

ARTICLE OPEN

Global optimization of quantum dynamics with AlphaZero deep exploration

Mogens Dalgaard¹, Felix Motzoi¹, Jens Jakob Sørensen¹ and Jacob Sherson^{1*}

While a large number of algorithms for optimizing quantum dynamics for different objectives have been developed, a common limitation is the reliance on good initial guesses, being either random or based on heuristics and intuitions. Here we implement a *tabula rasa* deep quantum exploration version of the Deepmind AlphaZero algorithm for systematically averting this limitation. AlphaZero employs a deep neural network in conjunction with deep lookahead in a guided tree search, which allows for predictive hidden-variable approximation of the quantum parameter landscape. To emphasize transferability, we apply and benchmark the algorithm on three classes of control problems using only a single common set of algorithmic hyperparameters. AlphaZero achieves substantial improvements in both the quality and quantity of good solution clusters compared to earlier methods. It is able to spontaneously learn unexpected hidden structure and global symmetry in the solutions, going beyond even human heuristics.

npj Quantum Information (2020)6:6 ; <https://doi.org/10.1038/s41534-019-0241-0>

Recent progress on technologies with quantum speedup focuses largely on optimizing dynamical quantum cost functionals via a set of external classical parameters. Such research includes quantum variational eigensolvers,¹ annealers,² simulators,^{3,4} circuit optimization,^{5,6} optimal control theory,^{7–9} and Boltzmann machines.¹⁰ The minimized functional could be for example the energy of a simulated system, or the distance to a quantum computational gate.

A shared algorithmic feature is domain knowledge about where to search, such as near the Hartree–Fock Ansatz for variational eigensolvers, or in the analytical gradient direction. An open question in optimization research is how much this specialized approach can be supplanted by a problem-agnostic methodology: One which does not require expert knowledge, avoiding both the overhead in human labor¹¹ and the potential for local, suboptimal trapping.^{12–14} In other words, an autonomous machine learning approach has the potential to plan its solutions strategically.

It has been argued that, due to the inherent smoothness of unitary quantum physics,¹⁵ local *exploitation* of quantum dynamics can be sufficient for efficiently finding good solutions.¹⁶ Local search has been especially successful in the well-established field of Quantum Optimal Control Theory (QOCT), enjoying a half century of continued progress in NMR,¹⁷ quantum chemistry,^{7,18} and spectroscopy.¹⁹ This has culminated in Hessian extraction approaches²⁰ that generally outperform other local methods.^{21,22}

Yet, similar to classical NP-complete problems,²³ quantum functionals can suffer a phase transition²⁴ from easier to “needle in a haystack” instances that require global *exploration* of parameters.

Mounting evidence has shown that imposing significant constraints in the dynamics may lead to such complexity,^{11,24–26} especially as QOCT has veered into high-precision quantum computation,²⁷ circuit compilation,²⁸ and architecture design.²⁹ It is therefore crucial to balance resources for exploitation of smooth, local quantum landscapes with state-of-the-art classical methods for domain-agnostic exploration.

In the literature, optimization of dynamically evolving systems is characterized by a lookahead-depth, i.e. how far into the future one plans current actions. A shallow depth may broaden exploration, a strategy typically found in Reinforcement Learning (RL).³⁰ This has been powerfully combined with Deep Neural Networks (DNN)^{31–35} and applied recently to quantum systems.^{36–43} Unfortunately, single-step lookaheads are inherently local and thus require a slower learning rate, with no performance gain found over full-depth, domain-specialized (Hessian approximation) methods in QOCT. Other full-depth methods have also had mixed success, e.g. Genetic Algorithms^{44,45} and Differential Evolution,²⁵ but they typically require careful fine-tuning since they are based on ad hoc heuristics rather than being mathematically rooted.

A recent stunning breakthrough has been due to the AlphaZero class of algorithms.^{46–48} AlphaZero has already effectively out-classed all adversaries in the games of Go, Chess, Shogi, and Starcraft. The key to the success of AlphaZero was the combination of a Monte Carlo tree search with a one-step lookahead DNN. As a result, the lookahead information from far down the tree dramatically increases the trained DNN precision, and together they compound to produce much more focused and heuristic-free exploration.

Here, we implement and benchmark a QOCT version of AlphaZero for optimizing quantum dynamics. We characterize improvements in learning and exploration compared to traditional methods. We find a crossover from difficult problems where AlphaZero learning alone is ideal and those where a combination of deep exploration and quantum-specialized exploitation is optimal. We show this leads to a dramatic increase in both the quality and quantity of good solution clusters. Our AlphaZero implementation retains the *tabula rasa* character of ref. ⁴⁷ in two important respects. Firstly, it efficiently learns to solve three different optimization problem classes using the same algorithmic hyperparameters. Secondly, we demonstrate that AlphaZero is able to identify quantum-specific heuristics in the form of hidden symmetries without the need for expert knowledge.

¹Department of Physics and Astronomy, Aarhus University, Ny Munkegade 120, 8000 Aarhus C, Denmark. *email: sherson@phys.au.dk

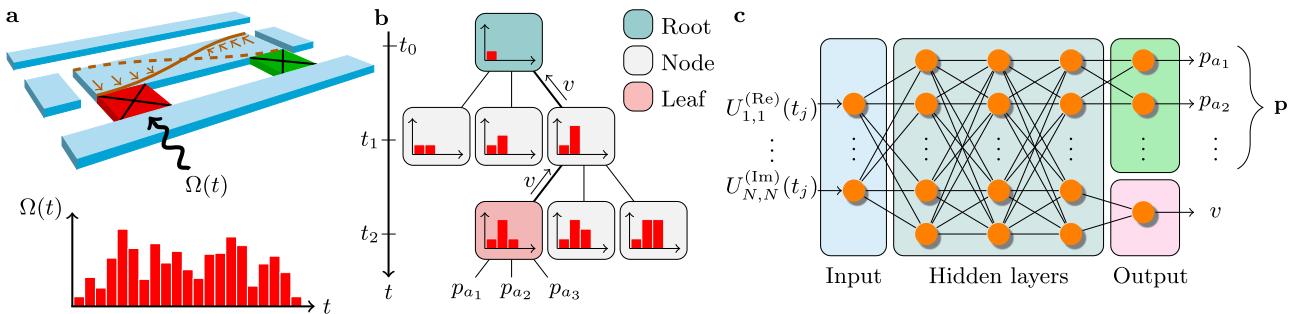


Fig. 1 Overview of circuit schematic and the AlphaZero algorithm. a Circuit-QED architecture consisting of two qubits (colored boxes) mounted on either side of a transmission line resonator. The first qubit is directly driven at the resonance frequency of the second one for a cross-resonance gate. An example of a piecewise-constant pulse is depicted below the setup. **b** The schematics of a Monte Carlo tree search. Here the nodes are depicted as pulse sequences and the edges as lines. A single search consists of a forward propagation, expansion, and a back-pass (see text). During the expansion phase, the corresponding state (unitary) is fed as input to the neural network. The edges are initialized using the probabilities \mathbf{p} from the neural network and the back-pass uses the value v also given by the neural network. **c** The neural network architecture used in AlphaZero. The network takes the state (unitary with $N = 4$) encountered in the tree search, as input and outputs probabilities for selecting individual actions $\mathbf{p} = \{p_{a_1}, p_{a_2}, \dots\}$ and an estimate of the final score (fidelity) v .

RESULTS

Unified quantum exploration algorithm

In this work, we seek to obtain pulse sequences that can unitarily steer a quantum system towards given desired dynamics. For our purposes, we quantify this task through the state-averaged overlap fidelity $\mathcal{F}(U(t))$ with respect to a target unitary \hat{U}_{target} ,

$$\mathcal{F}(\hat{U}(t)) = \left| \frac{1}{\dim} \text{Tr} \left[\hat{U}^\dagger(t) \hat{U}_{\text{target}} \right] \right|^2. \quad (1)$$

Here, $\hat{U}(t)$ denotes the time-evolution operator of the system, which solves the Schrödinger equation. We fix for concreteness our physical architecture as superconducting circuit QED,⁴⁹ being both a highly tunable and potentially scalable architecture, with potential near-term applications.⁵⁰ The system is chosen to be a resonator-coupled two-transmon system, as depicted in Fig. 1a. Here the transmon qubits are mounted on either side of a linear resonator and we drive the first qubit with an external control Ω , which could be a piecewise constant pulse as depicted in the bottom of the figure. The system dynamics are governed by the Hamiltonian⁵¹

$$\hat{H}(t) = \Delta \hat{b}_1^\dagger \hat{b}_1 + J(\hat{b}_1^\dagger \hat{b}_2 + \hat{b}_1 \hat{b}_2^\dagger) + \Omega(t)(\hat{b}_1^\dagger + \hat{b}_1), \quad (2)$$

where b_j is the qubit-lowering operator for transmon j , and the external control $\Omega(t)$ is shaped by the optimization algorithm to maximize (1), with $\hat{U}_{\text{target}} = \sqrt{ZX}$ being a standard entangling gate (see Methods). \hat{U}_{target} with single-qubit gates form a universal gate set, e.g., for quantum computation on a surface code circuit-QED layout. We fix the parameters to be within typical experimental values (see e.g. refs^{52,53}) for the qubit–qubit coupling $J/2\pi = 5$ MHz and the detuning $\Delta/2\pi = 0.35$ GHz.

We consider three optimization classes to test a unified AlphaZero algorithm and benchmark it against both domain-specialized and domain-agnostic algorithms. These three correspond to control parameters $\Omega(t)$ that are digital, i.e. taken from a discrete set of possibilities; that can vary continuously as a function of continuous but highly filtered controls; and lastly, piecewise-constant controls, which is standard in the QOCT approximation.

Within the RL framework, an autonomous agent must interact with an environment that at each time step t inhabits a state s_t . Here we choose the unitary $\hat{U}(t)$ to represent this state. The agent then alters the unitary at each time step t by applying an action a_t (here $\Omega(t)$) that transforms the unitary $\hat{U}(t) \rightarrow \hat{U}(t + \Delta t)$. For instance if the task is to create a piecewise-constant pulse sequence as depicted in Fig. 1a, the different actions would correspond to a finite set of predefined pulse amplitudes. This is

unlike standard QOCT, where the amplitudes are allowed to vary continuously. In the following, we will refer to the generation of an entire pulse sequence as an episode.

The purpose of the agent is to maximize an expected score z at the end of an episode, which we choose to be the fidelity $z = \mathcal{F}(U(T))$. Here T denotes the final time. This is done by implementing a probabilistic policy $\pi(s_t) = (\pi_{a_1}, \pi_{a_2}, \dots)$, which maps states s_t to probabilities of applying actions, i.e. $\pi_{a_j} = \Pr(a_j|s_t)$. The agent attempts to improve the policy by gradually updating it with respect to the experience it gains.

Figure 1b, c illustrate the tree search and the neural network for AlphaZero, respectively. The unitary found in the tree search is used as input for the neural network. The upper output of the neural network approximates the present policy for a given input state, i.e. $p_a \sim \pi_a$. Meanwhile, the lower output provides a value function which estimates the expected final reward, that is $v(s_t) \sim \mathcal{F}(T)$. In our work, we have found that providing AlphaZero with complete information of the physical system in form of the unitary to benefit its performance, though this may scale poorly for systems with larger Hilbert spaces.

Both functions use only information about the current state and suffer from being lower-dimensional approximations of extremely high-dimensional state and action spaces. The insight of the AlphaZero algorithm is to supplement the predictive power of the value function $v(s_t)$ with retrodictive information coming from future action decisions in a Monte Carlo search tree. The tree depicted in Fig. 1b consists of nodes, which represent states (here depicted as pulses) and edges, which are state-action pairs (depicted as lines). At each branch in the tree, the algorithm chooses actions based on a combination of those with the highest expected reward and the highest uncertainty, a measure of which edges remain unexplored. Whenever new states (called leaf-nodes) are explored, the neural network is used to estimate the value of that node, and the information is propagated backward in the tree to the root node. The forward and backward traversals of the tree are described in greater detail in Methods. For the interested reader we have also provided a step-by-step walkthrough of the algorithm in the Supplementary Materials.

In the manner described above, the predictive nature of the network is able to inform choices in the tree while the retrodictive information coming back in time is able to give better estimates of the state values already explored, which are then used to train the network, i.e. update the network parameters in order to improve its predictions. The training occurs after each completed episode. This reinforcing mechanism is thus able to globally learn about the parameter landscape by choosing the most promising branches while effectively culling the vast majority of the rest. The result is

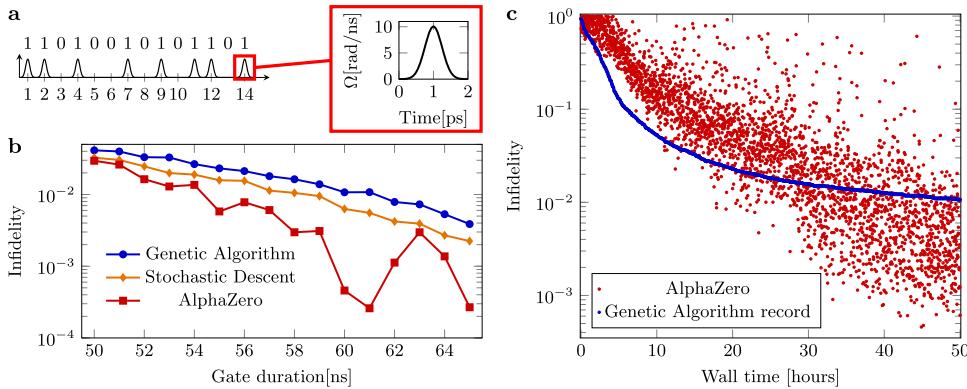


Fig. 2 Optimization in the digital case (SFQ). a Illustration of how a SFQ pulse train can be encoded into a bit string, with bits displayed above the figure and gate number below. The figure also contains a zoom-in that depicts the exact shape of a single pulse. **b** The best obtained infidelity ($1 - \mathcal{F}$) at each gate duration for different discrete optimization algorithms. Each algorithm had 50 h wall time for each gate duration. **c** A comparison between the infidelities obtained by AlphaZero and the GA at 60 ns. For AlphaZero, each dot represents the infidelity obtained at the end of a unique episode, i.e. generation of an entire pulse-sequence, while for the GA each dot represents the highest scoring member in the population after each iteration.

neither an exhaustive sampling at full depth, which would yield the true landscape albeit at a computationally untenable cost, nor is it an exhaustive sampling at shallow depth, which would require a prohibitively slow learning rate for information from the full depth of the tree to propagate back. Instead, AlphaZero intelligently balances the depth and the breadth of the search below each node. While the hidden-variable approximation given by the neural network and MC tree are certainly not exhaustive and cannot find solutions with exponentially small footprint, it is nonetheless able to discover patterns and learn an effective global policy strategy that produces robust, heterogeneous classes of promising solutions. In our implementation we restrict AlphaZero such that it can only find new unique solutions, which is done by cutting off branches in the tree that have previously been fully explored. Hence, each solution found by AlphaZero is different from any previously found solution.

In what follows we apply the algorithm with a unified set of algorithmic parameters (hyperparameters) to three optimization classes: discrete, continuous, and continuous with strong constraints. We have found these hyperparameters by fine-tuning them with respect to the continuous problem. The three problem types accentuate different optimization strategies. In the discrete optimization case, we show how AlphaZero stands up against other domain-agnostic methods (where the gradient is not defined) and compare their abilities to learn structures in the parameters. For the constrained continuous pulses, we validate the hypothesis that the analytical gradient, while computable, is highly inefficient and indeed unable to find near global solutions that are at least as good as those found by AlphaZero. Finally, in the continuous-valued piecewise-constant case, we show the balance between state-of-the-art physics-specialized and agnostic AlphaZero approaches. We show that the combination of exploration and exploitation is able to produce new clusters of high-quality solutions that are otherwise highly unlikely to be found, while learning hidden problem symmetry.

Digital gate sequences

As a first application with AlphaZero, we demonstrate optimal control using Single Flux Quantum (SFQ) pulses.^{45,54,55} The aim is to control the quantum system by using a pulse train that consists of individual, very short pulses typically in the pico-second scale. This technology originated as way of utilizing superconductors for large-scale, ultrafast, digital, classical computing.⁵⁶ At each time slice there either is a pulse or not, which implies that the unitary evolution is governed by two unitaries \hat{U}_1 and \hat{U}_0 . Hence, the pulse

train can be stored as a digital bit string with 0 and 1 denoting no pulse and a single pulse respectively. SFQ devices are interesting candidates for quantum computation since they potentially allow for ultrafast gate operations as well as scalable quantum hardware.⁵⁵ We model the pulses as $\Delta t = 2.0$ ps Gaussian functions $\Omega(t) = \frac{a}{\sqrt{2\pi\tau}} e^{-\frac{(t-\Delta t/2)^2}{2\tau^2}}$, where $\tau = 0.25$ ps and $a = 2\pi/1000$. The pulse is depicted to the right in Fig. 2a.

The optimization task is to find the input string that maximizes the fidelity functional (1). The current approach for this type of optimization is to apply a genetic algorithm (GA)^{45,57,58}. Besides GA and AlphaZero, we also compare a conventional algorithms, stochastic descent (SD) as in ref. ²⁴ SD is a time-local, greedy optimizer that changes the pulse at a randomly chosen time if this results in an increasing fidelity.

Our unified AlphaZero algorithm has an action space of 60 for the neural network, and thus we group together binary SFQ action choices of multiple time steps. For this purpose, we take larger steps in time, and the 60 action choices are given using bit strings from a randomly chosen basis (see Methods). We benchmark the different algorithms by using equal wall-time simulations. For all simulations presented in this paper, we used a wall-time of 50 h on an Intel Xeon X5650 CPU (2.7 GHz) processor. Similar to ref. ⁴⁵ we use a population size of 70 with a mutation probability of 0.001 for the GA (see Methods). In the Supplementary Material, we provide an alternative analysis of the wall time consumption needed to reach a predefined infidelity.

The results are plotted in Fig. 2b. Amongst conventional approaches, we see the SD algorithm performs slightly better than the GA. We attribute this to the fact that the SD algorithm is a greedy exploitation algorithm, while the GA is an exploration algorithm performing random permutations. As with many exploration algorithms, learning can be quite slow. We emphasize that AlphaZero contains a deep lookahead tree search, which we found crucial to the success of our RL implementation (having originally tested a simpler RL algorithm, Deep Q-Network (DQN), on a less complicated problem). We see in Fig. 2b that AlphaZero indeed performs dramatically better than the greedy approach, with up to over an order of magnitude improvement in the low error regime. Interestingly, AlphaZero's solutions begin to fluctuate in the low infidelity regime. We attribute this to the neural network's incapability of distinguishing between different unitaries belonging to the very low infidelity regime. However, AlphaZero still maintains a systematic improvement over the other algorithms.

AlphaZero and GA are both learning algorithms in the sense that they utilize previous obtained solutions in order to form new ones. We compare the learning curves for the two algorithms in Fig. 2c, where we have plotted the infidelity as a function of wall time at 60 ns. For AlphaZero, we use the infidelity after each episode, where each data point is unique. For GA, we use the best performing solution in the population after each iteration. Since GA is a relatively greedy algorithm it performs very well initially, but fails to explore the larger solution space as the members in the population converge upon a single class of solution and the learning curve flattens out. In contrast, AlphaZero keeps a high level of exploration that ultimately allows it to reach a very large number of different high-fidelity solutions.

Constrained analog pulses

A common challenge within quantum optimization is achieving realistic and efficient controls when experimental limitations constrain the underlying dynamics. Such constraints become very important when high precision is required, e.g. for very high-fidelity operation of quantum technologies. Here, we consider standard constraints on duration, bandwidth, and maximum energy. Such constraints can be expected to greatly increase the computational cost of Hessian approximation-based solutions, which are otherwise known to converge quickly¹⁶ and generally outperform other greedy methods.^{21,22} The workhorse algorithm for this is GRAPE,⁸ with quasi-Newton²⁰ and exact derivative⁵⁹ enhancements being crucial to the state of the art and its super-linear convergence. Note that GRAPE, unlike AlphaZero, can handle pulses that are continuous in amplitude.

We model the bandwidth constraints via a convolution with a Gaussian filter function

$$\tilde{\Omega}(t) = \int_{-\infty}^{\infty} e^{-\frac{(t-t')^2}{\sigma^2}} \Omega(t') dt', \quad (3)$$

where $\tilde{\Omega}(t)$ denotes the filtered control function. Figure 3a illustrates the effect of this filter. Here, a piecewise-constant pulse (dark blue) with amplitudes a_{1-4} is convoluted into a smooth pulse (light orange) via Eq. (3). Throughout the remainder of this paper, we constrain the pulse amplitude to lie between 0 and $\Omega_{\max}/2\pi = 1.0$ GHz.

Most commonly, GRAPE is applied to piecewise-constant pulses, but it can be modified to include filtering,^{59,60} as we also do here. Each time-step is divided into a number of substeps (giving the resolution) and the filtered pulse is then approximated as being constant within each substep. This subdivision is depicted in Fig. 3a as light orange vertical lines. In order to obtain the gradient, GRAPE calculates the time-evolution unitary using matrix exponentiation at each substep.

Figure 3b shows the error (infidelity) between the exact and discretized unitaries as a function of the resolution. If we seek errors well below the desired gate error (10^{-2}), the resolution should be well above a hundred. This significantly impedes the performance of GRAPE for this type of problem, since it requires considerably more matrix multiplications. A different strategy is to limit the control to a set of discretized amplitudes whose corresponding unitary can be calculated in advance and then apply a discretized optimization algorithm such as AlphaZero. In order to do so, we apply a two-action update strategy, where we propagate from half the previous pulse to halfway into the next one. So, if the previous action was a_2 and the next one a_3 then the unitary $U_{2,3}$ would correspond to the shaded region in Fig. 3a. Here we ignore negligible contributions from adjacent pulses. For instance, calculating $U_{2,3}$ would be independent of a_1 and a_4 . Here we limit the amplitude to 60 different values (out of a continuous set), hence this method requires calculating $60^2 = 3600$ unitaries, which we do in the beginning of the simulation.

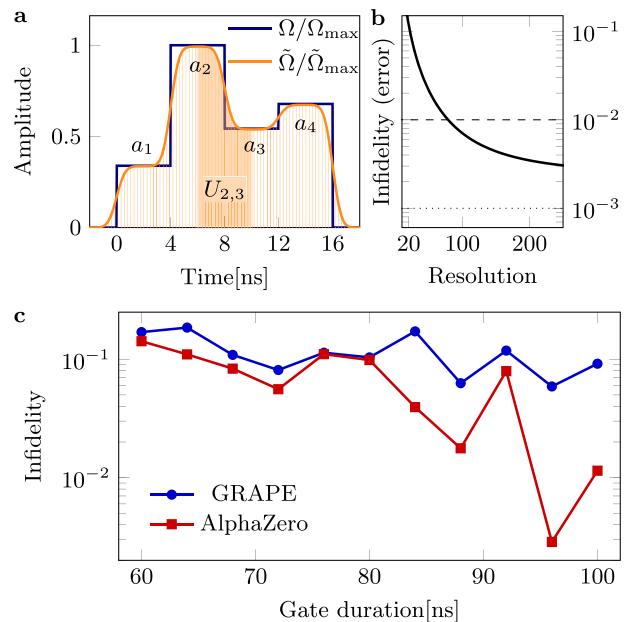


Fig. 3 Optimization of constrained continuous pulses (Gaussian filtering). **a** A piecewise-constant pulse (dark blue) convoluted by a Gaussian filter (light orange). The vertical orange lines depict the resolution, i.e. number of substeps per 4.0 ns. Here $\sigma = 0.7$ ns. **b** The infidelity (error) between the exact and the calculated unitary as a function of its resolution. **c** Comparison between AlphaZero and GRAPE on the cross-resonance gate using Gaussian filtered pulses. Each point represents the best obtained infidelity during a 50 h wall time simulation at that specific gate duration.

In our comparison between AlphaZero and GRAPE, we choose 4.0 ns convoluted pulses using $\sigma = 0.7$ ns. For GRAPE, we choose a resolution of 200, i.e., every 4.0 ns interval is divided into 200 substeps. Figure 3c shows the results of an equal wall-time simulation, where GRAPE was applied with random seeding. Here, AlphaZero obtains a systematic improvement over its domain-specialized counterpart. At 96 ns, AlphaZero outperforms GRAPE with an improvement that is significantly above one order of magnitude. Interestingly, both graphs show significant fluctuations, which we attribute to the difficulty of the optimization task itself caused by the highly constrained dynamics. This is likely compounded by the random initialization of the neural network which can effect the convergence properties of AlphaZero. Despite these fluctuations, AlphaZero performs significantly better in the regime of interest corresponding to infidelities below 10^{-2} .

Piecewise-constant analog pulses

So far, we have considered problems where gradient searches have not been applicable (digital sequence) or where gradient searches become inefficient (constrained analog pulses). For specific tasks where highly specialized algorithms exist and are known to perform relatively well, domain-agnostic algorithms typically perform inadequately. Thus, to properly benchmark our algorithm we have also considered the domain of piecewise-constant pulses, a scenario where GRAPE typically performs extremely well due to the presence of high-frequency components and the limited number of matrix multiplications. In the following we hence focus on piecewise constant pulses where we choose a single-step duration of 2 ns. In this scenario, we characterize the performance of the exploitation and exploration algorithms in terms of both the variety of solutions found and the quality of the solutions. Here we also introduce a new RL algorithm, named Q-learning. Q-learning was one of the first RL algorithms developed, and applied recently to quantum

control.^{24,37} It is a tabular-based algorithm that applies one-step updates in order to solve the optimal Bellman equation⁶¹ (see Methods). Here we apply an extended version that uses eligibility traces and regular replays, similar to ref.³⁷ Originally, we attempted to apply Q-learning to digital gates (SFQ) pulses, but ultimately the algorithm failed due to the very large search space, where tabular-based methods are known to break down. This is one reason why modern RL algorithms use deep neural networks instead, motivating also our use of AlphaZero.

At first, we compare the algorithms already discussed, namely Q-learning, Stochastic Descent, AlphaZero, and GRAPE. Figure 4a shows GRAPE is able to outperform the other algorithms for piecewise-constant pulses. However, AlphaZero still performs well despite its limitation of only having amplitude-discretized controls. To improve the AlphaZero algorithm further we conceive a hybrid algorithm where GRAPE optimizes the solutions found by AlphaZero. The hybrid algorithm, which is given the same wall-time as the others, is also plotted in Fig. 4a. Here the hybrid algorithm shows a significant improvement over GRAPE near 60 ns,

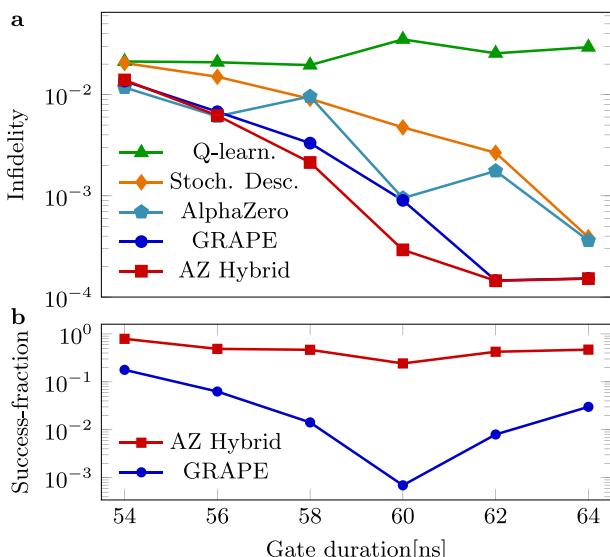


Fig. 4 Optimization on continuous (piecewise-constant) pulses. a An equal wall-time comparison between the various algorithms. The AlphaZero (here abbreviated AZ) Hybrid is presented in the text. **b** The fraction of successful solutions found by AlphaZero Hybrid and the GRAPE algorithm.

which we relate to the presence of a quantum speed limit (QSL) where the optimization task becomes difficult due to induced traps in the fidelity landscape.^{22,24,26} It is also worth noting that the optimization curve flattens out and the two algorithms again perform equally well when the pulse duration goes beyond 62 ns. Interestingly, there is an entirely different time scale, beyond 200 ns (not plotted here), for reaching gate infidelities below 10^{-4} .

We also quantify the number of successful solutions found by either GRAPE or the hybrid AlphaZero algorithm, which we define as solutions having infidelities within four times the lowest infidelity obtained. The fraction of successful solutions are plotted in Fig. 4b. Here the improvement is even more substantial. At 60 ns, we find almost three orders of magnitude more successful solutions compared to GRAPE with random seeding. The fact that the GRAPE-curve dips around 60 ns seems to confirm our previous statement about the QSL in the sense that this is a combinatorially harder region to obtain relatively good solutions. We believe this marks a transition into a combinatorially harder optimization region in the control landscape similar to the glassy phase transition studied in ref.⁶² Having a large number of good solutions is especially important because experimentally it may be that some are better suited or some provide additional advantages.

To further investigate the differences between the two algorithms, we perform a clustering analysis of the obtained solutions similar to ref.¹¹ We compare the exploration of the control parameter landscape using a two-dimensional embedding provided by the t-SNE (t-Distributed Stochastic Neighbor Embedding) visualization method.⁶³

The t-SNE algorithm represents a set of high-dimensional vectors by a set of low-dimensional vectors, e.g. two dimensions as used here. If the Euclidean distance between any pair of vectors in the high-dimensional set is low (high) then the Euclidean distance between the same pair in the low-dimensional set should also be low (high). This allows one to visualize and investigate clustering tendencies. Specifically, the distances are weighted by the Student's t-distribution whereby nearby points are strongly favored. The algorithm then calculates joint probability distributions that any pair of vectors are "close" to each other both in the high- and low-dimensional set, and then minimizes the Kullback-Leibler divergence between the two distributions. See ref.⁶³ for a more detailed description.

We do a single t-SNE analysis at 60 ns, plotted in Fig. 5, which we have separated for clarity into different figures for GRAPE (a), AlphaZero i.e. Hybrid before optimization (b), and after optimization (c). Here the color scale depicts the infidelity. We used a

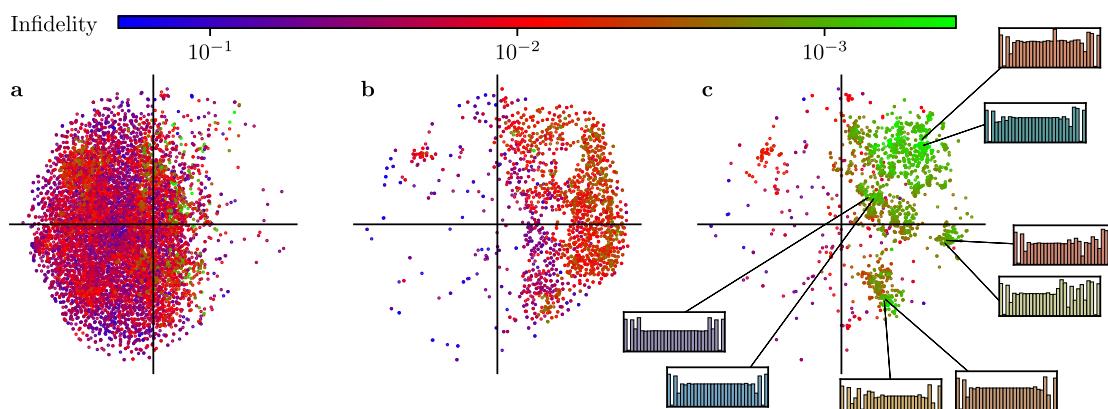


Fig. 5 Two-dimensional representation of the final pulse vectors at 60 ns using the t-SNE algorithm. t-SNE is a visualization method that here represents the final pulse vectors by a set of two-dimensional vectors, in such a way that if any two pulse vectors are close to each other then the coordinates of their two-dimensional representations should be relatively close to each other as well. The color scale shows the infidelity of the pulses. **a** GRAPE with random seeding, **b** AlphaZero, **c** The Hybrid, i.e. AlphaZero solutions after being optimized with GRAPE. In the latter case, some example high-fidelity pulses are shown.

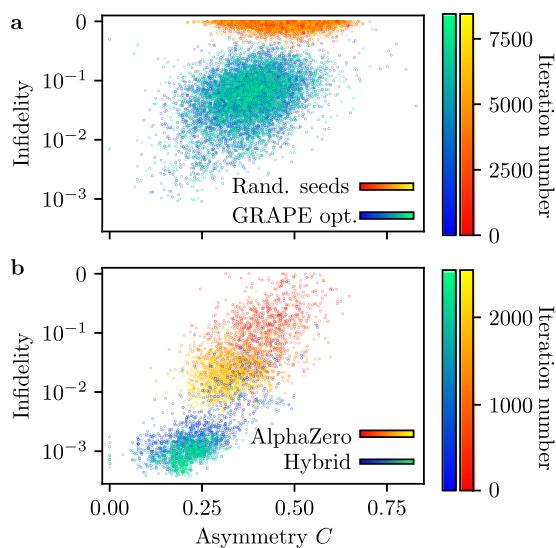


Fig. 6 Asymmetry of solutions. The figure depicts the infidelity ($1 - \mathcal{F}$) as a function of the asymmetry measure (4) at 60 ns for: **a** Random seeds (plotted in red-yellow scale) followed by an optimization with GRAPE (plotted in blue-green scale) and **b** AlphaZero solution (plotted in red-yellow scale) followed by an optimization with GRAPE (i.e. the Hybrid algorithm, plotted in blue-green scale). The color scale depicts the iteration of the algorithm.

perplexity of 30 (a parameter t-SNE uses that estimates how many nearest neighbors each point has), which we found gave reasonable clustering-plots.

Strikingly, the two algorithms seem to prefer entirely different portions of the landscape. GRAPE mostly finds solutions clustered to the left in the t-SNE representation, but its high performing solutions are actually clustered to the right. Interestingly, AlphaZero primarily finds solutions in the right region, which implies that AlphaZero has identified an underlying basic generic structure of good solutions. When all the AlphaZero solutions are optimized this leads to a large quantity of high performing solutions that inhabit the same region in the t-SNE representation.

We also see that the hybrid solutions naturally cluster towards some general basins of attraction. This suggests that AlphaZero has not converged on a single class but multiple different classes of solutions with different underlying physics. Some pulses from different clusters are depicted, showing some resemblance to typical bang-bang sequences. The different clustering that occurs demonstrates that a global exploration has indeed taken place, effectively finding different classes of solutions in different parts of the landscape.

We further test the hypothesis that AlphaZero has found underlying structure that supersedes a shallow heuristic search. Note that the solutions seem to have at least some symmetry with respect to a reflection around the center of the time-axis. In fact, this symmetry already exists in the control problem. Since the Hamiltonian is real and the target its own transpose, the fidelity is unchanged if the pulse sequence is reversed i.e. $\mathcal{F}(\Omega_1, \Omega_2, \dots, \Omega_{N-1}, \Omega_N) = \mathcal{F}(\Omega_N, \Omega_{N-1}, \dots, \Omega_2, \Omega_1)$. This property defines a discrete \mathbb{Z}_2 symmetry similar to what was studied in ref. ⁶⁴

However, it is not a priori clear that satisfying this symmetry is a good control strategy. We quantify the degree of time-asymmetry in the pulses via the measure

$$C(\{\Omega(t)\}) = \frac{1}{N} \sqrt{\sum_{j=1}^N |\Omega_j - \Omega_{N-j}|^2}, \quad (4)$$

where $C = 0$ implies pulses that are completely palindromic, i.e. symmetric with respect to reversion of the sequence.

We plot in Fig. 6 the infidelity and the asymmetry for the two algorithms, i.e. in (a) we plot random seeds and the same seeds optimized by GRAPE; in (b) we plot the AlphaZero solutions and the same solutions further optimized by GRAPE, which is the Hybrid algorithm.

Here the color scale depicts the iteration number. The first thing to notice is that high-fidelity solutions tend to maintain this symmetry. The second feature is that GRAPE often only partially satisfies this symmetry. In contrast, AlphaZero learns over its training to increasingly prefer this symmetry, moving towards the bottom left of the plot. After post-optimization using GRAPE, the solutions improve significantly in infidelity and move ever further to the bottom left emphasizing this trend. Note that we are not arguing that $C = 0$ is optimal, but merely that increasing the symmetry to some extent seems preferable. We conclude that AlphaZero has identified this underlying symmetry specific to the problem instance we have chosen. Naturally, hard-coding such heuristics would not only be inefficient, but for many problems finding symmetries is nontrivial. Using deep learning, AlphaZero is able to learn these hidden symmetries without the need for human intervention. We therefore expect that AlphaZero's ability to learn hidden problem structures generalizes to other problems as well.

DISCUSSION

From our three examples, we conclude that the AlphaZero methodology of combining neural network and guided tree search reinforces global information about good solutions that can also mark a significant algorithmic advantage for quantum optimization. This is true for specific problems, but especially when comparing across a range of problems. None of the other algorithms we have considered are able to do well on all three problems, be it with heuristic, machine learning or domain-specialized approaches.

The three problems considered marked different optimization tasks, but AlphaZero is able to find high-fidelity solutions with a single set of algorithmic hyperparameters. This suggests that learning the control landscape can be performed with minimal expert knowledge about the physical problem.

This conclusion is further enforced by the realization that hidden symmetries in the dynamics can be effectively learned by AlphaZero during its training. Such unexpected symmetries are not trivial to find for many Hamiltonians and would require significant human intervention even where they can be found. Moreover, hard-coding such heuristics into optimization algorithms can have many pitfalls, limiting broad exploration and potentially leading to suboptimal trapping in the optimization landscape.

Nonetheless, because the deep exploration methodology is by design agnostic to expert knowledge, it is most powerful when combined with specialized knowledge about locally exploiting promising seeds, leveraging the vast body of literature about local quantum optimization. This tradeoff between exploitation and exploration is a common trend in reinforcement learning and optimization in general. For example, in AlphaZero's chess matches with its competing AI, Stockfish,⁶⁵ the latter was trained with sophisticated domain knowledge and thus was generally acknowledged as outperforming in the final moves of games. Combining the domain-agnostic exploration of the former with the domain-specialized exploitation of the latter seems like a common sense solution, as we have done here in the quantum dynamics case. An even tighter integration of the two approaches that examines the tradeoffs during different learning stages may also be promising. Alternatively, one could also relax the *tabula rasa* character of the learning to enhance the exploration

abilities using specialized knowledge. Supervised learning can in principle speed up the initial learning phase, perhaps most seemlessly when integrated with other broad exploration strategies, for instance crowd sourcing.^{11,66}

In this work we have considered digital, constrained, and underconstrained optimization of controlled quantum dynamics in the context of the design and execution of physical quantum-mechanical devices. This choice was deliberately made because the most advanced algorithms exist in this field owing to half a century of dedicated research. That being said, many of the more abstract and potentially groundbreaking dynamics algorithms, including those used in the design of digital sequences of quantum circuits or for analog evolutions in annealers and variational eigensolvers, can be seen as direct analogs of the algorithmic framework illustrated here.

METHODS

Reinforcement Learning

A general RL setup consists of an environment and an agent. At each time step t , the environment is characterized by a state s_t . Given s_t , the agent selects an action a_t that changes the environment to a new state s_{t+1} . Based on this change the agent receives a feedback signal called a reward, $r_{t+1} \in \mathbb{R}$. The agent must learn how to maximize the sum of rewards it receives during an episode. This is done by implementing a policy π , which is a mapping from all states of the environment to probabilities of selecting possible actions $\Pr(a|s) = p_a(s)$. The state-value function describes the quality of a given policy

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{t' > t} r_{t'} | s = s_t \right], \quad (5)$$

which is simply the expected sum of future reward staring from state s and subsequently following the policy π . Given two policies π and π' we say that $\pi \geq \pi'$ if $v_\pi(s) \geq v_{\pi'}(s)$ for all states s .

The task considered here is to construct a pulse sequence, which realizes a target unitary. At each time step, the agent must select an action that updates the unitary representing the state of the system. At each time step, the reward is zero except at the last step where it is simply the fidelity given by Eq. (1).

AlphaZero implementation

AlphaZero is a policy improvement algorithm that combines a neural network with a Monte Carlo Tree Search (MCTS) as depicted in Fig. 1b, c.^{47,48} The neural network maps from states to policies $\mathbf{p} = (p_1, p_2, \dots)$ and values v . The MCTS, guided by the neural network, also computes a policy π that the actions are drawn from. At each time step, the policy π is stored in a replay buffer, i.e. database. At the end of an episode, the final score $z = \sum_t r_t$ is also stored in the buffer. Training of the neural network uses data drawn uniformly at random from the replay buffer in order to let the network predictions (\mathbf{p}, v) approach the stored data (π, z) . This is done by minimizing the loss function

$$l(\theta) = (z - v)^2 - \pi^\top \log \mathbf{p} + c \|\theta\|^2, \quad (6)$$

where the last term denotes L2 regularization with respect to the network parameters θ . The training of the neural network occurs after each completed episode.

A MCTS is a way of looking several steps ahead by only visiting a small subset of possible future states. The tree is built by nodes (states) connected to each other by edges (state-action pairs). Each edge has four numbers associated with it: The number of visits $N(s, a)$, the total action value $W(s, a)$, the mean action value $Q(s, a)$, and a prior probability of selecting set edge $P(s, a)$. Starting from the root node (initial state), a single tree search moves through the tree by selecting actions according to $a_t = \arg \max_a (Q(s_t, a) + U(s_t, a))$, where $U(s_t, a)$ denotes an uncertainty given by

$$U(s, a) = c_{\text{puct}} P(s, a) \frac{\sqrt{\sum_b N(s, b)}}{N(s, a)}. \quad (7)$$

Here c_{puct} denotes a parameter determining the level of exploration. If a terminal node or a leaf node (i.e. a not-previously-visited state) is encountered, the search stops. The tree is expanded in the latter case by

M. Dalgaard et al.

adding the node and initializing its edges as $N(s, a) = W(s, a) = Q(s, a) = 0$ and $P(s, a) = p_a$, where p_a is given by the neural network. The rest of the tree is updated by using the state-value v in a backwards pass through all the visited edges since the root node according to $N(s, a) \leftarrow N(s, a) + 1$, $W(s, a) \leftarrow W(s, a) + v$, and $Q(s, a) \leftarrow W(s, a) / N(s, a)$. After a pre-set number of such searches have been conducted, an actual policy is calculated according to

$$\pi(a|s_0) = \frac{N(s_0, a)^{1/\tau}}{\sum_b N(s_0, b)^{1/\tau}}, \quad (8)$$

where s_0 is the root state and τ denotes a parameter controlling the level of exploration, which is annealed during the simulations. The action is drawn from the policy and the rest of the tree is reused for subsequent searches during the episode.

For all tasks presented in this paper we used the same algorithmic parameters. The learning rate was 0.01, $c_{\text{puct}} = 1.0$, and τ was hyperbolically annealed from 1.0 using an annealing rate of 0.001. After τ was annealed below a value of 0.90 we switched to deterministic policies by setting the largest policy value to one and the others zero. The neural network was a simple feed forward network where the hidden nodes consisted of four layers. Each layer contained 400 nodes followed by batch normalization and a rectified linear unit. The neural network parameters were initialized at random.

Both the policy and the value head of the neural network consisted of a single hidden layer as well, where the policy head ended in a sigmoid-layer with same dimension as the action space and the value head ended in a single linear node. The L2 regularization parameter was $c = 0.001$ and we used stochastic gradient descent (SGD) for training the network. Similar to the AlphaZero paper⁴⁷ we achieve more exploration by adding Dirichlet noise to the search probabilities for the root nodes $P(s, a) = (1 - \epsilon)p_a + \epsilon\eta_a$, where $\eta \sim \text{Dir}(0.03)$ and $\epsilon = 0.25$.

GA implementation

A genetic algorithm (GA) works by iteratively updating a population of solutions, which are bit strings.^{57,58} A GA generates new solutions based on the old population via processes inspired by biological evolution, namely crossover and mutations, which respectively combine two parent solutions by choosing elements from each parent at random and flip individual bits at random. If any improved solutions are found, these replace the worst ones in the population. Similar to ref.,⁴⁵ we used a population size of 70 and a mutation probability of 0.001. At each iteration we would select 2×30 parent solutions.

Q-learning implementation

Similar to Eq. (5) one can define an action-value function

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t' > t} r_{t'} | s = s_t, a = a_t \right], \quad (9)$$

which is the expected reward if we choose action a from state s and then follow the policy π .³⁰ Q-learning is a tabular-based RL algorithm, which approximates the optimal action-value function i.e. the action-value function for the optimal policy $\pi_{\text{opt}} = \max_\pi v_\pi(s)$. The approximation $Q(s, a)$ is initialized at random and subsequently updated according to

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \quad (10)$$

where α denotes the learning rate. Similar to ref.²⁴ we choose our state to be a tuple of time and control $s = (t, \Omega)$. The learning rate was $\alpha = 0.01$ and we followed an epsilon-greedy strategy with linear annealing of epsilon.³⁰ We also implemented eligibility traces and regular replays of the best encountered episode similar to ref.³⁷

Cross-resonance gate

The cross-resonance (CR) gate^{51,67,68} is currently the standard fixed-frequency qubit entangling gate used on transmon systems. Its main advantage is avoiding the overhead associated with magnetic (flux) tuning of the frequency,^{69,70} which can be a leading cause of dephasing. As illustrated in Fig. 1a, the physical setup we optimize includes two fixed-frequency qubits that are coupled to each other via a transmission line resonator. The transmons⁷⁰ may be modeled as anharmonically spaced Duffing oscillators,⁷¹

resulting in an extended Jaynes-Cummings model Hamiltonian

$$\hat{H} = \sum_{j=1,2} \left(\omega_j \hat{b}_j^\dagger \hat{b}_j + \frac{\delta_j}{2} \hat{b}_j^\dagger \hat{b}_j (\hat{b}_j^\dagger \hat{b}_j - 1) + \omega_r \hat{a}^\dagger \hat{a} + g_j (\hat{b}_j^\dagger \hat{a} + \hat{b}_j \hat{a}^\dagger) + \Omega(t) (\hat{b}_j + \hat{b}_j^\dagger) \right), \quad (11)$$

where $\hat{b}_{1,2}^\dagger$ ($\hat{b}_{1,2}$) and \hat{a}^\dagger (\hat{a}) are the transmon and cavity creation (annihilation) operators respectively. Here $\omega_1 \neq \omega_2$ is the transmon resonance frequency, $\delta_{1,2}$ denotes the anharmonicity, ω_r denotes the cavity resonance, and $g_{1,2}$ the transmon-cavity coupling. The transmons are directly driven by external control parameters $\Omega(t)$, increasing the controllability compared to earlier architectures that drive through the common cavity. The transition of the second qubit is then driven resonantly through the control line of the first.⁶⁰

This model may be significantly simplified using the method in ref.⁵¹ After adiabatic elimination of the cavity and block diagonalization into the qubit subspace, the authors derive an equivalent equation (Eq. (3.3)), which is the same as our Eq. (2). Here we seek to replicate the unitary

$$\hat{U}_{\text{target}} = \exp\left(-i\frac{\pi}{4}XZ\right) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -i & 0 & 0 \\ -i & 1 & 0 & 0 \\ 0 & 0 & 1 & i \\ 0 & 0 & i & 1 \end{bmatrix}, \quad (12)$$

which is a \sqrt{ZX} gate since $\hat{U}_{\text{target}}^2 = ZX$ up to a global phase. To see that the natural gate that is produced from this Hamiltonian is a \sqrt{ZX} gate, a (Schrieffer-Wolff) perturbative expansion shows⁶⁸ that the leading coefficients in the effective driving terms are given by

$$\hat{H}_d = \Omega(t) \left[XI + \frac{J}{\Delta} ZX + mIX \right], \quad (13)$$

where Z and X are Pauli matrices acting on the respective qubits, I is the identity, and m is a hand-tuned crosstalk parameter. The single-qubit terms and higher order terms (not shown) must be decoupled in the control optimization in order to correctly implement the CR gate, preventing a simple analytic solution.

Digital pulses

For each time step, the evolution of the system is governed by either one of two unitaries \hat{U}_0 and \hat{U}_1 , which respectively corresponds to the amplitude being zero or not.⁷² We calculate these unitaries in advance by solving the Schrödinger equation numerically. The entire pulse sequence can be encoded as a bit string as illustrated to the left in Fig. 2a and the corresponding unitary can be calculated as $\hat{U}(T) = \prod_{j=1}^N \hat{U}_{b_j}$ where $b_j \in [0, 1]$. Pulse durations in the nano-second scale require $10^4 - 10^5$ steps.

For AlphaZero we create 60 unitaries $\{\hat{U}^{(i)}\}_{i=1}^{60}$ by drawing a bit string $b_1^{(i)}, b_2^{(i)}, \dots, b_{500}^{(i)}$ at random, where $b_j^{(i)} \in [0, 1]$, which we then multiply $\hat{U}^{(i)} = \prod_{j=1}^{500} \hat{U}_{b_j^{(i)}}$. In order to obtain pulse sequences that have both high and low concentrations of $b_j^{(i)} = 0$ we anneal the probability $\text{Pr}(b_j^{(i)} = 0)$ linearly from one ($i = 1$) to zero ($i = 60$). The 60 unitaries constitute the action space and the unitary is now calculated as $\hat{U}(t) = \prod_{t' \leq t} \hat{U}^{(a_{t'})}$. The 60 actions allows us to use the same neural network architecture as for piecewise constant and filtered pulses which have the same input space dimension.

DATA AVAILABILITY

All data presented in this paper is available upon request. All requests should be directed at J. Sherson (sherson@phys.au.dk).

CODE AVAILABILITY

All code used to generate the presented data in this paper is available upon request. All requests should be directed at J. Sherson (sherson@phys.au.dk).

Received: 23 July 2019; Accepted: 6 December 2019;

Published online: 14 January 2020

REFERENCES

- Kandala, A. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* **549**, 242 (2017).
- Johnson, M. W. Quantum annealing with manufactured spins. *Nature* **473**, 194 (2011).
- Lloyd, S. Universal quantum simulators. *Science* **273**, 1073–1078 (1996).
- Georgescu, I. M., Ashhab, S. & Nori, F. Quantum simulation. *Rev. Mod. Phys.* **86**, 153–185 (2014).
- Bocharov, A., Roetteler, M. & Svore, K. M. Efficient synthesis of universal repeat-until-success quantum circuits. *Phys. Rev. Lett.* **114**, 080502 (2015).
- Motzoi, F., Kaicher, M. P. & Wilhelm, F. K. Linear and logarithmic time compositions of quantum many-body operators. *Phys. Rev. Lett.* **119**, 160503 (2017).
- Warren, W. S., Rabitz, H. & Dahleh, M. Coherent control of quantum dynamics: the dream is alive. *Science* **259**, 1581–1589 (1993).
- Khaneja, N., Reiss, T., Kehlet, C., Schulze-Herbrüggen, T. & Glaser, S. J. Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms. *J. Mag. Res.* **172**, 296–305 (2005).
- Glaser, S. J. Training schrödinger's cat: quantum optimal control. *Eur. Phys. J. D* **69**, 279 (2015).
- Biamonte, J. Quantum machine learning. *Nature* **549**, 195 (2017).
- Sørensen, J. J. W. H. Exploring the quantum speed limit with computer games. *Nature* **532**, 210 EP (2016).
- Pechen, A. N. & Tannor, D. J. Are there traps in quantum control landscapes? *Phys. Rev. Lett.* **106**, 120402 (2011).
- De Fouquieres, P. & Schirmer, S. G. A closer look at quantum control landscapes and their implication for control optimization. *Infin. Dimensional Anal. Quantum Probab. Relat. Top.* **16**, 1350021 (2013).
- Zhdanov, D. V. & Seideman, T. Role of control constraints in quantum optimal control. *Phys. Rev. A* **92**, 052109 (2015).
- Hardy, L. Quantum theory from five reasonable axioms. Preprint at arXiv: quant-ph/0101012 (2001).
- Rabitz, H. A., Hsieh, M. M. & Rosenthal, C. M. Quantum optimally controlled transition landscapes. *Science* **303**, 1998–2001 (2004).
- Freeman, R. *Handbook of Nuclear Magnetic Resonance*. (John Wiley and Sons, New York, NY, United States, 1987).
- Tannor, D. J. & Rice, S. A. *Coherent Pulse Sequence Control of Product Formation in Chemical Reactions* 441–523 (John Wiley and Sons Ltd, 2007).
- Kawashima, H., Wefers, M. M. & Nelson, K. A. Femtosecond pulse shaping, multiple-pulse spectroscopy, and optical control. *Annu. Rev. Phys. Chem.* **46**, 627–656 (1995).
- de Fouquieres, P., Schirmer, S., Glaser, S. & Kuprov, I. Second order gradient ascent pulse engineering. *J. Mag. Res.* **212**, 412–417 (2011).
- Machnes, S. Comparing optimizing, and benchmarking quantum-control algorithms in a unifying programming framework. *Phys. Rev. A* **84**, 022305 (2011).
- Sørensen, J., Aranburu, M., Heinzel, T. & Sherson, J. Quantum optimal control in a chopped basis: applications in control of bose-einstein condensates. *Phys. Rev. A* **98**, 022119 (2018).
- Cheeseman, P., Kanefsky, B. & Taylor, W. M. Computational complexity and phase transitions. In *Workshop on Physics and Computation* 63–68 (1992).
- Bukov, M. Reinforcement learning in different phases of quantum control. *Phys. Rev. X* **8**, 031086 (2018).
- Zahedinejad, E., Schirmer, S. & Sanders, B. C. Evolutionary algorithms for hard quantum control. *Phys. Rev. A* **90**, 032310 (2014).
- Moore Tibbetts, K. W. Exploring the tradeoff between fidelity and time optimal control of quantum unitary transformations. *Phys. Rev. A* **86**, 062309 (2012).
- Negrevergne, C. Benchmarking quantum control methods on a 12-qubit system. *Phys. Rev. Lett.* **96**, 170501 (2006).
- Shende, V. V., Bullock, S. S. & Markov, I. L. Synthesis of quantum-logic circuits. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **25**, 1000–1010 (2006).
- Goerz, M. H., Motzoi, F., Whaley, K. B. & Koch, C. P. Charting the circuit qed design landscape using optimal control theory. *npj Quantum Inf.* **3**, 37 (2017).
- Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An introduction*. (MIT Press, Cambridge, MA, 2011).
- Mnih, V. Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).
- Lillicrap, T. P. Continuous control with deep reinforcement learning. Preprint at <https://arxiv.org/abs/1509.02971> (2015).
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. Preprint at <https://arxiv.org/pdf/1707.06347.pdf> Senson (2017).
- Salimans, T., Ho, J., Chen, X., Sidor, S. & Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. Preprint at <https://arxiv.org/abs/1703.03864> (2017).

35. Mania, H., Guy, A. & Recht, B. Simple random search provides a competitive approach to reinforcement learning. Preprint at <https://arxiv.org/abs/1803.07055> (2018).
36. Carleo, G. & Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **355**, 602–606 (2017).
37. Bukov, M. Reinforcement learning to autonomously prepare floquet-engineered states: Inverting the quantum kapitza oscillator. *Phys. Rev. B* **98**, 224305 (2018).
38. 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).
39. Fösel, T., Tighineanu, P., Weiss, T. & Marquardt, F. Reinforcement learning with neural networks for quantum feedback. *Phys. Rev. X* **8**, 031084 (2018).
40. Niu, M. Y., Boixo, S., Smelyanskiy, V. N. & Neven, H. Universal quantum control through deep reinforcement learning. *npj Quantum Inf.* **5**, 33 (2019).
41. Albarrán-Ariagada, F., Retamal, J. C., Solano, E. & Lamata, L. Measurement-based adaptation protocol with quantum reinforcement learning. *Phys. Rev. A* **98**, 042315 (2018).
42. An, Z. & Zhou, D. Deep reinforcement learning for quantum gate control. Preprint at <http://arxiv.org/abs/1902.08418> (2019).
43. Xu, H. et al. Generalizable control for quantum parameter estimation through reinforcement learning. *npj Quantum Inf.* **5**, 1–8 (2019).
44. Amstrup, B., Toth, G. J., Szabo, G., Rabitz, H. & Loerincz, A. Genetic algorithm with migration on topology conserving maps for optimal control of quantum systems. *J. Phys. Chem.* **99**, 5206–5213 (1995).
45. Liebermann, P. J. & Wilhelm, F. K. Optimal qubit control using single-flux quantum pulses. *Phys. Rev. Appl.* **6**, 024022 (2016).
46. Silver, D. Mastering the game of go with deep neural networks and tree search. *Nature* **529**, 484 (2016).
47. Silver, D. Mastering the game of go without human knowledge. *Nature* **550**, 354 (2017).
48. Silver, D. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **362**, 1140–1144 (2018).
49. Clarke, J. & Wilhelm, F. K. Superconducting quantum bits. *Nature* **453**, 1031 EP (2008).
50. Preskill, J. Quantum computing in the nisq era and beyond. *Quantum* **2**, 79 (2018).
51. Magesan, E. & Gambetta, J. M. Effective hamiltonian models of the cross-resonance gate. Preprint at arXiv:1804.04073 (2018).
52. Leek, P. Using sideband transitions for two-qubit operations in superconducting circuits. *Phys. Rev. B* **79**, 180511 (2009).
53. Sheldon, S., Magesan, E., Chow, J. M. & Gambetta, J. M. Procedure for systematically tuning up cross-talk in the cross-resonance gate. *Phys. Rev. A* **93**, 060302 (2016).
54. McDermott, R. & Vavilov, M. Accurate qubit control with single flux quantum pulses. *Phys. Rev. Applied* **2**, 014007 (2014).
55. Li, K., McDermott, R. & Vavilov, M. G. Hardware-Efficient Qubit Control with Single-Flux-Quantum Pulse Sequences. *Phys. Rev. A* **12**, 014044 (2019).
56. Likharev, K. K. Superconductor digital electronics. *Physica C: Superconductivity Appl.* **482**, 6–18 (2012).
57. Sutton, P. & Boyden, S. Genetic algorithms: a general search procedure. *Am. J. Phys.* **62**, 549–552 (1994).
58. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **4**, 65–85 (1994).
59. Motzoi, F., Gambetta, J. M., Merkel, S. T. & Wilhelm, F. K. Optimal control methods for rapidly time-varying hamiltonians. *Phys. Rev. A* **84**, 022307 (2011).
60. Kirchhoff, S. Optimized cross-resonance gate for coupled transmon systems. *Phys. Rev. A* **97**, 042348 (2018).
61. Watkins, C. J. & Dayan, P. Q-learning. *Mach. Learn.* **8**, 279–292 (1992).
62. Day, A. G., Bukov, M., Weinberg, P., Mehta, P. & Sels, D. Glassy phase of optimal quantum control. *Phys. Rev. Lett.* **122**, 020601 (2019).
63. Maaten, Lvd & Hinton, G. Visualizing data using t-sne. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).
64. Bukov, M. Broken symmetry in a two-qubit quantum control landscape. *Phys. Rev. A* **97**, 052114 (2018).
65. Stockfish: Strong open source chess engine. <http://www.stockfishchess.org>.
66. Heck, R. Remote optimization of an ultracold atoms experiment by experts and citizen scientists. *Proc. Natl Acad. Sci. USA* **115**, E11231–E11237 (2018).

ACKNOWLEDGEMENTS

This work was funded by the ERC, H2020 grant 639560 (MECTRL), and the John Templeton and Carlsberg Foundations. The authors would also like to personally thank Jesper H. M. Jensen, Carrie Ann Weidner, and Miroslav Gajdacz for fruitful discussions and input. The numerical results presented in this work were obtained at the Centre for Scientific Computing, Aarhus phys.au.dk/forskning/cscaa.

AUTHOR CONTRIBUTIONS

M.D. and F.M. wrote and implemented the software, performed the simulations, and analyzed the data. All authors contributed to interpreting data as well as participating in useful scientific discussions. J.S. planned and supervised this project. J.J.S. advised the project on a day-to-day basis. All authors contributed to writing this paper.

COMPETING INTERESTS

The authors declare no competing interests.

ADDITIONAL INFORMATION

Supplementary information is available for this paper at <https://doi.org/10.1038/s41534-019-0241-0>.

Correspondence and requests for materials should be addressed to J.S.

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) 2020