Sidford et al., 2018a) 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 qEst. Note that the subtracted terms always equal the preceding estimation error which enforces one-sided error. The overall effect of the monotonicity technique is to ensure the value function at each iteration is at most the value function *of the policy* at that iteration (which is in turn at most v^*). Hence, we avoid the problem of an ϵ-optimal $\hat{v}$ not giving an ϵ-optimal $\hat{\pi}$.

We can get better dependence in Γ by leveraging two other techniques introduced in (Sidford et al., 2018a;b; Wainwright, 2019): "variance reduction" and "total variance". We incorporate these techniques in SolveMdp1 at the cost of re-inflating the A dependence back to linear. The reason we no longer get $\sqrt{A}$ is because applying qArgmax 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 SolveMdp1 are indexed by k. At the l-th iteration of epoch k, we need to estimate $\mathbb{E}[v_{k,l}[s']]$, where $v_{k,l}$ is the current value function. The mean can be rewritten as

$$\mathbb{E}[v_{k,l}[s']] = \mathbb{E}[(v_{k,l} - v_{k,0})[s']] + \mathbb{E}[v_{k,0}], \qquad (3)$$

where $v_{k,0}$ is the value function at the start of the epoch. There are SA of these equations, one corresponding to each $(s,a) \in \mathcal{S} \times \mathcal{A}$ 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 ϵ range for which SolveMdp1 is optimal. Consider the second term, $\mathbb{E}[v_{k,0}]$. This again needs to be estimated to error ϵ/Γ 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 $\mathbb{E}[v_{k,0}]$ *once per epoch* and reuse its value throughout the epoch. But there are only logarithmically many epochs, so the overall cost becomes $\widetilde{O}(SA\Gamma^4/\epsilon^2)$ classically and $\widetilde{O}(SA\Gamma^2/\epsilon)$ 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 ϵ/Γ so that over Γ iterations, the accumulated error is ϵ. However, the error at each iteration i can actually be set larger, to $\epsilon\sqrt{\mathrm{Var}[v_i[s']]}/\Gamma^{1.5}$ (which could be as large as $\epsilon/\sqrt{\Gamma}$), and it can still be shown that the overall accumulated error is ϵ using properties of the standard deviation. More specifically, let us write $\sigma_i = \sqrt{\mathrm{Var}[v_i[s']]}$. 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 μ_i ($:= \mathbb{E}[v_i[s']]$) to an error of $\epsilon\sigma_i/\Gamma^{1.5}$, without needing to know the σ_is. 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 $\mathbb{E}[v_{k,0}]$ from before, we see that this yields an overall classical sample complexity of $\widetilde{O}(SA\Gamma^3/\epsilon^2)$. This is one main result of (Sidford et al., 2018a). 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 reason. The version of quantum mean estimation that should have corresponded to a more efficient Chebyshev's inequality, namely qEst2, is deficient compared to its classical counterpart in two ways. The first is that qEst2 cannot estimate μ_i to an error proportional to σ_i without knowing σ_i a priori. To remedy this, we first estimate σ_i using qEst1 to some additive error $b > 0$. Denote the estimate by $\hat{\sigma}_i$. Then we can use qEst2 to estimate μ_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 qEst2. In fact, qEst2 also requires an upper bound C on σ_i to function and uses $O(C/\epsilon)$ samples to guarantee additive error ϵ. 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 μ_i to error proportional to σ_{high}, so that $C/\,\sigma_{\text{high}} = 1$ becomes constant. Of course, this no longer maintains correctness as σ_{high} is larger than σ_i. However, we can bound $\sigma_{\text{high}} \leq \sigma_i + 2b$. 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 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 σ_i to error b by qEst1 is within our budget. With the technical challenges resolved, we see that the complexity of SolveMdp1 is $\widetilde{O}(SA(\epsilon/\Gamma^{1.5})^{-1}) = \widetilde{O}(SA\Gamma^{1.5}/\epsilon)$. Again, due to the first term on the RHS of Eq. (3), this only holds for $\epsilon = O(1/\sqrt{\Gamma})$. The ϵ range is smaller than before, which was $\epsilon = O(1)$, because there is relatively less quantum speedup for estimating that first term.

In summary, we have described SolveMdp1, which uses qEst to "quantize" all three techniques in (Sidford et al., 2018a): 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 approach to resolving that challenge could find uses in quantizing other classical algorithms as well. We have also described SolveMdp2, which offers a quadratic speedup in A using qArgmax. But because qArgmax conflicts with the variance reduction and total variance techniques, SolveMdp2 no longer has optimal Γ 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 $\mathcal{G}$ 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 π^* by instantiating our oracle $\mathcal{G}$ using standard Boolean oracles. For example, consider a quantum oracle $\mathcal{G}_{\text{coin}}$ that produces a state which represents a quantum sample of a coin toss with probability p of getting heads. $\mathcal{G}_{\text{coin}}$ 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 π^* using oracle $\mathcal{G}$.

This approach has some unexpected benefits. Because we reduce to standard problems in query complexity, our proof is very modular. Without extra effort, it allows us to 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 (Azar et al., 2012) 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 (Sidford et al., 2018a;b; Wainwright, 2019) which represent the latest development of classical *model-free* MDP solvers, which also recently include (Wang, 2017; 2020; Jin & Sidford, 2020) among others, that started with (Kearns & Singh, 1999). Sidford et al. (2018a) give algorithms with complexity $\widetilde{O}(SA\Gamma^3/\epsilon^2)$ when $\epsilon = O(1)$ for approximating all three of q^*, v^*, and π^*. On the model-free side, there has been even more recent progress culminating in the work of Li et al. (2020) which achieves $\widetilde{O}(SA\Gamma^3/\epsilon^2)$ for the full range of $\epsilon \in (0, \Gamma]$. That this bound is tight (up to log-factors) is established by (Azar et al., 2012) which is closely related to our work. Indeed, to prove our lower bounds, we use an instance inspired by (Azar et al., 2012). 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 (Azar et al., 2012; Agarwal et al., 2020; Li et al., 2020) 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 ϵ in ℓ_1-norm can be $O(n/\epsilon)$, which represents a quadratic speedup over classical in terms of ϵ. Even for this simple question, the answer is currently unknown, cf. (Chakraborty et al., 2010) for lower bounds.

On the quantum side, the broader subject of reinforcement learning "remains relatively unaddressed by the quantum community" (Jerbi et al., 2021). The relatively few works on the subject include (Dong et al., 2008; Dunjko et al., 2016; Paparo et al., 2014; Dunjko et al., 2017; Jerbi et al., 2021). 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 π^*, v^*, and q^*. Some of these works do mention the possibility of quadratic speedups by using quantum maximum finding (Dunjko et al., 2016). However, they do not consider how this technique could be used as part of a larger algorithm. As we have mentioned, our work shows that to achieve optimal T-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 optimal quadratic quantum speedup is shown in (Wang et al., 2021). In synergy with our work, Dunjko et al. (2017) 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 $\mathbf{1}$ (resp. $\mathbf{0}$) 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 $x \cdot \mathbf{1}$. 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 $X^{\mathcal{S}}$, X^A with $X^{\mathcal{A}}$, $X^{S\times A}$ with $X^{\mathcal{S}\times\mathcal{A}}$, and so on.

2.2. MDP Preliminaries

For a policy π, 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 $(s,a) \in \mathcal{S}\times\mathcal{A}$, 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 $(Pu)[s,a] = p_{s,a}^\mathsf{T} u$.

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

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

$$\mathcal{T}^\pi(u)[s] \coloneqq r(s,\pi[s]) + \gamma\, p_{s,\pi[s]}^\mathsf{T} u. \tag{5}$$

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

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 $\pi(q)[s] = \mathrm{argmax}_a\{q[s,a]\}$ 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. *(Agarwal et al., 2021; Azar et al., 2012) For any policy π, 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 computing notation (Dirac notation). We briefly review Dirac notation so that the following definitions make formal sense. We refer readers to (Nielsen & Chuang, 2000) for more information.

In Dirac notation, vectors v in a complex vector space $\mathbb{C}^n$ 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 $|i_1, \ldots, i_M\rangle$ with $i_j \in \{0, 1\}$ is interpreted as the vector $|i\rangle \in \mathbb{C}^{2^M}$, 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 Ω be a finite set of size n and $u \in \mathbb{R}^\Omega$ (equivalently $u : \Omega \to \mathbb{R}$) where each entry of u is represented to M bits of precision. 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 $i \in [n]$, where $\bar{u}_i$ is the binary representation of u_i.*

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 Ω be a finite set of size n and $p = (p_x)_{x\in\Omega}$ a discrete probability distribution on Ω. 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 $U_p : |0\rangle \otimes |0\rangle = \sum_{x\in\Omega} \sqrt{p_x}\ |x\rangle \otimes |v_{s'}\rangle$, where $0 \leq J \in \mathbb{Z}$ is arbitrary and $|v_{s'}\rangle \in \mathbb{C}^J$ are 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 $\mathcal{G} : \mathbb{C}^S \otimes \mathbb{C}^A \otimes \mathbb{C}^S \otimes \mathbb{C}^J \to \mathbb{C}^S \otimes \mathbb{C}^A \otimes \mathbb{C}^S \otimes \mathbb{C}^J$ such that*

$$\begin{aligned}&\mathcal{G} : |s\rangle \otimes |a\rangle \otimes |0\rangle \otimes |0\rangle) \\ &\mapsto |s\rangle \otimes |a\rangle \otimes \Big(\sum_{s'\in\mathcal{S}} \sqrt{p(s'|s,a)}\ |s'\rangle \otimes |v_{s'}\rangle \Big),\end{aligned} \tag{7}$$

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

We stress that the quantum state output by $\mathcal{G}$ in Eq. (7) is analogous to a *sample* drawn from the classical probability distribution $\{p(s'|s, a)\}_{s'\in\mathcal{S}}$ as opposed to that distribution fully written out on a piece of paper.

3. Analysis of Quantum Algorithms

In this section, we formally analyze our two algorithms SolveMdp1 and 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 (Brassard et al., 2000; Montanaro, 2015))**.** *There are two quantum algorithms qEst1 and qEst2 with the following specifications. Let Ω be a finite set, $p = (p_x)_{x\in\Omega}$ a discrete probability distribution on Ω, and function $v : \Omega \to \mathbb{R}$. Given quantum oracles U_p and U_v encoding p and v respectively. Then,*

1. *qEst1 requires $u, \epsilon > 0$ as additional inputs and a promise $\mathbf{0} \leq v \leq u$, in which case qEst1 uses $O(u/\epsilon + \sqrt{u/\epsilon})$ queries to U_p, alternatively*
2. *qEst2 requires $\sigma > 0$ and $\epsilon \in (0, 4\sigma)$ as additional inputs and a promise $\mathrm{Var}[v(x) \,|\, x \sim p] \leq \sigma^2$, in which case qEst2 uses $O((\sigma/\epsilon)\log^2(\sigma/\epsilon))$ queries to U_p*

to output an estimate $\hat{\mu}'$ of $\mu := \mathbb{E}[v[x] \,|\, x \sim p] = p^\mathsf{T} v$ satisfying $\Pr(|\hat{\mu}' - \mu| > \epsilon) < 1/3$. Moreover, by repeating one of qEst1 or qEst2 $O(\log(1/\delta))$ times and taking the median output yields another estimate $\hat{\mu}$ of μ satisfying $\Pr(|\hat{\mu} - \mu| < \epsilon) > 1 - \delta$.

For $i \in \{1, 2\}$, we write $\mathsf{qEst}\{\mathsf{i}\}_\delta(p^\mathsf{T} v, \epsilon)$ for an estimate of the mean of $v[x]$, with x distributed as p, to error $< \epsilon$ with probability $> 1 - \delta$, using $\mathsf{qEst}\{\mathsf{i}\}$.

The median-of-means part of Theorem 2 is sometimes referred to as the "powering lemma" (Jerrum et al., 1986).

Theorem 3 (Quantum maximum finding (Dürr & Høyer, 1996))**.** *There exists a universal constant $c_{max} > 0$ such that the following holds. There is a quantum algorithm qArgmax such that, given a quantum oracle U_u encoding a vector $u \in \mathbb{R}^n$, $\mathcal{A}_{\max}$ at most $c_{\max}\sqrt{n}\log(1/\delta)$ queries to U_u and finds $\mathrm{argmax}_i(u_i)$ with probability $> 1 - \delta$.*

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

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:

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

$$v_{k,l} \leq v^{\pi_{k,l}} \leq v^*, \tag{8}$$
$$q_{k,l} \leq q^{\pi_{k,l}} \leq q^*, \tag{9}$$

with probability at least $1-\delta$.

The proof is similar to that found in Section E of (Sidford et al., 2018b); the key point is to show that $v_{k,l} \leq \mathcal{T}^{v_{k,l}}(v_{k,l})$. We present it in Appendix A for completeness.

Next, we prove the following in Appendix A:

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

$$v^* - \epsilon_k \leq v_{k,L}, \tag{10}$$
$$q^* - \epsilon_k \leq q_{k,L}, \tag{11}$$

with probability at least $1-\delta$.

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 (Sidford et al., 2018a).

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*

$$v^* - \epsilon \leq \hat{v} \leq v^{\hat{\pi}} \leq v^*, \tag{12}$$
$$q^* - \epsilon \leq \hat{q} \leq q^{\hat{\pi}} \leq q^*, \tag{13}$$

with probability at least $1-\delta$.

Having shown correctness, we turn to complexity:

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

$$O\big(SA(\Gamma^{1.5}\,\epsilon^{-1} + \Gamma^2)\,\log^4(\Gamma/\epsilon)\log(SA\Gamma/\delta)\big). \tag{14}$$

The proof of Theorem 5 is also in Appendix A and involves showing Theorem 2 is applicable and applying it.

3.3. Analysis of SolveMdp2

If we only require v^* and π^* 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, $\mathsf{qArgmax}_f\{q_{l-1,s}[a] \;:\; a \in \mathcal{A}\}$ on Line 6 which uses the quantum oracle encoding $U_{q_{l-1},s}$ created in the previous iteration $l-1$.

SolveMdp2 is similar to SolveMdp1 but with k fixed to 1 so that there is no outer loop over k. As a result, the correctness and complexity of SolveMdp2 follow similarly to that of SolveMdp1. Therefore, we defer the proofs of the following to Appendix B.

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

$$v^* - \epsilon \leq \hat{v} \leq v^{\hat{\pi}} \leq v^*, \tag{15}$$

with probability at least $1-\delta$.

Theorem 7 (Complexity of SolveMdp2)**.** *The quantum query complexity of SolveMdp2 is*

$$O\big(S\sqrt{A}\,\Gamma^3\,\epsilon^{-1}\,\log^2(\Gamma/\epsilon)\log(SA\Gamma/\delta)\big). \tag{16}$$

4. Lower Bounds

We now state our lower bounds on the number of samples needed to compute q^*, v^*, π^*. 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 (Azar et al., 2012; Sidford et al., 2018a) 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 T 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 $T \geq 10$ (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 γ such that the following lower bounds hold:*

1. *Given access to a classical generative oracle, any algorithm that computes an ϵ-approximation to q^*, v^*, or π^* must make $\Omega(SA\Gamma^3/\epsilon^2)$ queries.*
2. *Given access to a quantum generative oracle, any algorithm that computes an ϵ-approximation to q^* must make $\Omega(SA\Gamma^{1.5}/\epsilon)$ queries and any algorithm that computes an ϵ-approximation to v^* or π^* must make $\Omega(S\sqrt{A}\Gamma^{1.5}/\epsilon)$ queries.*

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 Γ, ϵ, and A in calculating q^*, v^*, and π^*. We show our algorithms are either optimal, or optimal assuming

T or A is constant, for certain ranges of ϵ. We conclude by discussing the open questions left from our work:

1. Can we give optimal algorithms in all parameters (S, A, T, ϵ) for an unrestricted range of ϵ? 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 π^* 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. By assuming the MDP has more structure, there may be greater quantum speedups beyond 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 vectors q^*, v^*, and π^*.
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 (Li et al., 2020). Therefore it is natural to try to construct a quantum model-based algorithm. As mentioned in the Introduction, a first step would be to get a tight bound for distribution learning in ℓ_1-norm.

6. Acknowledgments

We especially thank Wen Sun for suggesting the tabular MDP setting as the first place to search for quantum speedups and for suggesting many useful references, including (Agarwal et al., 2021). We also thank Aaron Sidford, Mengdi Wang, and Xian Wu for helpful discussions on (Sidford et al., 2018b). 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.

References

Agarwal, A., Kakade, S., and Yang, L. F. Model-Based Reinforcement Learning with a Generative Model is Minimax Optimal. In Abernethy, J. and Agarwal, S. (eds.), *Proceedings of the 33rd Conference On Learning Theory (COLT)*, volume 125 of *Proceedings of Machine Learning Research*, pp. 67–83. PMLR, 09–12 Jul 2020.

Agarwal, A., Jiang, N., Kakade, S. M., and Wen, S. *Reinforcement Learning: Theory and Algorithms*. 2021. Book in preparation, draft available at rltheorybook.github.io.

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G. S. L., Buell, D. A., Burkett, B., Chen, Y., Chen, Z., Chiaro, B., Collins, R., Courtney, W., Dunsworth, A., Farhi, E., Foxen, B., Fowler, A., Gidney, C., Giustina, M., Graff, R., Guerin, K., Habegger, S., Harrigan, M. P., Hartmann, M. J., Ho, A., Hoffmann, M., Huang, T., Humble, T. S., Isakov, S. V., Jeffrey, E., Jiang, Z., Kafri, D., Kechedzhi, K., Kelly, J., Klimov, P. V., Knysh, S., Korotkov, A., Kostritsa, F., Landhuis, D., Lindmark, M., Lucero, E., Lyakh, D., Mandrà, S., McClean, J. R., McEwen, M., Megrant, A., Mi, X., Michielsen, K., Mohseni, M., Mutus, J., Naaman, O., Neeley, M., Neill, C., Niu, M. Y., Ostby, E., Petukhov, A., Platt, J. C., Quintana, C., Rieffel, E. G., Roushan, P., Rubin, N. C., Sank, D., Satzinger, K. J., Smelyanskiy, V., Sung, K. J., Trevithick, M. D., Vainsencher, A., Villalonga, B., White, T., Yao, Z. J., Yeh, P., Zalcman, A., Neven, H., and Martinis, J. M. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

Azar, M. G., Munos, R., and Kappen, H. J. On the Sample Complexity of Reinforcement Learning with a Generative Model. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pp. 1707–1714, 2012. ISBN 9781450312851.

Bertsekas, D. *Dynamic Programming and Optimal Control*. Number v. 1 in Athena Scientific optimization and computation series. Athena Scientific, 2000. ISBN 9781886529090.

Bertsekas, D. *Abstract Dynamic Programming*. Athena Scientific, 2013. ISBN 9781886529427.

Brassard, G., Hoyer, P., Mosca, M., and Tapp, A. Quantum Amplitude Amplification and Estimation. 2000. arXiv:quant-ph/0005055

Chakraborty, S., Fischer, E., Matsliah, A., and de Wolf, R. New Results on Quantum Property Testing. In Lodaya, K. and Mahajan, M. (eds.), *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 145–156, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-23-1.

Dong, D., Chen, C., Li, H., and Tarn, T. Quantum Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1207–1220, 2008.

Dunjko, V., Taylor, J. M., and Briegel, H. J. Quantum-Enhanced Machine Learning. *Physical Review Letters*, 117(13), Sep 2016. ISSN 1079-7114.

Dunjko, V., Taylor, J. M., and Briegel, H. J. Advances in quantum reinforcement learning. In *2017 IEEE Inter-*

national Conference on Systems, Man, and Cybernetics (SMC), pp. 282–287, 2017.

Dürr, C. and Høyer, P. A Quantum Algorithm for Finding the Minimum. 1996. arXiv:quant-ph/9607014

Giovannetti, V., Lloyd, S., and Maccone, L. Quantum Random Access Memory. *Physical Review Letters*, 100: 160501, 2008.

Göös, M., Jayram, T. S., Pitassi, T., and Watson, T. Randomized Communication vs. Partition Number. In Chatzigiannakis, I., Indyk, P., Kuhn, F., and Muscholl, A. (eds.), *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 52:1–52:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-041-5.

Grover, L. K. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the 28th ACM Symposium on the Theory of Computing (STOC)*, pp. 212–219, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855.

Jerbi, S., Trenkwalder, L. M., Poulsen Nautrup, H., Briegel, H. J., and Dunjko, V. Quantum Enhancements for Deep Reinforcement Learning in Large Spaces. *PRX Quantum*, 2:010328, Feb 2021.

Jerrum, M. R., Valiant, L. G., and Vazirani, V. V. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169 – 188, 1986. ISSN 0304-3975.

Jin, Y. and Sidford, A. Efficiently Solving MDPs with Stochastic Mirror Descent. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4890–4900. PMLR, 13–18 Jul 2020.

Kakade, S. M. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, 2003.

Kearns, M. and Singh, S. Finite-Sample Convergence Rates for Q-Learning and Indirect Algorithms. In *Advances in Neural Information Processing Systems 11 (NeurIPS)*, pp. 996–1002, 1999. ISBN 0262112450.

Kearns, M., Mansour, Y., and Ng, A. Y. A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. *Machine Learning*, 49(2):193–208, 2002.

Li, G., Wei, Y., Chi, Y., Gu, Y., and Chen, Y. Breaking the Sample Size Barrier in Model-Based Reinforcement Learning with a Generative Model. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33 (NeurIPS)*, volume 33, pp. 12861–12872. Curran Associates, Inc., 2020.

Montanaro, A. Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, 2015.

Nayak, A. and Wu, F. The Quantum Query Complexity of Approximating the Median and Related Statistics. In *Proceedings of the 31st ACM Symposium on the Theory of Computing (STOC)*, pp. 384–393, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 1581130678.

Nielsen, M. A. and Chuang, I. L. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

Paparo, G. D., Dunjko, V., Makmal, A., Martin-Delgado, M. A., and Briegel, H. J. Quantum Speedup for Active Learning Agents. *Physical Review X*, 4:031002, 2014.

Reichardt, B. W. Reflections for Quantum Query Algorithms. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 560–569, USA, 2011. Society for Industrial and Applied Mathematics.

Shor, P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

Sidford, A., Wang, M., Wu, X., Yang, L., and Ye, Y. Near-Optimal Time and Sample Complexities for Solving Markov Decision Processes with a Generative Model. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pp. 5186–5196. Curran Associates, Inc., 2018a.

Sidford, A., Wang, M., Wu, X., and Ye, Y. Variance Reduced Value Iteration and Faster Algorithms for Solving Markov Decision Processes. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, SODA '18, pp. 770–787, USA, 2018b. Society for Industrial and Applied Mathematics. ISBN 9781611975031.

Singh, S. P. and Yee, R. C. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, USA, 2018. ISBN 9780262193986.

Szepesvári, C. Algorithms for Reinforcement Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 2010.

Wainwright, M. J. Variance-reduced Q-learning is minimax optimal, 2019. arXiv:1906.04697

Wang, D., You, X., Li, T., and Childs, A. M. Quantum Exploration Algorithms for Multi-Armed Bandits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35 (11):10102–10110, 2021.

Wang, M. Primal-Dual π Learning: Sample Complexity and Sublinear Run Time for Ergodic Markov Decision Problems, 2017. arXiv:1710.06100

Wang, M. Randomized Linear Programming Solves the Markov Decision Problem in Nearly Linear (Sometimes Sublinear) Time. *Mathematics of Operations Research*, 45(2):517–546, 2020.