

ARTICLE OPEN

When does reinforcement learning stand out in quantum control? A comparative study on state preparation

Xiao-Ming Zhang ^{1,2}, Zezhu Wei¹, Raza Asad³, Xu-Chen Yang ⁴ and Xin Wang ^{1,2*}

Reinforcement learning has been widely used in many problems, including quantum control of qubits. However, such problems can, at the same time, be solved by traditional, non-machine-learning methods, such as stochastic gradient descent and Krotov algorithms, and it remains unclear which one is most suitable when the control has specific constraints. In this work, we perform a comparative study on the efficacy of three reinforcement learning algorithms: tabular Q-learning, deep Q-learning, and policy gradient, as well as two non-machine-learning methods: stochastic gradient descent and Krotov algorithms, in the problem of preparing a desired quantum state. We found that overall, the deep Q-learning and policy gradient algorithms outperform others when the problem is discretized, e.g. allowing discrete values of control, and when the problem scales up. The reinforcement learning algorithms can also adaptively reduce the complexity of the control sequences, shortening the operation time and improving the fidelity. Our comparison provides insights into the suitability of reinforcement learning in quantum control problems.

npj Quantum Information (2019)5:85

; <https://doi.org/10.1038/s41534-019-0201-8>

INTRODUCTION

Reinforcement learning, a branch of machine learning in artificial intelligence, has proven to be a powerful tool to solve a wide range of complex problems, such as the games of Go¹ and Atari.² Reinforcement learning has also been applied to a variety of problems in quantum physics with vast success,^{3–24} including quantum state preparation,^{3–8} state transfer,⁹ quantum gate design,¹⁰ and error correction.¹¹ In many cases, it outperforms commonly used conventional algorithms, such as Krotov and Stochastic Gradient Descent (SGD) algorithms.^{9,10} In the reinforcement learning algorithm, an optimization problem is converted to a set of policies that governs the behavior of a computer agent, i.e. its choices of actions and, consequently, the reward it receives. By simulating sequences of actions taken by the agent maximizing the reward, one finds an optimal solution to the desired problem.²⁵

The development of techniques that efficiently optimize control protocols is key to quantum physics. While some problems can be solved analytically using methods such as reverse engineering,²⁶ in most cases numerical solutions are required. Various numerical methods are therefore put forward, such as gradient-based methods (including SGD,¹⁷ GRAPE²⁷, and variants²⁸), the Krotov method,²⁹ the Nelder-Mead method³⁰ and convex programming.³¹ Recently, there is a frenetic attempt to apply reinforcement learning and other machine-learning-based algorithms^{32–37} to a wide range of physics problems. In particular, the introduction of reinforcement learning to quantum control have revealed new interesting physics,^{3–8} and these techniques have therefore received increasing attention. A fundamental question then arises: under what situation is reinforcement learning the most suitable method?

In this paper, we consider problems related to quantum control of a qubit. The goal of these problems is typically to steer the qubit toward a target state under certain constraints. The mismatch between the final qubit state and the target state naturally serves as the cost function used in the SGD or Krotov methods, and the

negative cost function can serve as the reward function in the reinforcement learning procedure. Our question then becomes: under different scenarios of constraints, which algorithm works best? In this work, we compare the efficacy of two commonly-used traditional methods: SGD and the Krotov method, and three algorithms based on reinforcement learning: tabular Q-learning (TQL),²⁵ deep Q-learning (DQL),² and policy gradient (PG),³⁸ under situations with different types of control constraints.

In ref.,⁵ the Q-learning techniques (TQL and DQL) have been applied to the problem of quantum state preparation, revealing different stages of quantum control. The problem of preparing a desired quantum state from a given initial state is on one hand simple enough to be investigated in full detail, and on the other hand contains sufficient physics allowing for various types of control constraints. We therefore take quantum state preparation as the platform that our comparison of different algorithms is based on. While a detailed description of quantum state preparation is provided in Results, we briefly introduce the five algorithms we are comparing in this work here. (Detailed implementations are provided in the Methods and Supplementary Method 1).

SGD is one of the simplest gradient-based optimization algorithms. In each iteration, a direction in the parameter space is randomly chosen, along which the control field is updated using the gradient of the cost function defined as the mismatch between the evolved state and the target state. Ideally, the gradient is zero when the calculation has converged to the optimal solution. The Krotov algorithm has a different strategy: The initial state is first propagated forward obtaining the evolved state. The evolved state is then projected to the target state, defining a co-state encapsulating the mismatch between the two. Then the co-state is propagated backward to the initial state, during which process the control fields are updated. When the calculation is converged, the co-state is identical to the target state.

¹Department of Physics, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR, China. ²Shenzhen Research Institute, City University of Hong Kong, 518057 Shenzhen, Guangdong, China. ³Department of Mathematics, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR, China. ⁴Department of Physics, The University of Hong Kong, Pokfulam, Hong Kong SAR, China. *email: x.wang@cityu.edu.hk

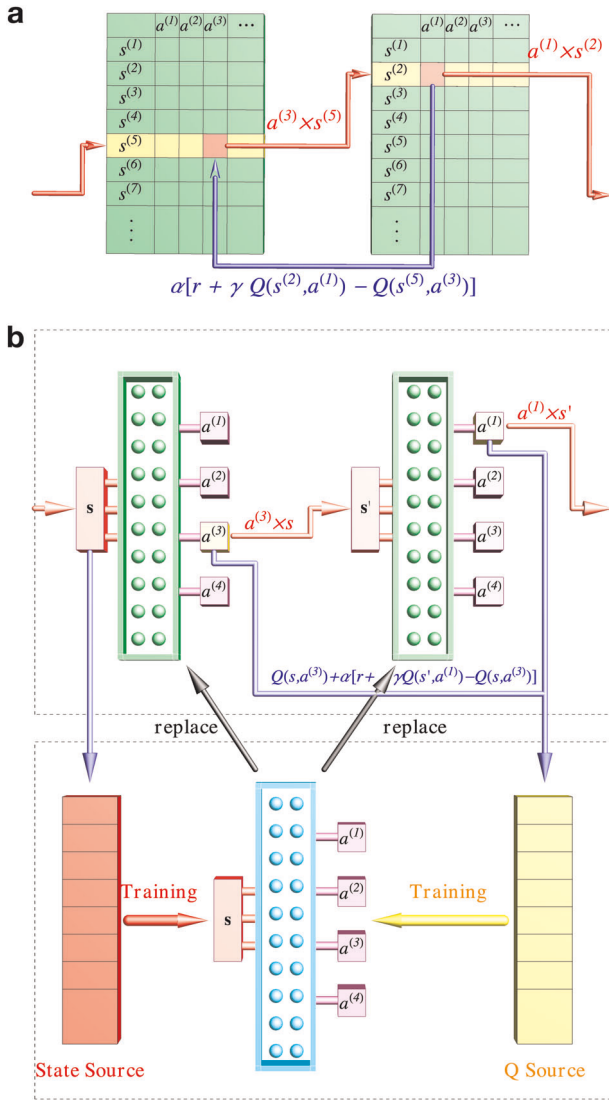


Fig. 1 Sketch of the procedure of TQL and DQL. **a** In TQL, the $Q(s, a)$ values are stored in the Q-table. When the agent is at state $s^{(5)}$, it reviews $Q(s^{(5)}, a^{(l)})$ for all possible actions and chooses one with the maximum “Q-value” (which we assume is $a^{(3)}$). As a result, the state then evolves to $s^{(2)}$. Depending on the distance between $s^{(2)}$ and the target, the Q-values (e.g. $Q(s^{(5)}, a^{(3)})$) is updated according to Eq. (8). This process is then repeated at the new state $s^{(2)}$ and so forth. **b** In DQL, the Q-table is replaced by the Q-network. Instead of choosing an action with the maximum Q-value from a list, this process is done by a neural network, the Q-network, which takes the input state (\mathbf{s}) and outputs an action that it finds most appropriate. Evaluation of the resulting state (\mathbf{s}') after the action suggests how the neural network should be updated (trained). For detailed implementation, see Methods and Supplementary Method 1

In Q-learning (including TQL and DQL), a computer agent evolves in an environment. All information required for optimization is encoded in the environment, which is allowed to be in a set of states \mathcal{S} . In each step, the agent chooses an action from a set \mathcal{A} , bringing the environment of the agent to another state. As a consequence, the agent acquires a reward, which encapsulates the desired optimization problem. Fig. 1a schematically shows how TQL works. At each state $s \in \mathcal{S}$, the agent chooses actions $a \in \mathcal{A}$ according to the action-value function $Q(s, a)$, defined as the estimated total reward starting from state s and action a , forming the so-called Q-table. Each time the agent takes an action,

a reward r is generated according to the distance between the resulting state and the target, which updates the Q-table. An optimal solution is found by iterating this process sufficient times. We note that since a table has a finite number of entries, both the states and actions should be discretized. Figure 1b shows DQL, in which the role of the Q-table is replaced by a neural network, called the Q-network. The agent then chooses its action according to the output of the Q-network, and the reward is used to update the network. In this case, although the allowed actions are typically discrete, the input state can actually be continuous.

Similar to TQL and DQL, PG also requires the sets of states \mathcal{S} , actions \mathcal{A} , and rewards r . The policy of the agent is represented by a neural network. With the state as the input, the network outputs the probability of choosing each action. After each episode, the policy network is updated toward a direction that increases the total reward. Since the state is encoded as the input of the neural network, PG can also accommodate continuous input states.

RESULTS

Single-qubit case

We start with the preparation of a single-qubit state. Consider the time dependent Hamiltonian

$$H[J(t)] = 4J(t)\sigma_z + h\sigma_x, \quad (1)$$

where σ_x and σ_z are Pauli matrices. The Hamiltonian may describe a singlet-triplet qubit³⁹ or a single spin with energy gap h under tunable control fields.^{40,41} In these systems, it is difficult to vary h during gate operations, and we therefore assume that h is a constant in our work, which at the same time serves as our energy unit. Quantum control of the qubit is then achieved by altering $J(t)$ dynamically.

Quantum state preparation refers to the problem to find $J(t)$ such that a given initial state $|\psi_0\rangle$ evolves, within time T , to a final state $|\psi_f\rangle$ that is as close as possible to the target state $|\phi\rangle$. The quality of the state transfer is evaluated using the fidelity, defined as

$$F = |\langle \psi_f | \phi \rangle|^2. \quad (2)$$

We typically use the averaged fidelity \bar{F} over many runs of a given algorithm in our comparison (unless otherwise noted, we average 100 runs to obtain \bar{F}), because the initial guesses of the control sequences are random, and the reinforcement learning procedure is probabilistic.

In this work, we take $|\psi_0\rangle = |0\rangle$, $|\phi\rangle = |1\rangle$ and $T = 2\pi$ unless otherwise specified. Under different situations, there are various kinds of preferences or restrictions of control. We consider the following types of restrictions:

- (i) Assuming that control is performed with a sequence of piecewise constant pulses, and in this work, we further assume that the time duration of each piece is equal to each other for convenience. For this purpose, we divide the total time T into N equal time steps, each of which having a step size $dt = T/N$, with N denoting the maximum number of pieces required by the control. $J(t)$ is accordingly discretized, so that on the i th time step, $J(t) = J_i$ and the system evolves under $H(J_i)$. Denoting the state at the end of the i th time step as $|\psi_i\rangle$, the evolution at the i th step is $|\psi_i\rangle = U_i|\psi_{i-1}\rangle$, where $U_i = \exp\{-iH(J_i)dt\}$. In principle, the evolution time can be less than T , namely the evolution may conclude at the i_f th time step with $i_f \leq N$. (In our calculations, the evolution is terminated when the fidelity $F \geq 0.999$.) Due to their nature, SGD and Krotov have to finish all time steps, i.e. $i_f = N$. However, as we shall see below, QL and DQL frequently have $i_f < N$.
- (ii) We also consider the case where the magnitude of the control field is bounded, i.e. $J_i \in [J_{\max}, J_{\min}]$ for all i . The

constraint can be straightforwardly satisfied in TQL and DQL, since they only operate within the given set of actions thus cannot exceed the bounds. For SGD and Krotov, updates to the control fields may exceed the bounds, in which case we need to enforce the bounds by setting J_i as J_{\max} when the updated value is greater than J_{\max} , and as J_{\min} when the updated value is smaller than J_{\min} . In the case in which either of them is not restricted, we simply note $J_{\min} \rightarrow -\infty$ or $J_{\max} \rightarrow \infty$.

- (iii) The values of the control field may be discretized in the given range, i.e., $J_i \in \{J_{\min}, J_{\min} + dJ/M, J_{\min} + 2dJ/M, \dots, J_{\max}\}$ where $dJ = (J_{\max} - J_{\min})$, so that the control field can take $M + 1$ values including J_{\min} and J_{\max} . In reality this situation may arise, for example, when decomposing a quantum operation into a set of given gates.^{42–44} For a reason similar to (ii), TQL and DQL only select actions within the given set so the constraint is satisfied. For SGD and Krotov which keep updating the values of the control field during iterations, we enforce the constraint by setting the value of each control field to the nearest allowed value at the end of the execution.

To sum up, the number of pieces in control sequences N , the bounds of the control field J_{\min} and J_{\max} , as well as the number of the discrete values of the control field $M + 1$ are the main factors characterizing situations to prepare quantum states, based on which our comparison of different algorithms is conducted. We also define N^{iter} as the number of iterations performed in executing an algorithm, which is typically taken as equal for different algorithms to ensure a fair comparison. Unless otherwise noted, $N^{\text{iter}} = 500$ in all results shown.

In Fig. 2 we study a situation where the maximum number of pieces in the control sequence N is given, and the results are shown as the averaged fidelities as functions of N . Here, the quality of an algorithm is assessed by the averaged fidelity of the state it prepares (as compared to the target state) \bar{F} , but not by the computational resources it costs. For $N \leq 10$, the Krotov method gives the lowest fidelity, possibly due to the fact that Krotov requires a reasonable level of continuity in the control sequence, and one with a few pieces is unlikely to reach convergence. As N increases, the performance of Krotov is much improved, which has the highest fidelity when N is large ($N \geq 30$ as seen in the figure). SGD performs better than Krotov for $N \leq 10$, but worse otherwise, because as N increases, the algorithm has to search over a much larger parameter space. Within the given number of iterations

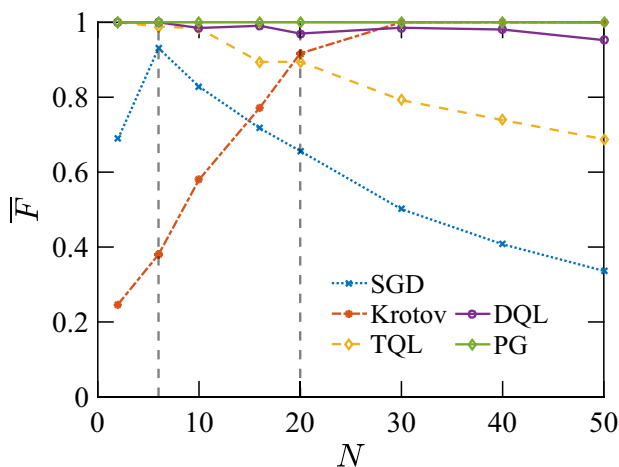


Fig. 2 Average fidelities as functions of the maximum number of control pieces. For TQL, DQL, and PG, $J_i \in \{0, 1\}$ (i.e. $M = 1$). For SGD and Krotov, no restriction is imposed on the range of J_i and M (i.e. $M \rightarrow \infty$). The vertical dashed lines correspond to results shown with respective M values in Fig. 4

($N^{\text{iter}} = 500$ as noted above), it concludes with a lower fidelity. Of course, this result can be improved if more iterations are allowed, and we shall show relevant results in Supplementary Discussion 2. The SGD results at $N = 2$ is irregular (thus the cusp at $N = 6$), due to the lack of flexibility in the control sequence which makes it difficult to achieve high fidelity with only two steps.

The fidelity for TQL is higher than SGD and Krotov, but is still lower than that of DQL and PG, indicating the superior ability of deep learning. Nevertheless, we note that the TQL may sometimes fail: it occasionally arrives at a final state which is completely different than the target state. On the other hand, SGD could fail by being trapped at a local minimum, but even in that case it is not drastically different from the optimal solution in terms of the fidelity. This is the reason why the TQL results drop for $N > 10$. For larger N , the failure rate for TQL is higher (possibly due to the higher dimensionality of the Q-table), and therefore the averaged fidelity is lower. Among all five algorithms, PG is consistently the best. Apart from PG, DQL gives the highest fidelity for $N < 30$, but due to its nonzero failure probability, it is outperformed by Krotov for $N > 30$. Nevertheless, the effect is moderate and the fidelity is still very close to 1 ($\bar{F} = 0.9988$).

To further understand the results shown in Fig. 2, we take examples from $N = 20$ and plot the pulse profiles and the corresponding trajectories on the Bloch sphere in Fig. 3. We immediately realize that reinforcement learning (TQL, DQL, and PG) yield very simple pulse shapes: one only has to keep the control at zero for time $T/2$, and the desired target state ($|1\rangle$) will

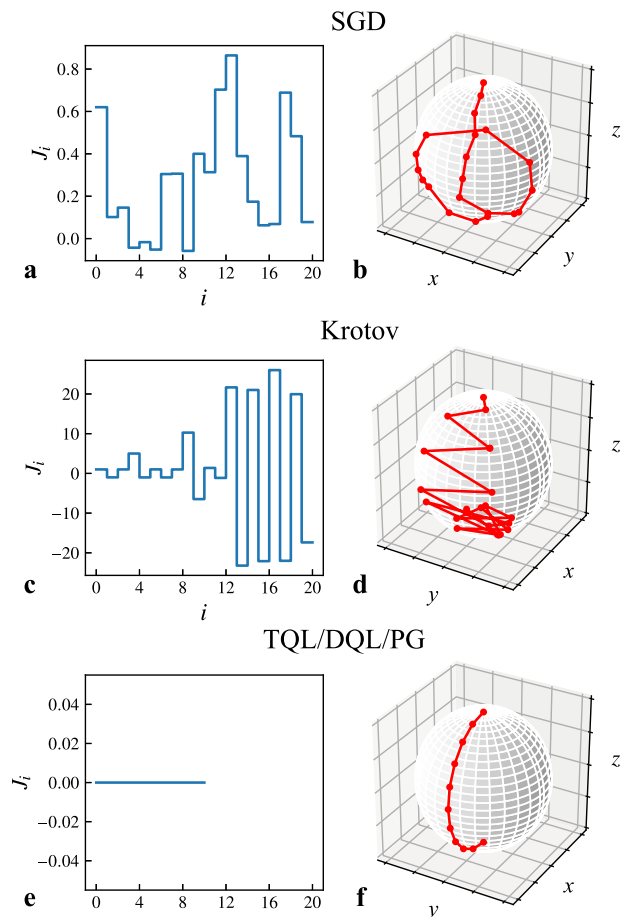


Fig. 3 Pulse profiles and the corresponding trajectories on the Bloch sphere. **a, c, e** Example pulse profiles taken from results of Fig. 2 with $N = 20$. **b, d, f** Evolution of the state corresponding to the respective control sequence in the left column. TQL, DQL, and PG give the same optimal results and are thus shown together

be achieved. However, to find the result, the algorithm has to somehow realize that one does not have to complete all N pieces, which implies their ability to adaptively generating the control sequence. As can be seen from Fig. 3a, c, SGD and Krotov only search for pulse sequences with exactly N pieces and therefore miss the optimal solution. Their trajectories on the Bloch sphere are much more complex as compared to those of reinforcement learning. In practice, the complex pulse shapes and longer gate times mean that they are difficult to realize in the laboratory, and potentially introduces error to the control procedure (In Supplementary Discussion 4 we provide more details on this issue). From Fig. 3 we also notice that reinforcement learning possesses better ability to adaptively sequencing, which is particularly suitable for problems that involve optimization of gate time or speed, such as the quantum speed limit.^{5,9} On the other hand, application of SGD or Krotov to the same problem requires searching over various different N values before an optimal solution can be found, which cost much more resources.^{45,46}

We now study the effect of restrictions on the performances of algorithms. Namely, the control field is bounded between J_{\min} and J_{\max} , with $M + 1$ allowed values including the bounds. In Fig. 4, we impose the same restriction $J \in [0, 1]$ to all five methods and vary M from $M = 1$ to $M = 49$. It is interesting to note that the averaged fidelities of three reinforcement learning algorithms decreases with M , albeit not considerably. This is because TQL, DQL and PG favor bounded and concrete sets of actions, and more choices will only add burden to the searching process, rendering the algorithms inefficient. Improvements may be made by increasing the number of iterations (cf. Supplementary Discussion 2), and using a larger neural network with stronger representational power. For $N = 6$ (Fig. 4a), TQL and DQL are comparable and have overall the best performance except for $M > 14$ in which SGD becomes slightly better. On the other hand, \bar{F} for PG drops rapidly for $M \geq 30$. For $N = 20$ (Fig. 4b), DQL and PG have the best performance, but for large M they are not significantly better than other methods. More results involving SGD and Krotov are given in Supplementary Discussion 1, from which we conclude that the effect of boundaries in control is

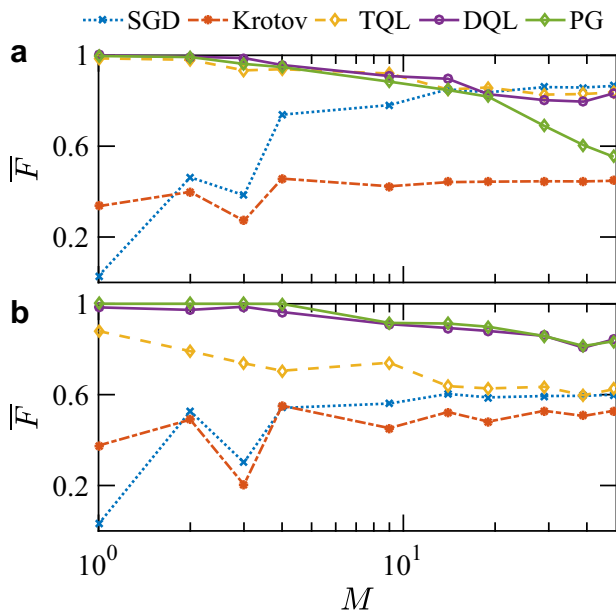


Fig. 4 Effect of discrete control fields on the averaged fidelity for all five methods considered. The strength of control field is restricted to $J_i \in [0, 1]$, and $M + 1$ discrete values (including 0 and 1) are allowed. **a** shows the case of $N = 6$ while **b** $N = 20$, corresponding to the two vertical dashed lines in Fig. 2

much more obvious for Krotov method than SGD, since Krotov performs much larger updates at each iteration. Meanwhile, the effect of discretization (decreasing M) are severe for both Krotov and SGD methods, indicating that successful implementations of them depend crucially on the continuity of the problem.

Finally, we note that all results obtained have the target state being $|1\rangle$. Preparing a quantum state other than $|1\rangle$ may have different results, for which an example is presented in Supplementary Discussion 3. Nevertheless, the overall observation of the pros and cons of the algorithms should remain similar.

Multi-qubit case

We now consider a case preparing a multi-qubit state as sketched in Fig. 5a, b. Our system is described by the following Hamiltonian:

$$H(t) = C \sum_{k=1}^{K-1} (S_x^k S_x^{k+1} + S_y^k S_y^{k+1}) + \sum_{k=1}^K 2B_k(t) S_z^k, \quad (3)$$

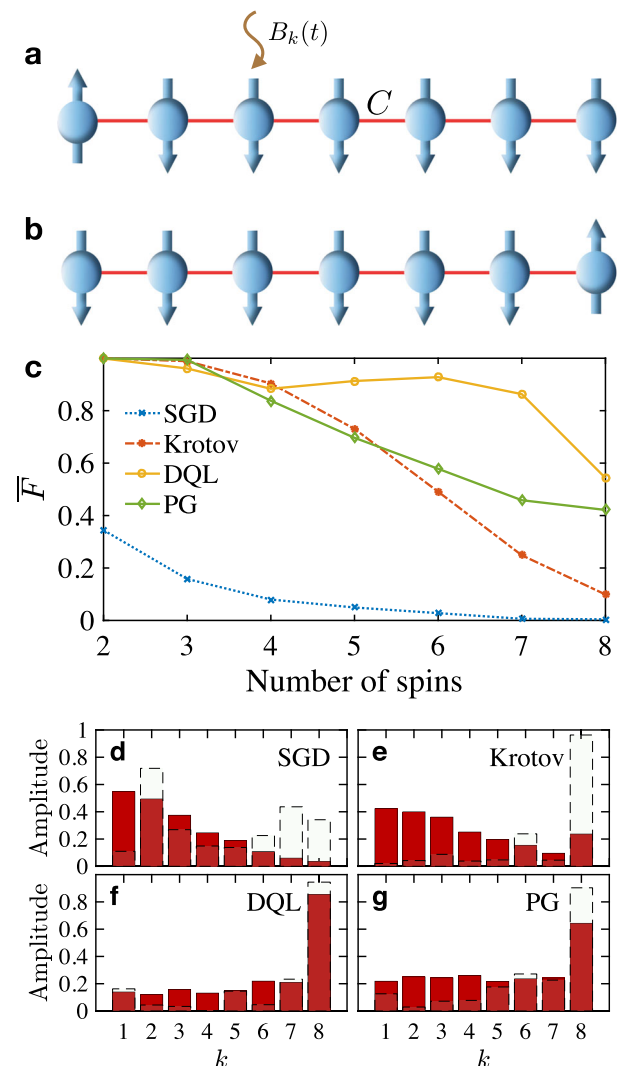


Fig. 5 Spin transfer as preparation of a multi-qubit state. **a** The multi-qubit system is initialized with the leftmost spin being up and all others down. **b** The target state has the rightmost spin being up and all others down. **c** The average fidelity versus the number of spins from different algorithms; **d–g** The amplitudes (visualization of the final prepared state) at different spins for $K = 8$. The red solid bars correspond to results averaged over 100 runs, and the hollow bars enclosed by dashed line shows the results with the highest fidelities

Table 1. Summary of the performances under different situations

	SGD	Krotov	TQL	DQL	PG
Performance vs number of time steps N	↘	↗	↘	↘	*
Ability to adaptively segment			*	*	*
Discrete operation set (M small)				*	*
Continuous operation set (M large)		*			
Scaled-up problems (multi-qubits)				*	

A "*" indicates that the algorithm performs best, while the arrow "↘" ("↗") denotes decrease (increase) of the performance versus increase of the variable concerned

where K is the total number of spins, S_x^k , S_y^k and S_z^k are the k th spin operator, C describes the constant nearest-neighbor coupling strength (set to be $C = 1$), and $B_k(t)$ is the time-dependent local magnetic field applied at the k th spin to perform control. This is essentially a task transferring a spin: the system is initialized to a state with the leftmost spin being up and all others down, and the goal is to prepare a state with the rightmost spin being up and all others down. We set the operation time duration to be $T = (K - 1)\pi/2$, which is divided to 20 equal time steps (i.e. $N = 20$). The external field is restricted to $B_k(t)/C \in [0, 40]$ for SGD and Krotov, and $B_k(t)/C \in \{0, 40\}$ for all three reinforcement learning algorithms. Note that TQL fails for $K \geq 2$ due to the large size of the Q-table, and is thus excluded in the comparison.

Figure 5c shows the average fidelity versus the number of spins (K), after each algorithm is run for 500 iterations. As K increases, the dimensionality of the problem increases and therefore the performances of all algorithms deteriorate. When $K < 4$, Krotov, DQL and PG have comparable performances, while SGD has the lowest fidelity. As K increases, \bar{F} for PG and DQL drop much more slowly as compared to Krotov. At $K = 8$, we have $\bar{F} = 0.0989$ (Krotov), $\bar{F} = 0.4214$ (PG), $\bar{F} = 0.5433$ (DQL), respectively. Here, we have not assumed a particular form of the control field, so one has to search over a very large space. Specializing the control to certain types would improve performances of the algorithms.⁴⁷

In order to visualize the final states prepared, we define the amplitude, A^k , as the absolute value of the inner product between the final states and the state with the k th spin being up while all others being down. A perfect transfer would be that the amplitude is 1 for the rightmost spin and 0 otherwise. Taking $K = 8$ as an example, we show how the amplitudes distribute over different spins in Fig. 5d–g. We compare two different kinds of results: one showing the averaged results over 100 runs (shown as red solid bars), and the other the best result among the 100 runs (hollow bars enclosed by dashed lines). We see that SGD completely fails to prepare the desired state. The best results from Krotov, DQL and PG are comparable, but considering the average over many runs, DQL and PG have better performances. Moreover, the optimal control sequences for different algorithms are provided in Supplementary Tables 1–4.

DISCUSSION

In this paper, we have examined performances of five algorithms: SGD, Krotov, TQL, DQL, and PG, on the problem of quantum state preparation. From the comparison, we can summarize the characteristics of the algorithms under different situations as follows (see also Table 1).

Dependence on the maximum number of pieces in the control sequence, N : When all algorithms are executed with the same number of iterations, PG has overall the best performance, but the corresponding fidelity still drops slightly as N increases. In fact, the fidelities from all methods decrease as N increases, except the Krotov method, for which the fidelity increases when N is large.

Ability to adaptively segment: During the optimization process, TQL, DQL, and PG can adaptively reduce the number of pieces required and can thus find optimal solutions efficiently. SGD and Krotov, on the other hand, always work with a fixed number of N and thus sometimes miss the optimal solution.

Dependence on restricted ranges of the strength of the control field: TQL, DQL, and PG naturally work with restricted sets of actions so they perform well when the strength of the control field is restricted. Such restriction reduces the efficiency for both SGD and Krotov method, but the effect is moderate for SGD because its updates on the control field are essentially local. However, the Krotov method makes significant updates during its execution and thus becomes severely compromised when the strength of the control field is restricted.

Ability to work with control fields taking $M + 1$ discrete values: TQL, DQL, and PG again naturally work with discrete values of the control field. In fact, the fidelities from them decrease as the allowed values of the control fields become more continuous (M increases). This problem may be circumvented using more sophisticated algorithms such as Actor-Critic,^{48,49} and the deep deterministic policy gradient method.⁵⁰ SGD is not sensitive to M because it works with a relatively small range of control field and a reasonable discretization is sufficient. The Krotov method, on the other hand, strongly favors continuous problem, i.e. M being large.

Ability to accommodate scaled-up problems (multiple qubits): Except for TQL, all other algorithms can be straightforwardly generalized to treat quantum control problems with more than one qubit. However, SGD is rather inefficient, and DQL generally outperforms all others for cases considered in this work ($K \leq 8$).

Moreover, we have found that PG and DQL methods, in general, have the best performances among the five algorithms considered, demonstrating the power of reinforcement learning in conjunction with neural networks in treating complex optimization problems.

Our direct comparison of different methods may also shed light on how these algorithms can be improved. For example, the Krotov method strongly favors the "continuous" problem, for which TQL, DQL, and PG do not perform well. It should be possible that gradients in the Krotov method can be applied in the Q-learning procedures and thereby improves their performances. We hope that our work has elucidated the effectiveness of reinforcement learning in problems with different types of constraints, and in addition, it may provide hints on how these algorithms can be improved in future studies.

METHODS

In this section, we give a brief description of our implementation of TQL, DQL, and PG in this work. The full algorithms for all methods used in this work are given in Supplementary Method 1.

TQL

For Q-learning, the key ingredients include a set of allowed states S , a set of actions \mathcal{A} , and the reward r . The state of qubit can be parametrized as

$$|\psi(\theta, \varphi)\rangle = \pm \left(\cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle \right), \quad (4)$$

where (θ, φ) corresponds to a point on the Bloch sphere, and a possible global phase of -1 has been included. Our set of allowed states is defined as

$$S \equiv \{ |\psi(\theta, \varphi)\rangle | \theta \in s_\theta, \varphi \in s_\varphi \}, \quad (5)$$

where

$$s_\theta = \left\{ \frac{0\pi}{30}, \frac{1\pi}{30}, \dots, \frac{29\pi}{30} \right\}, \quad s_\varphi = \left\{ \frac{0\pi}{30}, \frac{1\pi}{30}, \dots, \frac{59\pi}{30} \right\}. \quad (6)$$

We note that this is a discrete set of states, and after each step in the evolution, if the resulting state is not identical to any of the member in the set, it will be assigned as the member that is closest to the state, i.e. having the maximum fidelity in their overlap.

In the i th step of the evolution, the system is at a state $s_i = |\psi_i\rangle \in \mathcal{S}$, and the action is given by the evolution operator $a_i = U_i = \exp\{-iH(J_i)dt\}$. All allowed values of the control field J_i therefore form a set of possible actions \mathcal{A} . The resulting state $U_i|\psi_i\rangle$ after this step is then compared to the target state, and the reward is calculated using the fidelity between the two states as

$$r_i = \begin{cases} 10 & F \in (0.5, 0.9], \\ 100 & F \in (0.9, 0.999], \\ 5000 & F \in (0.999, 1], \end{cases} \quad (7)$$

so that the action that takes the state very close to the target is strongly rewarded. In practice, the agent chooses its action according to the ϵ -greedy algorithm,²⁵ i.e. the agent either chooses an action with the largest $Q(s, a)$ with $1 - \epsilon$ probability, or with probability ϵ it randomly chooses an action in the set. The introduction of a nonzero but small ϵ ensures that the system is not trapped in a poor local minimum. The elements in Q-tables are then updated as:

$$Q(s_{i-1}, a_i) \leftarrow Q(s_{i-1}, a_i) + \alpha[r_i + \gamma \max_{a'} Q(s_i, a') - Q(s_{i-1}, a_i)], \quad (8)$$

where a' refers to all possible a_i in this step, α is the learning rate, and γ is a reward discount to ensure the stability of the algorithm.

DQL

DQL stores the action-value functions with a neural network Θ . We take qubit case as an example. Defining an agent state as

$$\mathbf{s} = [\text{Re}(\langle 0|\psi\rangle), \text{Im}(\langle 0|\psi\rangle), \text{Re}(\langle 1|\psi\rangle), \text{Im}(\langle 1|\psi\rangle)]^T, \quad (9)$$

the network outputs the Q-value for each action $a \in \mathcal{A}$ as $Q(\mathbf{s}, a; \Theta)$. We note that in DQL, the discretization of states on the Bloch sphere is no longer necessary and we can deal with states that vary continuously. Otherwise the definitions of the set of actions and reward are the same as those in TQL.

We adopt the double Q-network training approach:² two neural networks, the evaluation network Θ and the target network Θ^- , are used in training. In the memory we store experiences defined as $e_i = (\mathbf{s}_{i-1}, a_i, r_i, \mathbf{s}_i)$. In each training step, an experience is randomly chosen from the memory, and the evaluation network is updated using the outcome derived from the experience.

PG

Similar to DQL, PG is based on neural networks. With the state \mathbf{s} as the input vector, the network of PG outputs the probability of choosing each action $\mathbf{p} = P(\mathbf{s}; \Theta)$, where $\mathbf{p} = [p_1, p_2, \dots]^T$. At each time step t , the agent chooses its action according to \mathbf{p} , and stores the total reward it has obtained $v_t = \sum_{i=1}^t \gamma^i r_i$. In each iteration, the network is updated in order to increase the total reward. This is done according to the gradient of $\log P(\mathbf{s}_t; \Theta) v_t$, the details of which can be found in Supplementary Method 1.

We note that unlike the case for SGD and Krotov, in which the fidelity monotonically increases with more training in most cases, the fidelity output by TQL, DQL, and PG may experience oscillations as the algorithm cannot guarantee optimal solutions in all trials. In this case, one just has to choose outputs which have higher fidelity as the learning outcome.

DATA AVAILABILITY

The data generated during this study are available from the corresponding author upon reasonable request.

CODE AVAILABILITY

The code for all algorithms used in this work is available on GitHub under MIT License (https://github.com/93xiaoming/RL_state_preparation).

Received: 15 February 2019; Accepted: 10 September 2019;

Published online: 08 October 2019

REFERENCES

- Silver, D. et al. Mastering the game of go without human knowledge. *Nature* **550**, 354–359 (2017).
- Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
- Chen, C., Dong, D., Li, H.-X., Chu, J. & Tarn, T.-J. Fidelity-based probabilistic q-learning for control of quantum systems. *IEEE transactions on neural networks and learning systems* **25**, 920–933 (2013).
- Chen, J.-J. & Xue, M. Manipulation of spin dynamics by deep reinforcement learning agent. Preprint at <https://arxiv.org/abs/1901.08748> (2019).
- Bukov, M. et al. Reinforcement learning in different phases of quantum control. *Phys. Rev. X* **8**, 031086 (2018).
- Bukov, M. Reinforcement learning for autonomous preparation of floquet-engineered states: Inverting the quantum Kapitza oscillator. *Phys. Rev. B* **98**, 224305 (2018).
- August, M. & Hernández-Lobato, J. M. Taking gradients through experiments: Lstms and memory proximal policy optimization for black-box quantum control. Preprint at <https://arxiv.org/abs/1802.04063> (2018).
- Albarrán-Arriagada, F., Retamal, J. C., Solano, E. & Lamata, L. Measurement-based adaptation protocol with quantum reinforcement learning. *Phys. Rev. A* **98**, 042315 (2018).
- Zhang, X.-M., Cui, Z.-W., Wang, X. & Yung, M.-H. Automatic spin-chain learning to explore the quantum speed limit. *Phys. Rev. A* **97**, 052333 (2018).
- Niu, M. Y., Boixo, S., Smelyanskiy, V. N. & Neven, H. Universal quantum control through deep reinforcement learning. *npj Quantum Information* **5**, 33 (2019).
- Fösel, T., Tighineanu, P., Weiss, T. & Marquardt, F. Reinforcement learning with neural networks for quantum feedback. *Phys. Rev. X* **8**, 031084 (2018).
- Melnikov, A. A. et al. Active learning machine learns to create new quantum experiments. *Proc. Natl Acad. Sci. USA* **115**, 1221–1226 (2018).
- Dunjko, V., Taylor, J. M. & Briegel, H. J. Quantum-enhanced machine learning. *Phys. Rev. Lett.* **117**, 130501 (2016).
- Hentschel, A. & Sanders, B. C. Machine learning for precise quantum measurement. *Phys. Rev. Lett.* **104**, 063603 (2010).
- Day, A. G., Bukov, M., Weinberg, P., Mehta, P. & Sels, D. Glassy phase of optimal quantum control. *Phys. Rev. Lett.* **122**, 020601 (2019).
- Wu, R.-B., Chu, B., Owens, D. H. & Rabitz, H. Data-driven gradient algorithm for high-precision quantum control. *Phys. Rev. A* **97**, 042122 (2018).
- Ferrie, C. Self-guided quantum tomography. *Phys. Rev. Lett.* **113**, 190404 (2014).
- Reddy, G., Celani, A., Sejnowski, T. J. & Vergassola, M. Learning to soar in turbulent environments. *Proc. Natl Acad. Sci. USA* **113**, E4877–E4884 (2016).
- Colabrese, S., Gustavsson, K., Celani, A. & Biferale, L. Flow navigation by smart microswimmers via reinforcement learning. *Phys. Rev. Lett.* **118**, 158004 (2017).
- Nautrup, H. P., Delfosse, N., Dunjko, V., Briegel, H. J. & Friis, N. Optimizing quantum error correction codes with reinforcement learning. Preprint at <https://arxiv.org/abs/1812.08451> (2018).
- Sweke, R., Kesselring, M. S., van Nieuwenburg, E. P. & Eisert, J. Reinforcement learning decoders for fault-tolerant quantum computation. Preprint at <https://arxiv.org/abs/1810.07207> (2018).
- Andreasson, P., Johansson, J., Liljestränd, S. & Granath, M. Quantum error correction for the toric code using deep reinforcement learning. *Quantum* **3**, 183 (2019).
- Halverson, J., Nelson, B. & Ruehle, F. Branes with brains: Exploring string vacua with deep reinforcement learning. *J. High Energy Phys.* **2019**, 3 (2019).
- Zhao, K.-W., Kao, W.-H., Wu, K.-H. & Kao, Y.-J. Generation of ice states through deep reinforcement learning. *Phys. Rev. E* **99**, 062106 (2019).
- Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press Cambridge, 1998).
- Barnes, E. & Das Sarma, S. Analytically solvable driven time-dependent two-level quantum systems. *Phys. Rev. Lett.* **109**, 060401 (2012).
- Khaneja, N., Reiss, T., Kehlet, C., Schulte-Herbrüggen, T. & Glaser, S. J. Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms. *J. Magn. Reson.* **172**, 296–305 (2005).
- Jäger, G., Reich, D. M., Goerz, M. H., Koch, C. P. & Hohenester, U. Optimal quantum control of bose-einstein condensates in magnetic microtraps: Comparison of gradient-ascent-pulse-engineering and krotov optimization schemes. *Phys. Rev. A* **90**, 033628 (2014).
- Krotov, V. F. *Global methods in optimal control theory*. (Marcel Dekker Inc., New York, 1996).
- Kelly, J. et al. Optimal quantum control using randomized benchmarking. *Phys. Rev. Lett.* **112**, 240504 (2014).
- Kosut, R. L., Grace, M. D. & Brif, C. Robust control of quantum gates via sequential convex programming. *Phys. Rev. A* **88**, 052326 (2013).
- Wang, L. Discovering phase transitions with unsupervised learning. *Phys. Rev. B* **94**, 195105 (2016).
- Carleo, G. & Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **355**, 602–606 (2017).
- Deng, D.-L., Li, X. & Das Sarma, S. Quantum entanglement in neural network states. *Phys. Rev. X* **7**, 021021 (2017).
- Carrasquilla, J. & Melko, R. G. Machine learning phases of matter. *Nat. Phys.* **13**, 431–434 (2017).

36. Li, J., Yang, X., Peng, X. & Sun, C.-P. Hybrid quantum-classical approach to quantum optimal control. *Phys. Rev. Lett.* **118**, 150503 (2017).
37. Hsu, Y.-T., Li, X., Deng, D.-L. & Das Sarma, S. Machine learning many-body localization: search for the elusive nonergodic metal. *Phys. Rev. Lett.* **121**, 245701 (2018).
38. Sutton, R. S., McAllester, D. A., Singh, S. P. & Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inf. Process. Syst.* **12**, 1057–1063 (2000).
39. Petta, J. R. et al. Coherent manipulation of coupled electron spins in semiconductor quantum dots. *Science* **309**, 2180–2184 (2005).
40. Greilich, A. et al. Ultrafast optical rotations of electron spins in quantum dots. *Nat. Phys.* **5**, 262–266 (2009).
41. Poem, E. et al. Optically induced rotation of an exciton spin in a semiconductor quantum dot. *Phys. Rev. Lett.* **107**, 087401 (2011).
42. Kitaev, A. Y., Shen, A. & Vyalys, M. N. Classical and quantum computation. American Mathematical Society (2002).
43. Harrow, A. W., Recht, B. & Chuang, I. L. Efficient discrete approximations of quantum gates. *Quantum Phys.* **43**, 4445–4451 (2002).
44. Campbell, E. T., Terhal, B. M. & Vuillot, C. Roads towards fault-tolerant universal quantum computation. *Nature* **549**, 172–179 (2017).
45. Caneva, T. et al. Optimal control at the quantum speed limit. *Phys. Rev. Lett.* **103**, 240501 (2009).
46. Murphy, M., Montangero, S., Giovannetti, V. & Calarco, T. Communication at the quantum speed limit along a spin chain. *Phys. Rev. A* **82**, 022318 (2010).
47. Zhang, Y. & Kim, E.-A. Quantum loop topography for machine learning. *Phys. Rev. Lett.* **118**, 216401 (2017).
48. Mnih, V. et al. Asynchronous methods for deep reinforcement learning. *Int. Conf. Mach. Learn.* **48**, 1928–1937 (2016).
49. Xu, H. et al. Transferable control for quantum parameter estimation through reinforcement learning. Preprint at <https://arxiv.org/abs/1904.11298> (2019).
50. Lillicrap, T. P. et al. Continuous control with deep reinforcement learning. Preprint at <https://arxiv.org/abs/1509.02971> (2015).

ACKNOWLEDGEMENTS

This work is supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (Grant Nos. CityU 21300116, CityU 11303617, CityU 11304018), the National Natural Science Foundation of China (Grant Nos. 11874312, 11604277), the Guangdong Innovative and Entrepreneurial Research Team Program (Grant No. 2016ZT06D348), and Key R&D Program of Guangdong province (Grant No. 2018B030326001).

AUTHOR CONTRIBUTIONS

X.W. and X.-M.Z. conceived the project, X.-M.Z., Z.W., R.A. and X.-C.Y. performed calculations. All authors discussed the results and implications at all stages and wrote the paper.

COMPETING INTERESTS

The authors declare no competing interests.

ADDITIONAL INFORMATION

Supplementary Information is available for this paper at <https://doi.org/10.1038/s41534-019-0201-8>.

Correspondence and requests for materials should be addressed to X.W.

Reprints and permission information is available at <http://www.nature.com/reprints>

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



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

© The Author(s) 2019