Quantum annealing is a practical approach to approximately implement the adiabatic quantum computational model in a real-world setting. The goal of an adiabatic algorithm is to prepare the ground state of a problem-encoded Hamiltonian at the end of an annealing path. This is typically achieved by driving the dynamical evolution of a quantum system slowly to enforce adiabaticity. Properly optimized annealing schedules often considerably accelerate the computational process. Inspired by the recent success of deep reinforcement learning such as DeepMind’s AlphaZero, we propose a Monte Carlo tree search (MCTS) algorithm and its enhanced version boosted by neural networks—which we name QuantumZero (QZero)—to automate the design of annealing schedules in a hybrid quantum–classical framework. Both the MCTS and QZero algorithms perform remarkably well in discovering effective annealing schedules even when the annealing time is short for the 3-SAT examples considered in this study. Furthermore, the flexibility of neural networks allows us to apply transfer-learning techniques to boost QZero’s performance. We demonstrate in benchmark studies that MCTS and QZero perform more efficiently than other reinforcement learning algorithms in designing annealing schedules.

In this work we address one of these challenges by proposing automated designs of annealing schedules using the Monte Carlo tree search (MCTS) and its enhanced version incorporating neural networks (NNs) to further improve the performance. This enhanced version, named QuantumZero (QZero), is inspired by the recent success of DeepMind’s AlphaZero in mastering the game of Go. The proposed methods share many similarities with the design principles of hybrid quantum–classical algorithms for quantum circuits in the noisy intermediate-scale quantum era, especially in a related work that implements a deep quantum exploration version of the AlphaZero algorithm for control problems and achieves substantial improvements in both the quality and quantity of good solution clusters compared with earlier methods. In fact, both approaches can be viewed more broadly as examples of computer-automated experimental designs. A classical subroutine iteratively revises its design of annealing schedules or gate parameters, such that an annealer or a circuit may generate a desired quantum state. This classical subroutine solves an optimization problem with either a gradient-based approach (as commonly adopted in the training of neural networks) or a gradient-free optimizer such as the Bayesian approach, genetic algorithm or evolution strategy. Proposals based on reinforcement learning (RL) to automate experimental designs have recently also emerged as popular alternatives. Conceptually, RL is a machine learning method that learns to accomplish tasks by interacting with an environment as opposed to simply extracting useful patterns from static data. This type of learning process makes RL perform more robustly (in comparison to other machine learning methods) in a noisy and inherently stochastic environment. Reinforcement learning algorithms have been used in many scientific and engineering fields to address difficult problems after witnessing the remarkable accomplishments of AlphaGo and AlphaZero. Yet we have not seen attempts in adopting MCTS (another indispensable ingredient for AlphaGo and AlphaZero) to automate design of annealing schedules. In fact, the underlying search mechanism of MCTS can be viewed as a learning algorithm for the Markov decision process, the central model in RL; therefore, MCTS can perform similar tasks like other RL algorithms. In this work we adopt MCTS and modify the standard AlphaZero to design optimal annealing schedules.

Under the AQC paradigm, a computational problem is framed in such a way that the desired solution corresponds to the ground state of a problem-specific Hamiltonian $H_{\text{ham}}$. Quantum annealing is a heuristic approach to prepare the desired ground state. Typically, the approach begins by initializing a quantum annealer in the ground state of a simple Hamiltonian $\tilde{H}_{\text{ham}}$ (assuming this task can be accomplished efficiently). Next, one slowly tune the Hamiltonian towards $H_{\text{ham}}$. If the dynamical process proceeds slowly enough to largely avoid Landau–Zener transitions to excited states, the adiabatic theorem should be applicable. At the end of the annealing process, the quantum annealer should successfully prepare the ground state of $H_{\text{ham}}$ with high probability. In practice, however, the annealing time cannot be arbitrarily long due to detrimental noises lurking in the

---

1Tencent Quantum Laboratory, Tencent, Shenzhen, Guangdong, China. 2The Chinese University of Hong Kong, Hong Kong, Hong Kong. 3✉E-mail: kimhsieh@tencent.com
background, and the fact that we expect quantum computations to be fast. These conflicting requirements on annealing time constitute a real challenge to keep the quantum annealer in the instantaneous ground state of a time-dependent Hamiltonian with high probability. The difficulty of maintaining the adiabatic condition aggravates tremendously with the problem size; and it becomes crucial to optimize the annealing schedule to improve performance.

In this study we carefully benchmark how the MCTS performs against other RL algorithms in designing annealing schedules. First, we elucidate the advantages shared by MCTS and other RL methods in solving difficult optimization. As gradient-free methods, MCTS and other RL models mitigate the issues of local-minima trapping in a high-dimensional energy landscape. Second, both methods can efficiently handle combinatorial problems involving discrete variables. However, we hypothesize that MCTS should be a more suitable method than other RL techniques for automating quantum experiments (and quantum algorithmic designs) when it is expensive to generate high volume of training data. This hypothesis has been positively validated in our numerical study. The MCTS uses an order of magnitude less queries in finding optimal solution as manifested in the comparison of training efficiency of various algorithms in Fig. 6.

Another major distinction between our proposed methods and past approaches is our treatment of transfer learning. Skill in transfer learning is both remarkable and extremely useful for acquiring optimal solutions efficiently when switching from one scenario to another, as has been studied in many other works. In the case of AlphaZero, once an RL agent is trained to devise strategy under the environment of Go, it is only expected to apply the same strategy over and over again. However, in the context of AQC, every optimization problem is embedded in a different Hamiltonian, resulting in a different learning environment for an RL agent to learn how to prepare the corresponding ground state. Although there are meta-learning strategies that allow an RL agent to adapt to different environments, it typically requires even more training time and data. In this work we propose to simply pre-train QZero’s value and policy NNs with a small set of sample problems (solved by the MCTS) such that one only needs to fine-tune the NNs when the algorithm is applied to a new problem.

Finally, the proposed MCTS approaches for designing annealing schedule may be ported to the quantum circuit model. By drawing the analogy between the quantum approximate optimization algorithm and digitized quantum annealing, it is straightforward to build this connection (see the Supplementary Information for a detailed discussion). Looking more broadly, we also argue that these methods can be generalized for the automated designs for other quantum technologies. Some examples include the quantum control, quantum error corrections, quantum metrology, quantum optics and quantum communications.

Quantum annealing and 3-SAT problem

We first introduce the essential background of the AQC model and elucidate how the design of an annealing schedule can be automated under the RL framework. We next present a constrained optimization problem, 3-SAT, used to benchmark algorithms in this work.

**Annealing schedule as a problem for optimal control.** Quantum annealers are typically used to solve problems under the AQC framework, which relates the solutions of a problem to the ground states of a problem-encoded Hamiltonian, \( H_{\text{init}} \). Preparing the ground state of an arbitrary Hamiltonian is not a simple task. A common approach is to prepare the ground state of an alternative Hamiltonian \( H_{\text{final}} \) that we can experimentally achieve with high success probability. We next slowly tune the time-dependent Hamiltonian \( H(s) \) along a predefined annealing path, towards \( H_{\text{final}} \) at the end. According to the adiabatic theorem, the time-evolved wave function will be highly overlapped with the instantaneous ground state of \( H(s) \). Hence, one expects to retrieve the correct solution at the end of an annealing process with high probability. More precisely, in each AQC calculation, we need to engineer a time-dependent Hamiltonian,

\[
H(s) = (1 - s)H_{\text{init}} + sH_{\text{final}}, \quad s \in [0, 1].
\]

The process of tuning the Hamiltonian has to be implemented slowly in comparison to the timescale set by the minimal spectral gap of \( H(s) \) along the annealing path. The time required to complete an AQC calculation clearly crucially depends on the spectral gap of \( H(s) \). In reality, it is often necessary to finish the calculation within a finite duration, \( T \), for various reasons; for example, expected quantum speedup and minimization of noise-induced errors. This time constraint (on annealing) may violate the adiabatic evolution strictly required by AQC. Nevertheless, one can still run a quantum annealer with some schedule, \( s(t) \), in hope of reaching the ground state of \( H_{\text{final}} \) with high probability. We note this task of optimizing the schedule \( s(t) \) may be framed as an optimal control problem aiming to minimize the energy as the cost function,

\[
\arg\min \langle \psi(T) | H_{\text{final}} | \psi(T) \rangle, \quad s(t)
\]

where \( s(t) : t \in [0, T] \) governs the state evolution \( \{ | \psi(t) \rangle : t \in [0, T] \} \) through the Schrödinger equation \( \dot{|\psi(t)\rangle} = -iH(s(t)) |\psi(t)\rangle \) with a starting state and ground state of \( |\psi(0)\rangle \) and \( H_{\text{final}} \), respectively. We remark that the adiabaticity along the annealing path is not directly reflected or assumed in the cost function, which only depends on the expected energy of the final state, \( |\psi(T)\rangle \). By solving the optimal control problem above, it is likely that an optimal solution would entail a wave function \( |\psi(t)\rangle \) that deviates considerably from the instantaneous ground state along a portion of the annealing path. Usually, guided by the adiabatic theorem, it is desirable to follow the adiabatic trajectories to prepare the ground state of \( H_{\text{final}} \). Yet it has been recently pointed out that arbitrarily long annealing time
3-SAT problem. In this work we use 3-SAT problems to benchmark algorithms. It is a paradigmatic example of a non-deterministic polynomial problem. A 3-SAT problem is defined by a logical statement involving \( n \) boolean variables, \( b_i \). The logical statement consists of \( m \) clauses, \( C_n \), in conjunction: \( C_1 \land C_2 \land \ldots \land C_m \). Each clause is a disjunction of three literals, where a literal is a boolean variable, \( b_i \) or its negation, \( \neg b_i \). For instance, a clause may read \( (b_1 \lor \neg b_2 \lor \neg b_3) \) with three boolean variables having indices \( j, k, l \). The task is to first decide whether a given 3-SAT problem is satisfiable, and if so, then assign appropriate binary values to satisfy the logical statement.

We can map a 3-SAT problem to a Hamiltonian for a set of qubits. Under this mapping, each binary variable \( b_i \) is represented as a qubit state. Thus, an \( n \)-variable 3-SAT problem is mapped into a Hilbert space of dimension \( N = 2^n \). Furthermore, each clause of the logical statement is translated to a projector, which projects out the bitstrings that do not satisfy each given clause. Hence, a logical statement with \( m \) clauses may be translated to the following Hamiltonian,

\[
H_{\text{final}} = \sum_{i=1}^{m} \left| b_i^* b_i^* b_i^* b_i^* \right\rangle \langle b_i^* b_i^* b_i^* b_i^* | + \left| b_i^* b_i^* b_i^* b_i^* \right\rangle \langle b_i^* b_i^* b_i^* b_i^* |
\]

This Hamiltonian is diagonal in the computational basis and the spectrum has a unit gap between eigenvalues. Each of the \( m \) configurations that appear in \( H_{\text{final}} \) specify the violation of a clause in the logical statement. Hence, a solution only exists if the lowest eigenvalue of \( H_{\text{final}} \) is zero. One approach to drive the \( n \)-qubit system to the ground state of \( H_{\text{final}} \) is to use a quantum annealer under the AQF framework.

We next briefly mention other details essential to reproduce the numerical results in this work. Following the standard convention, we choose \( H_m \) for the quantum annealing algorithm to be a sum of one-qubit Hamiltonians \( H_i \) acting on the \( i \)-th qubit:

\[
H_{\text{init}} = \frac{1}{2} \sum_{i=1}^{n} h_i \otimes \mathbb{1}, \quad h_i = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}
\]

The ground state of \( H_{\text{init}} \) has zero energy (that is, \( E_0=0 \)) and is a uniform superposition of all computational states that can be easily prepared by a quantum annealer.

As the computational complexity is defined in terms of the worst-case performance, hard instances of 3-SAT have been intensively studied in the past. Following ref. 60, we focus on a particular set of 3-SAT instances, where each is characterized with a unique solution and a ratio of \( m/n = 3 \). We note that this ratio of three is different from the phase-transition point \( m/n = 4.2 \) (refs. 62–69), which has been intensively explored in studies that characterize the degrees of satisfiability of random 3-SAT problems. The subtle distinction is that the phase-transition point characterizes the notion of hardness (with respect to the \( m/n \) ratio) by averaging over 3-SAT instances having variable number of solutions. However, when the focus is to identify the most difficult 3-SAT instances having unique solution, it has been empirically found that these instances tend to have an \( m/n \) ratio lower than the phase-transition point.

Results

In this section we describe several numerical experiments to illustrate the strengths of our proposed methods.

MCTS-designed annealing schedules. We explain the MCTS-based automated design of annealing schedules for 3-SAT examples. Monte Carlo tree search is extremely efficient at solving high-dimensional optimization problems, the details of which can be seen in the Methods. Usually, a proper choice of the search space would greatly simplify the process. We elaborate on three different domains, one of which may formulate the search problem (Methods). In this work we mainly focus on the design of \( s(t) \) in the frequency domain, as detailed in equation (6).

Following equation (6), the goal is to pick a sequence of \( \{x_i, x_2, x_3, \ldots, x_n\} \) (where \( x_i \) is a control parameter) to minimize the energy with respect to \( H_{\text{final}} \) at the end of an annealing path. As specified in equation (6), each \( x_i \) corresponds to the amplitude of a frequency component when \( s(t) \) is decomposed into a Fourier sine series in the \([0, T]\) domain. As MCTS (Fig. 2) is a search algorithm, we consider \( x_i \) to assume only discretized values of \([-l_i, -l_i + \Delta_n, \ldots, l_i - \Delta_n, l_i]\) where \( l_i \) and \( \Delta_n \) are user-defined boundary value and discretization step, respectively. There is a total of \( \prod_{i=1}^{M} (2l_i/\Delta_n + 1) \) options of \( \{x_i, x_2, x_3, \ldots, x_n\} \) for the MCTS algorithms to explore. For simplicity, we set \( l_i = 1 \) and \( \Delta_n = 1 \) in this study. In particular, we should compare the path designed by our MCTS algorithm with stochastic descent (SD), a greedy method that targets local minima in the energy landscape. The modified MCTS algorithm is presented in the Methods, whereas the SD algorithm is briefly described in the Supplementary Information.

When the overall annealing time \( T \) is sufficiently large with respect to the timescale set by the minimal spectral gap along a given annealing path, almost any schedule, including the linear one—that is, setting \( x_i = 0 \) in equation (6)—leads to a satisfactory solution, with the annealer-prepared quantum state \( |\Psi(T)\rangle \) having a high overlap with \( |\Psi_\text{gs}\rangle \), the ground state of \( H_{\text{final}} \). When \( T \) is not sufficiently long, the linear schedule starts to fail as Landau–Zener transitions are likely to occur when the system passes through the minimal-gap regime; however, when resorting to methods such as MCTS or SD, it is still possible to recover non-linear schedules that greatly suppress the diabatic transitions when the tuning of the time-dependent Hamiltonian operates at a reduced rate around the critical point of minimal gap. We should also note the low-end regime: when \( T \) is further reduced below the threshold of the quantum speed limit (QST), the quantum annealer is no longer
controllable, that is, there is no way to attain perfect fidelity at the end of an annealing process. As we deal with 3-SAT instances with unique solutions in this study, designing an optimal annealing schedule is exactly the same as the optimal control for the state-to-state transition. As discussed in ref. 17, the infidelity for state preparation (as a function of \( x_i \)) transforms to a correlated phase with many non-degenerate local minima scattering around a rugged landscape. Finding a global optimum (without perfect fidelity) clearly becomes extremely difficult in this regime, \( T < T_{\text{critical}} \).

Although the proposed annealing schedules are no longer characterized by adiabatic evolutions, the benchmarks in this section still meaningfully manifest the capability of each algorithm in solving the challenging optimization problems. Before we present our results, we describe the benchmark procedures that we consider as a fair comparison between MCTS and SD. When solving a 3-SAT instance, MCTS has to perform many rounds of simulations as it explores the control space of \( x_i \) and learns to estimate the likelihood of finding a particular annealing schedule being an optimal one. Each instance of this explorative simulation requires feedback from a quantum annealer experiment with a particular annealing path. In comparison, every SD local search (randomly initialized with \( x_i \)) quickly gets stuck in a local minimum in this difficult regime. We argue it is not fair to compare one run of MCTS search with one run of SD search, as the SD tends to query the quantum annealer significantly more times than MCTS in one run. Rather, we will repeat SD many times (initialized with different \( x_i \)) such that the total number of access to a quantum annealer is comparable to that in one MCTS search.

In Fig. 3a, we present the success probability of solving an example of a 3-SAT instance of the same structure \((n = 11\text{ and } m = 33)\) under different \( T \). In this study we fix the number of Fourier components to \( M = 5 \), the bound strength of each Fourier component by \( l = 0.2 \) and set the discretization interval as \( d = 0.01 \). The blue points represent fidelity (or the success probability) of simple linear schedules of different annealing durations. The green points represent the average fidelity of 40 SD search with random initial conditions. The green lines give the error bars associated with SD searches. The red points represent the average fidelity of 80 episodes of a single MCTS search. A single run of SD requires roughly 100 queries to the quantum annealer for energy feedback, whereas an episode of MCTS requires roughly 50 such queries. Thus, to make a fair comparison in terms of queries to quantum annealers, we consider twice as many MCTS episodes as SD runs (that is, \( 40 \times 100 = 80 \times 50 \)). According to Fig. 3a, those large error bars of SD indicate a complex optimization landscape comprising multiple local minima in which SD easily gets stuck in. On the other hand, using roughly the same number of queries to a quantum annealer, the solutions found by MCTS achieve higher successful probability.

In Fig. 3b, we present the success probability of solving several 3-SAT instances with different structures under relatively short annealing times. The results here are presented as relative values of success probability under linear path. We successfully plot the probability of finding the ground state of SD and MCTS via a box plot, which is a systematic way of displaying the data distribution based on five indicators: the minimum, first quartile (Q1), median, third quartile (Q3) and maximum. Comparisons in Fig. 3b are, again, based on having almost the same number of queries to the quantum annealers as explained in the previous paragraph. As shown in the comparisons, when the optimization landscape features many local minima, a local method such as SD has a high probability to get stuck, yet a global method such as MCTS shows resilience and has a better chance to escape from these traps. As the problem size gets larger the optimization landscape is more likely to become more rugged, widening the performance between MCTS and SD (see, for instance, \( n = 11, 13, 15 \) in Fig. 3b).

**Fig. 3** | **Success probability of solving several 3-SAT instances with different structures.** \( a \), The fidelity (or success probability) of obtaining the ground states for an example of a 3-SAT instance (composed of \( n = 11 \) variables) in a quantum annealer evolved under different \( T \). The red points denote the statistical fluctuations of SD and MCTS results. \( b \), The success probability of obtaining the ground states for 3-SAT instances (composed of \( n/m = 7/21; 9/27; 11/33; 13/39 \) and 15/45 variables) in a quantum annealer evolved under various annealing durations \((T = 25, 40, 200, 300, 1,000, \text{respectively})\). Here we present the results as relative values of success probability compared to linear path \( F_{\text{lin.}} \).

**Transfer of annealing schedules.** As demonstrated in the previous section, MCTS gives higher-quality solutions than SD, which holds even if SD is given multiple chances with different initial conditions to facilitate the exploration of the solution space. Nevertheless, a single run of MCTS still requires repeated episodes to balance the trade-off of exploration and exploitation. In the near term, quantum resources are expensive, hence it is desirable to seek alternatives that could minimize dependence on a quantum annealer. To this end, we resort to recent developments that combine MCTS with neural networks.

It is highly desirable if MCTS can learn from accumulated experiences of solving similar problems in the past. In the field of deep learning, a similar goal is achieved for NNs via transfer learning. For instance, NNs pre-trained on a large dataset can be easily adapted to predict the properties of a small dataset. Inspired by this flexibility of NNs, we further modify MCTS by incorporating NNs, as done in DeepMind’s AlphaZero; however, the off-the-shelf AlphaZero is not a suitable model for our purpose. For instance, AlphaZero only needs to learn to win the game of Go under one set
of rules, but we need an algorithm that prepares the ground state of multiple Hamiltonians (analogous to different rules for the game). Another issue is that AlphaZero needs to find a winning strategy for a two-player game while there are no such competitions in our scenario. Several modifications are required before AlphaZero could be used for quantum annealing (details on these modifications can be found in the Methods). For clarity, we name the adapted method QuantumZero (QZero).

Here we investigate the effectiveness of transferring an annealing schedule learnt from a set of training instances to a set of test instances under three different scenarios. The idea is that we first use MCTS to solve some sample instances similar to the actual problems we are interested in. The optimal solution returned by the MCTS is then used in three different ways to guide the search for annealing schedules for new instances. In the first scenario, we simply solve one sample instance and apply the same annealing schedule to a set of test instances. In the second scenario, we transfer an average optimal annealing schedule to test instances. Here the average optimal annealing schedule is found by using MCTS to search for a schedule that gives highest average success probability for multiple sample instances. In the third scenario, we construct a training dataset out of optimal solutions for sample instances to train the policy and value neural networks for QZero. When feeding QZero with new test instances, QZero still conducts a few rounds of MCTS to fine tune the neural networks before settling on optimal solutions. As explained in the Methods, the pre-training of policy and value NNs is a relatively simple computational task as it is formulated as standard supervised learning.

In Fig. 4a–d, we present numerical study on the transferability of optimal annealing schedules across 3-SAT instances with different annealing duration $T = 40, 60, 80, 100$. We consider a sample set of 45 training cases and a test set of 280 examples; all problem instances share the same number of variables $n = 7$ and same number of clauses $m = 21$. For the first scenario, in each annealing...
duration considered, we randomly select an MCTS-found schedule \( \mathbf{x} \) for a particular training example and apply this schedule to all test cases. The results are plotted as pink-coloured distributions in Fig. 4a–d. For the second scenario, under different annealing durations, we take an average optimal annealing schedule (that gives the highest average success probability of all 45 sample instances) and apply it to test samples (these results are green in Fig. 4e–f). Finally, yellow results are given by QZero pre-trained with 45 training cases. We caution that the reported results given by QZero are obtained after a few rounds of fine tuning the neural networks. For comparisons, we also plot the results from the naïve linear schedule to all test cases under different annealing durations (see the grey distributions in Fig. 4a–d). Going from long \( T = 100 \) to short \( T = 40 \) duration, it becomes progressively harder to achieve high success probability with the naïve linear schedules. The pink results (a non-linear schedule adapted from a random instance) generally perform better than the linear schedule. This excellent transferability of a single annealing schedule is explained at the end of this section. Next, green results given by the average optimal annealing schedule manifests high percentage of obtaining a satisfying solution to any peculiarity associated with individual test cases. Finally, the pre-trained QZero (yellow) gives the best results for all annealing durations. We reiterate that one needs to perform some light training to fine tune QZero’s value and policy NNs for each test instance.

We next investigate the transferability of annealing schedules (fixed at \( T = 80 \)) across 3-SAT instances with different \( (n, m) \) parameters. In Fig. 4e,f, the applicability of transferring knowledge gained from optimal schedules for 45 training samples \((n = 7, m = 21)\) to 350 test samples \((n = 7, m = 18)\) and from 45 training samples \((n = 7, m = 21)\) to 350 test samples \((n = 7, m = 23)\). We again consider three different strategies to use the knowledge obtained from the training set. It is obvious that the success probability using the optimal path transferred from a single training instance (pink) is higher than using a linear path (grey). In turn, the success probability of solving new test instances with the average optimal schedule (green) is higher than that of the optimal path of a single instance (pink). If we pre-train QZero’s policy and value NNs, the results are again the best among all scenarios considered. To address the concern (whether pre-train really accelerates the search) of having to fine tune QZero’s NNs, we investigate the training efficiency of QZero in the next subsection.

We finally return to the transferability of annealing schedules across 3-SAT problems. In Fig. 5, we present the distribution of min-gaps (the smallest energy gap between the first excited state and the ground state of an instantaneous Hamiltonian along annealing paths) for the 3-SAT instances used in Fig. 4a–f. As seen, all of these instances have their min-gap at around \( s = 0.6 \), with a rather restricted energy range. This high similarity of the min-gap structure along different annealing paths is responsible for the high transferability of annealing schedules, even across instances without sophisticated treatments (as shown by the pink results in Fig. 4a–f); however, this does not imply that these instances are nearly trivially identical. In the Supplementary Information we further analyse the optimal pulse profiles for some randomly chosen instances with \( m/n = 3 \). It is clear that these pulse profiles look sufficiently distinct, which implies these instances also possess their own unique gap profiles along the annealing paths. This also explains why the transferability drops when \( T = 40 \) is small except for the QZero model, which performs the standard transfer learning with additional learning steps to fine tune the designed path for each individual problem.

The differences between the ground-state energy and the expected energy of the time-evolved quantum state following SD or QZero annealing schedules are carefully investigated in Fig. 5a,b, respectively. The energy difference \( \Delta E \) reflects how strongly the adiabaticity is violated along different paths. As shown, the pre-trained QZero is not only able to find optimal solutions but also to enforce adiabaticity better than SD.

Comparing learning efficiency of QZero and other RL methods. Finally, we compare the learning efficiency of QZero with other popular RL methods mentioned in the introduction. Similar to QZero, these RL methods are capable of finding the global optimum, even for difficult problems such as the ones discussed in the previous sections; however, training typical RL methods is notoriously resource consuming. Here we demonstrate that QZero achieves the same level of performance (as other RL methods) using less computational resources. In particular, our assessment is based on the number of queries to a quantum annealer required by each method. In this benchmark, we compare two variants of MCTS algorithms, QZero with pre-training (QZero-pre) and QZero without pre-training (QZero-nopre) with three other RL models—deep Q-networks (DQN)\(^{47,48}\), Advantage Actor-Critic (A2C)\(^{44}\) and proximal policy optimization (PPO)\(^{49}\). See the Supplementary Information for details on these three RL algorithms.

We use two 3-SAT examples—denoted by \( H_{\text{NQAD}} \) and \( H_{\text{JQAD}} \)—of size \( n = 7, m = 21 \), as benchmark for this efficiency test to design an annealing schedule with duration \( T = 70 \). We formulate all RL algorithms to possess an identical set of actions for designing the Fourier components of all allowable schedules defined in equation (6). In particular, we consider five frequency components, \( M = 5 \), and each coefficient \( x \) belongs to a discretized space of \([-l, l + \Delta, \ldots, l - \Delta, l]\), where \( l = 0.2 \) and \( \Delta = 0.01 \). The efficiency test is summarized in Fig. 6. We look at how fast each algorithm finishes its training and returns an optimal solution. In Fig. 6, a query specifically refers to operating a quantum annealer with an annealing schedule to provide feedback. To make fair comparisons, the queries hidden inside of the simulation playouts of QZero are explicitly taken into account. As manifested in Fig. 6, QZero-nopre performs more efficiently than all of the other RL methods (DQN, PPO, A2C) as the MCTS performs efficient searches. QZero-pre
In this study we have demonstrated that MCTS outperforms the stochastic descent—a local search algorithm—when addressing tough problems descent—a local search algorithm—when addressing tough problems due to complex energy landscape. We also compare MCTS and QZero with a host of other RL algorithms that recently attracted a lot of attention due to their potential to improve quantum annealing when solving combinatorial problems. We have found that MCTS and QZero outperform all other RL algorithms considered in our benchmark study. In particular, the pre-trained QZero turned out to be the most efficient among all RL algorithms reported in this work. Our work shows that MCTS and QZero are highly competitive methods for automating designs of quantum annealing schedules.

**Methods**

In this section, we introduce our proposed strategy to design \( s(t) \) with MCTS and QZero.

**MCTS.** Monte Carlo tree search aims to find a vector of discrete variables \( x \) that maximizes or minimizes a target property \( f(x) \) evaluated by a problem-specific learning environment. For designing an annealing schedule, \( x = [x_1, x_2, \ldots, x_M] \) corresponds to coefficients of Fourier series introduced in equation (6). Each \( x_i \) is a binary variable that can be set to \( 1 \) or \( -1 \). The upper and lower bound for the amplitudes of \( i \)-th frequency component, and \( \Delta_i \) is the discretized increment in the frequency space. The whole search space is composed of \( 2^M(2\Delta_i + 1) \) grid points. In our case, \( x_1 = (\psi_1(T)/|\psi_{final}(T)|, \psi_1(T)) \) is the expected energy, where \( |\psi_1(T)| \) is the time-evolved quantum state at the end of an annealing path.

Monte Carlo tree search performs the search on an \( (M+1) \)-level tree structure. The zeroth level is just a root node, which denotes a starting point and carries no information. The nodes at the \( k \)-th level correspond to the \( (2\Delta_i+1) \)-value assignments of \( x_i \) with \( k = 1, \ldots, M \). Every solution \( x \) specifies a path along the tree from root to bottom. In Fig. 2, we illustrate how an MCTS search is conducted on a \( (3+1) \)-level tree composed of three nodes on each level. Ignoring the zeroth level, the tree structure actually looks like the \( 3 \times 3 \) square board shown in Fig. 2. The MCTS starts at the root and traverses the tree level by level. The algorithm has to select a node (blue boxes in Fig. 2) before proceeding to the next level. As illustrated in Fig. 2, the MCTS decides a path by sequentially inserting \( x_i \), \( x_2 \) and \( x_3 \) into an array specifying the path \([j_1, j_2, j_3, j_4, j_5, j_6] \). Each round of MCTS consists of four stages: selection, expansion, simulation, and back-propagation. In the selection stage, a path is traversed from the root down to a node \( x_\text{a} \) at the \( k \)-th level by choosing the nodes \( x_i \) (with \( k \leq 5 \)) with a maximum upper confidence bound (UCB) score at each level \( x_i = \max_{x_i} \) where the maximum is over candidate actions \( x \). The UCB score indicates how promising it is to explore the subtree under the current node and is defined as

\[
\text{UCB} = \text{AverageMerit} + C \times \sqrt{\frac{2 \log N_{\text{parent}}}{N_{\text{node}}}}
\]

where the visit count \( N_{\text{node}} \) denotes the number of visits to node \( a \) during the search process, \( N_{\text{parent}} \) is the visit count of the parent node, the cumulative merit \( w_a \) is defined as the sum of all direct merits for all descendant nodes including itself, and \( C \) is a constant that balances the exploration and exploitation. This traversal terminates at the \( k \)-th level when all its children nodes have not been visited before. At this point, the search enters the expansion stage. \( N_{\text{node}} \) new children nodes are added under the current node \( x_i \); with the following initializations: \( v_a = w_a = f = 0 \), \( N_{\text{node}} = N_{\text{final}} \) relevant for the UCB score. Once new children nodes are created, the search transits to the simulation stage. \( N_{\text{sim}} \) times of random playouts are performed for each added node. A playout is a random selection of additional nodes to form a complete path from top to bottom, the \( m \)-th level. Once such a path has been randomly pick, \( f(x) = (\psi_1(T)/|\psi_{final}(T)|, \psi_1(T)) \) is evaluated and recorded as an immediate merit of the path. In the final stage of back propagation, the visit count of each ancestor nodes of \( x_i \) is incremented by one and the cumulative value is also updated to maintain consistency. We repeatedly run this four-phase search for a fixed number of times. At the end, the best solution would be returned as the final result. The random playouts of MCTS allow us to efficiently explore a large set of candidate solutions and identify promising directions to focus the search for optimal solutions.

For the experiments reported in the main text, we set \( C = 2 \) to balance the exploration and exploitation, the number of nodes added at each expansion \( N_{\text{node}} = 10 \), and the simulation times at node \( N_{\text{sim}} = 5 \).

**Search space for annealing schedules.** Monte Carlo tree search and related modern methods are extremely efficient at solving high-dimensional optimization problems that we encounter in this work; however, depending on the context of the problem at hand, a proper choice of the search space may greatly simplify the process. In this section, we elaborate on three different domains, in which one may formulate the search problem:

1. **Time domain.** Directly designing \( s(t) \) in the time domain is a straightforward idea. The optimal control problem is now turned into assigning values (from a predefined range) to a sequence \( x = [x(0), x(\Delta t), x(2\Delta t), \ldots, x(T)] \) after uniformly dividing the evolution time \( T \) into small segments of \( \Delta t \).\n
\[
s(t) = s_i(t) = \sum_{i=1}^{M} \alpha_i \sin \left( \frac{i \pi t}{T} \right)
\]

2. **Frequency domain.** \( s(t) \) can also be Fourier expanded around a monotonically increasing schedule, such as \( s_i(t) = i t / T \), in the frequency domain:

\[
s(t) = s_i(t) = \sum_{i=1}^{M} \alpha_i \sin \left( \frac{i \pi t}{T} \right)
\]
This is because we only consider the winning strategy for each sample instance in the pre-train dataset. The value and policy neural network are then trained as a supervised-learning task,

\[
(p', v') = G_{\theta}(s'),
\]

where \(\theta\) represents the weights of the neural network \(G\). These pre-trained NNs can be easily incorporated into MCTS as discussed below.

Although QZero is pre-trained, the NNs still require fine tuning when applied to a new problem instance. The training process proceeds in two stages. First, MCTS equipped with pre-trained NNs goes through a modified search procedure (same as AlphaZero) multiple times, picks new annealing schedule \(\chi\) each time, and obtains corresponding evaluation \(v\) given by the learning environment. In the second stage, this set of collected data \([s, v]\) is used to further train neural networks by following the AlphaZero algorithm. The detail is provided at the end of the next paragraph.

The four-stage MCTS is modified to make use of the action distribution and state value estimated by NNs as a guidance for selecting path traversal. The streamlined QZero comprises of a three-step procedure: selection, expansion with evaluation, and back propagation. The selection step relies on a score function to decide a path traversal along the tree structure. The modified score function reads

\[
U_{\tau,a} = \frac{W_{\tau,a}}{N_{\tau,a}} + C p - \sqrt{\frac{\sum_{s} N_{\tau s}}{1 + N_{\tau,a}}},
\]

where \(a\) and \(a'\) represent the candidate nodes (that could be appended to extend the current path \(\tau\)) at this selection step, the visit count \(N\) represents the visit times in the search process, \(\sum_{s} N_{\tau s}\) is the visit count of the parent node, the cumulative merit \(W\) is defined as the sum of all cumulative merits for its descendant nodes including itself. The direct merit here is the value \(v\) estimated by the value NN for a partial game or otherwise \(z\) for a win or loss for a complete game: \(p'\) is the policy value given by the policy NN; \(C\) is a constant to balance the exploration and exploitation. Repeating the selection step until arriving at a leaf node, the algorithm then expands the tree to the next level. Each leaf node at the new level is evaluated by the direct merit \(v\) defined earlier, and this merit is back propagated to update the cumulative merits \(W\) for all its parent nodes along the search tree. After \(N_{\tau s}\) simulations, QZero makes an actual move based on a new policy distribution, \(\pi\), which is updated with the frequency counts of attempted actions during simulations. An episode is finished when QZero makes a sequence of actual moves to fully specify an annealing schedule, that is, it reaches the bottom of the search tree. The feedback (whether the time-evolved state has a small enough energy at the end, a win-or-loss situation) by the quantum annealer essentially produces updated values \(z\) for all explored partial or full annealing schedules in this episode. After playing through a fixed number of episodes, the collected set of data is then subsequently used to retrain the neural networks by minimizing the following loss function.

\[
I = (z - v)^2 - \frac{\sum_{\tau} N_{\tau a}}{1 + N_{\tau a}} \theta^2,
\]

where \(\lambda\) corresponds to the regularization strength of NN weights. After calibrating the NNs with updated data, we carry out another round of MCTS guided by the new policy and value. By repeating this process of MCTS and calibrating NNs, a steady state could be reached, where the MCTS captures an optimal search strategy and training of neural networks converges with the loss of equation (10) tending to zero.

Finally, we report hyper-parameters for the QZero used in the section Result. We initialize the constant to balance the exploration and exploitation at \(C = 3\) and gradually decrease it to \(C = 0.5\), the number of simulations before each move is \(N_{\tau s} = 6\), policy NN has three dense layers of dimension \([256, 128, 2\Delta M]\), and value NN has four dense layers of dimension \([256, 128, 64, 1]\), learning rates for NN start at \(l = 0.0008\) and gradually decay to \(l = 0.0008\), and the energy error is set to \(e = 0.01\).

Data availability

The data that support the results in this paper is available at https://github.com/yutter21/quantumzero.

Code availability

The code that support the results in this paper is available at https://github.com/yutter21/quantumzero.git.

Received: 21 April 2021; Accepted: 12 January 2022; Published online: 14 March 2022

References

1. Jiang, S., Britt, K. A., McCaskey, A. J., Humble, T. S. & Kais, S. Quantum annealing for prime factorization. Sci. Rep. 8, 17667 (2018).
2. King, A. D. et al. Observation of topological phenomena in a programmable lattice of 1,800 qubits. Nature 560, 456–460 (2018).
3. Harris, R. et al. Phase transitions in a programmable quantum spin glass simulator. Science 361, 162–165 (2018).
4. Willisch, D., Willisch, M., De Raedt, H. & Michielsen, K. Support vector machines on the D-wave quantum annealer. Comput. Phys. Commun. 248, 107006 (2020).
5. Mott, A., Job, J., Vilmant, Jean-Roch, Lidar, D. & Spitrov, M. Solving a Higgs optimization problem with quantum annealing for machine learning. Nature 550, 375–379 (2017).
6. Li, R. Y., Di Felice, R., Roos, R. & Lidar, D. A. Quantum annealing versus classical machine learning applied to a simplified computational biology problem. npj Quant. Inf. 4, 14 (2018).
7. Hormouzi, L., Brown, E. W., Carleo, G. & Troyer, M. Nonstoquastic Hamiltonians and quantum annealing of an Ising spin glass. Phys. Rev. B 95, 184416 (2017).
8. Herr, D. et al. Optimizing schedules for quantum annealing. Preprint at https://arxiv.org/abs/1705.00420 (2017).
9. Zeng, L., Zhang, J. & Sarovar, M. Schedule path optimization for adiabatic quantum computation and optimization. J. Phys. A 49, 165305 (2016).
10. Susa, Y., Yamashiro, Y., Yamamoto, M. & Nishimori, H. Exponential speedup of quantum annealing by inhomogeneous driving of the transverse field. J. Phys. Soc. Jpn 87, 023002 (2018).
11. Albash, T. & Lidar, D. A. Adiabatic quantum computation. Rev. Mod. Phys. 90, 015002 (2018).
12. Hauke, P. et al. Perspectives of quantum annealing: Methods and implementations. Rep. Prog. Phys. 83, 054401 (2020).
13. Farhi, E., Goldstone, J., Gutmann, S. & Sipser, M. Quantum computation by adiabatic evolution. Preprint at https://arxiv.org/abs/quant-ph/0001106 (2000).
14. Farhi, E. et al. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. Science 292, 472–475 (2001).
15. Das, A. & Chakrabarti, B. K. Quantum Annealing and Related Optimization Methods (Springer, 2005).
16. Childs, A. M., Farhi, E. & Preskill, J. Robustness of adiabatic quantum algorithms. Phys. Rev. A 65, 012302 (2002).
17. Aharonev, D. et al. Adiabatic quantum computation is equivalent to standard quantum computation. SIAM Rev. 50, 755–787 (2008).
18. Susa, Y. & Nishimori, H. Variational optimization of the quantum annealing schedule for the Lechner–Hauke–Zoller scheme. Phys. Rev. A 103, 022619 (2021).
19. Herr, D. et al. Optimizing schedules for quantum annealing. Preprint at https://arxiv.org/abs/1705.00420 (2017).
20. Schifer, B. F., Turá, J. & Cirac, J. I. Adiabatic spectroscopy and a variational quantum adiabatic algorithm. Preprint at https://arxiv.org/abs/2103.01226 (2021).
21. Beixo, S. et al. Eigenpath traversal by phase randomization. Quant. Inf. Comput. 9, 833–855 (2009).
22. Coulom, R. in Computers and Games 72–83 (Springer, 2007); https://doi.org/10.1007/978-3-540-75358-8_7
23. Kocsis, L. & Szepesvári, C. Bandit based Monte-Carlo planning. In Artificial Intelligence and Machine Learning (2006).
24. Lee, Chang-Shing et al. The computational intelligence of Mogo revealed in Chess, Shogi, and Go through self-play. Science 550, 354–359 (2017).
25. Silver, D. et al. Mastering the game of go without human knowledge. Nature 529, 484–489 (2016).
26. Bello, T. A. et al. Determination of adiabatic variational quantum computing with a superconducting quantum coprocessor. Phys. Rev. Lett. 125, 180501 (2020).
27. Dalgaard, M., Motzoi, F., Sørensen, J. J. & Sherson, J. Global optimization of quantum dynamics with AlphaZero deep exploration. npj Quant. Inf. 6, 6 (2020).
28. Dalgaard, M., Motzoi, F., Sørensen, J. J. & Sherson, J. Global optimization of quantum dynamics with AlphaZero deep exploration. npj Quant. Inf. 6, 6 (2020).
29. Zhang, X.-M., Wei, Z., Asad, R., Yang, X.-C. & Wang, X. When reinforcement learning stands out in quantum control? A comparative study on state preparation. npj Quant. Inf. 5, 83 (2019).
30. Chen, C., Dong, D., Li, H.-X., Chu, J. & Tar, T.-J. Fidelity-based probabilistic Q-learning for control of quantum systems. IEEE Trans. Neural Netw. Learn. Syst. 25, 920–933 (2014).
31. Bukov, M. et al. Reinforcement learning in different phases of quantum control. Phys. Rev. X 8, 031086 (2018).
32. Niu, M.-Y., Boixo, S., Neven, H. & Neven, H. Universal quantum control through deep reinforcement learning. npj Quant. Inf. 5, 33 (2019).
33. McKeever, K. A., Davis, E., Alam, M. S. & Rigetti, C. Automated quantum programming via reinforcement learning for combinatorial optimization. Preprint at https://arxiv.org/abs/1908.08054 (2019).
34. Khairy, S., Shayanlou, R., Cincio, L., Alexe, Y. & Balaprakash, P. Reinforcement-learning-based variational quantum circuits optimization for combinatorial problems. Preprint at https://arxiv.org/abs/1911.04574 (2019).
35. Lin, J., Lai, Z. Y. & Li, X. Quantum adiabatic algorithm design using reinforcement learning. Phys. Rev. A 101, 052327 (2020).
36. Belorodov, D. et al. Reinforcement learning enhanced quantum-inspired algorithm for combinatorial optimization. Mach. Learn. Sci. Technol. 2, 025009 (2020).
37. Ayanzadeh, R., Halem, M. & Finin, T. Reinforcement quantum annealing: a quantum-assisted learning automata approach. Preprint at https://arxiv.org/abs/2001.00234 (2020).
38. Sutton, R. S. and Barto, A. G. Reinforcement Learning: An Introduction (Bradford, 2018).
39. van Otterlo, M. & Wiering, M. in Reinforcement Learning: An Introduction (Springer, 2012); https://doi.org/10.1007/978-3-642-27645-3_1
40. Minn, V. et al. Playing Atari with deep reinforcement learning. Preprint at https://arxiv.org/abs/1312.5602 (2013).
41. Minn, V. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).
42. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. Preprint at https://arxiv.org/abs/1707.06347 (2017).
43. Vodopivec, T., Samothrakis, S. & Ster, B. On Monte Carlo tree search and reinforcement learning. J. Artif. Intell. Res. 60, 881–936 (2017).
44. van Otterlo, M. & Wiering, M. in Reinforcement Learning: An Introduction (Springer, 2012); https://doi.org/10.1007/978-3-642-27645-3_1
45. Xu, H. et al. Generalizable control for quantum parameter estimation through reinforcement learning. npj Quant. Inf. 5, 82 (2019).
46. Wallnöfer, J. et al. Machine learning for long-distance quantum communication. PRX Quantum 1, 010301 (2020).
47. Karanikolas, V. & Kawabata, S. Improved performance of quantum annealing by a diabatic pulse application. Preprint at https://arxiv.org/abs/1806.08517 (2018).
48. King, J. et al. Quantum annealing amid local ruggedness and global frustration. J. Phys. Soc. Jpn 88, 061007 (2019).
49. Hogg, T. Adiabatic quantum computing for random satisfiability problems. Phys. Rev. A 84, 022326 (2011).
50. Fösel, T., Tighineanu, P. & Reitzmayer, J. Determining computational complexity of characteristic ‘phase transitions’. Nature 400, 133–137 (1999).
51. Brockman, G. et al. OpenAI Baselines (GitHub, 2017); https://github.com/openai/baselines
52. Lloyd, S. Quantum approximate optimization is computationally universal. Preprint at https://arxiv.org/abs/1812.11075 (2018).
53. Farhi, E., Goldstone, J. & Gutmann, S. Quantum adiabatic evolution algorithms versus simulated annealing. Preprint at https://arxiv.org/abs/quant-ph/0201031 (2002).
68. Kong, L. & Crosson, E. The performance of the quantum adiabatic algorithm on spike Hamiltonians. *Int. J. Quant. Inf.* **15**, 1750011 (2017).
69. Roland, J. & Cerf, N. J. Quantum search by local adiabatic evolution. *Phys. Rev. A* **65**, 042308 (2002).
70. Brady, L. T., Baldwin, C. L., Bapat, A., Kharkov, Y. & Gorshkov, A. V. Optimal protocols in quantum annealing and quantum approximate optimization algorithm problems. *Phys. Rev. Lett.* **126**, 070505 (2021).
71. Chen, Y. et al. yutuer21/quantumzero: Quantumzero (Zenodo, 2021); https://doi.org/10.5281/zenodo.5749588

Author contributions
Y.Q.C. performed the simulations and analysed the data. Y.Q.C. and Y.C. wrote the code. All authors contributed to interpreting data and engaged in useful scientific discussions. C.Y.H. conceived and supervised this project. All authors contributed to the writing.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-022-00446-y.
Correspondence and requests for materials should be addressed to Chang-Yu Hsieh.
Peer review information *Nature Machine Intelligence* thanks Evert van Nieuwenburg and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.
Reprints and permissions information is available at www.nature.com/reprints.
Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
© The Author(s), under exclusive licence to Springer Nature Limited 2022