Finding the ground state of spin Hamiltonians with reinforcement learning

Kyle Mills, Pooya Ronagh and Isaac Tamblyn

Reinforcement learning (RL) has become a proven method for optimizing a procedure for which success has been defined, but the specific actions needed to achieve it have not. Using a method we call ‘controlled online optimization learning’ (COOL), we apply the so-called ‘black box’ method of RL to simulated annealing (SA), demonstrating that an RL agent based on proximal policy optimization can, through experience alone, arrive at a temperature schedule that surpasses the performance of standard heuristic temperature schedules for two classes of Hamiltonians. When the system is initialized at a cool temperature, the RL agent learns to heat the system to ‘melt’ it and then slowly cool it in an effort to anneal to the ground state; if the system is initialized at a high temperature, the algorithm immediately cools the system. We investigate the performance of our RL-driven SA agent in generalizing to all Hamiltonians of a specific class. When trained on random Hamiltonians of nearest-neighbour spin glasses, the RL agent is able to control the SA process for other Hamiltonians, reaching the ground state with a higher probability than a simple linear annealing schedule. Furthermore, the scaling performance (with respect to system size) of the RL approach is far more favourable, achieving a performance improvement of almost two orders of magnitude on L = 142 systems. We demonstrate the robustness of the RL approach when the system operates in a ‘destructive observation’ mode, an allusion to a quantum system where measurements destroy the state of the system. The success of the RL agent could have far-reaching impacts, from classical optimization, to quantum annealing and to the simulation of physical systems.

The process of annealing is used in metallurgy and materials science to equilibrate the positions of atoms to obtain perfect low-energy crystals. Heat provides the energy necessary to break atomic bonds, and high-stress interfaces are eliminated by the migration of defects. By slowly cooling the metal to room temperature, the metal atoms become energetically locked in a lattice structure more favourable than the original structure. Metallurgists can tune the temperature schedule to arrive at final products that have desired characteristics, such as ductility and hardness. Annealing is a biased stochastic search for the ground state.

An analogous in silico technique, simulated annealing (SA), can be used to find the ground state of spin glass models, an NP-hard problem (NP, non-deterministic polynomial time). A spin glass is a graphical model consisting of binary spins $\sigma$. The connections between spins are defined by the coupling constants $J_{ij}$, and a linear term with coefficients $h_i$, can apply a bias to individual spins. The Hamiltonian

$$H = -\sum_{i<j} J_{ij} \sigma_i \sigma_j - \sum_i h_i \sigma_i, \quad \sigma_i = \pm 1$$

defines the energy of the microstates. The choices of the quadratic coupling coefficients $J_{ij}$ and the linear bias coefficients $h_i$ effect the interesting dynamics of the model: $J_{ij}$ can be randomly distributed according to a Gaussian distribution, encompass all $i, j$ combinations for a fully connected Hamiltonian, or be limited to short-range (for example, nearest-neighbour, $(i, j)$) interactions, to name a few. For example, when the positive, unit-magnitude coupling is limited to nearest-neighbour pairs, the ubiquitous ferromagnetic Ising model is recovered. Examples of the Hamiltonians we investigate in this work are presented in Fig. 1 and discussed in further detail in the ‘Hamiltonians’ section.

Finding the ground state of such systems (that is, ‘solving’) is interesting from the perspective of thermodynamics, as one can observe phenomena such as phase transitions, but it is also practically useful as discrete optimization problems can be mapped to spin glass models (for example, the travelling salesperson problem or the knapsack problem). The Metropolis–Hastings algorithm can be used to simulate the spin glass at arbitrary temperature, $T$; thus, it is used ubiquitously for SA. By beginning the simulation at a high temperature, one can slowly cool the system over time, providing sufficient thermal energy to escape local minima, and arrive at the ground state ‘solution’ to the problem. The challenge is to find a temperature schedule that minimizes computational effort while still arriving at a satisfactory solution; if the temperature is reduced too rapidly, the system will become trapped in a local minimum, and reducing the temperature too slowly results in an unnecessary computational expense. Kirkpatrick and colleagues proposed starting at a temperature that results in an 80% acceptance ratio (that is, 80% of Metropolis spin flips are accepted) and reducing the temperature geometrically. They also recommended monitoring the objective function and reducing the cooling rate if the objective value (for example, the energy) drops too quickly. More sophisticated adaptive temperature schedules have been investigated; however, simple linear and reciprocal temperature schedules are commonly used in practice. We will refer to SA using a linear schedule as ‘classic SA’ throughout this work. Nevertheless, in his 1987 paper, Bounds said that ‘choosing an annealing schedule for practical purposes is still something of a black art’.

1QB Information Technologies (1QBit), Vancouver, British Columbia, Canada. 2University of Ontario Institute of Technology, Oshawa, Ontario, Canada. 3Vector Institute for Artificial Intelligence, Toronto, Ontario, Canada. 4Institute for Quantum Computing (IQC), Waterloo, Ontario, Canada. 5Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario, Canada. 6National Research Council Canada, Ottawa, Ontario, Canada.

e-mail: kyle.mills@1qbit.com; pooya.ronagh@1qbit.com; isaac.tamblyn@nrc.ca
schedules for various problem Hamiltonians. The schedules that the RL agent produces are dynamic and reactive, adjusting to the current observations of the system to reach the ground state quickly and consistently without an a priori knowledge of a given Hamiltonian. We believe that RL will be important for quantum information processing, especially for hardware- and software-based control.

The environment and architecture

Reinforcement learning. Reinforcement learning is a branch of dynamic programming whereby an agent, residing in state \( s \), at time \( t \), learns to take an action \( a \) that maximizes a cumulative reward signal \( R \) by dynamically interacting with an environment. Through the training process, the agent arrives at a policy \( \pi \) that depends on some observation (or ‘state’) of the system, \( s \). In recent years, neural networks have taken over as the de facto function approximator for the policy. Deep reinforcement learning has seen unprecedented success, achieving superhuman performance in a variety of video games, board games, and other puzzles. Although many RL algorithms exist, we have chosen to use proximal policy optimization (PPO), implemented within Stable Baselines for its competitive performance on problems with continuous action spaces.

The environment. We developed an OpenAI gym environment, which serves as the interface to the ‘game’ of simulated annealing. Let us now define some terminology and parameters important to SA. For a given Hamiltonian, defining the interactions of \( L \) spins, we create \( N_{\text{rep}} \) randomly initialized replicas (unless otherwise specified). The initial spins of each replica are drawn from a Bernoulli distribution with probability of a spin-up being randomly drawn from a uniform distribution. These independent replicas are annealed in parallel. The replicas follow an identical temperature schedule with their uncoupled nature providing a mechanism for the statistics of the system to be represented through an ensemble of measurements. In the context of the Metropolis–Hastings framework, we define one ‘sweep’ to be \( L \) proposed random spin flips (per replica), and one ‘step’ to be \( N_{\text{rep}} \) sweeps. After every step, the environment returns an observation of the current state \( s \) of the system, an \( N_{\text{rep}} \times L \) array consisting of the binary spin values present. This observation can be used to make an informed decision of the action \( a \) that should be taken. The action, a single scalar value, corresponds to the total inverse temperature change \( \Delta \beta \) (where \( \beta = 1/T \)) that should be carried out over the subsequent step. The choice of action is provided to the environment, and the process repeats until \( N_{\text{step}} \) steps have been taken, comprising one full anneal, or ‘episode’ in the language of RL. If the chosen action would result in the temperature becoming negative, no change is made to the temperature and the system continues to evolve under the previous temperature.

In our investigations, we choose \( N_{\text{rep}} = 40 \) and \( N_{\text{steps}} = 100 \), resulting in 4,000 sweeps per episode. These values define the maximum size of system we can compare to classic SA. This number of sweeps is sufficient for a linear schedule to attain measurable success on all but the largest system size we investigate.

Observations. For the classical version of the problem, an observation consists of the explicit spins of an ensemble of replicas. In the case of an unknown Hamiltonian, the ensemble measurement is important as the instantaneous state of a single replica does not provide sufficient information about the current temperature of the system. Providing the agent with multiple replicas allows it to compute statistics and have the possibility of inferring the temperature.

For example, if there is considerable variation among replicas, then compute statistics and have the possibility of inferring the temperature. For instance, if there is considerable variation among replicas, then the system is probably hot, whereas if most replicas look the same, the system is probably cool.

When discussing a quantum system, where the spins represent qubits, direct mid-annal measurement of the system is not possible.
Fig. 2 | A neural network is used to learn the control parameters for several SA experiments. By observing a lattice of spins, the neural network can learn to control the temperature of the system in a dynamic fashion, annealing the system to the ground state. The spins at time \( t \) form the state \( s_t \) fed into the network. Two concurrent convolutional layers extract features from the state. These features are combined with a dense layer and fed into a recurrent module (an LSTM module) capable of capturing temporal characteristics. The LSTM module output is reduced to two parameters used to form the policy distribution \( \pi(\theta|s_t) \) as well as to approximate the value function \( V(s_t) \) used for the generalized advantage estimate.

As measurement causes a collapse of the wavefunction. To address this, we discuss experiments conducted in a ‘destructive observation’ environment, where measurement of the spins is treated as a ‘one-time’ opportunity for inclusion in RL training data. The subsequent observation is then based on a different set of replicas that have evolved through the same schedule, but from different initializations.

When running the classic SA baselines, to keep comparison fair, each episode consists of \( N_{\text{rep}} \) replicas as in the RL case. If even one replica reaches the ground state, the episode is considered a success.

Reinforcement learning algorithm. Through the framework of RL, we wish to produce a policy function \( \pi(\theta|s) \) that takes the observed binary spin state \( s_t \in \{-1, 1\}^{N_{\text{rep}} \times \text{ size} } \) and produces an action \( a_t \) corresponding to the optimal change in the inverse temperature.

We now briefly introduce PPO \(^{49}\). First we define our policy \( \pi(\theta|s) \) as the likelihood that the agent will take action \( a_t \) while in state \( s_t \); through training, the desire is that the best choice of action will become the most probable. To choose an action, this distribution can be sampled. We will use a neural network that is parameterized by weights \( \theta \) to represent the policy by assuming that \( \pi(\theta|s) \) is a normal distribution and interpreting the output nodes of the neural network as the mean, \( \mu \), and variance, \( \sigma^2 \).

We define a function \( Q_{\pi_{\theta}}(s_t, a_t) \) as the expected future discounted reward if the agent takes action \( a_t \) at time \( t \) and then follows policy \( \pi_{\theta} \) for the remainder of the episode. We additionally define a value function \( V_{\pi_{\theta}}(s_t) \) as the expected future discounted reward starting from state \( s_t \) and following the current policy \( \pi_{\theta} \) until the end of the episode. We introduce the concept of advantage, \( A_t(s_t, a_t) \), as the difference between these two quantities. \( Q_{\pi_{\theta}} \) and \( V_{\pi_{\theta}} \) are not known and must be approximated. We assume the features necessary to represent \( \pi \) are generally similar to the features necessary to estimate the value function, and thus we can use the same neural network to predict the value function by merely having it output a third quantity.

\( A_t \) is effectively an estimate of how much better the agent did in choosing action \( a_t \), compared to what was expected. We construct the typical policy gradient cost function by coupling the advantage of a state–action pair with the probability of the action being taken:

\[
L_{\text{PG}}(\theta) = \hat{E}_t \left[ \log \pi_{\theta}(a_t|s_t) A_t \right]
\]

which we want to maximize by modifying the weights \( \theta \) through the training process. It is, however, more efficient to maximize the improvement ratio \( r_t \) of the current policy over a policy from a previous iteration \( \pi_{\theta_{\text{old}}} \) (refs. \(^{52,53}\)):

\[
L_{\text{TRPO}}(\theta) = \hat{E}_t \left[ \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t \right] \equiv \hat{E}_t \left[ r_t(\theta) A_t \right]
\]

Note, however, that maximizing this quantity can be trivially achieved by making the new policy drastically different from the old policy, which is not the desired behaviour. The PPO algorithm\(^{49}\) deals with this by clipping the improvement and taking the minimum

\[
L_{\text{CLIP}}(\theta) = \hat{E}_t \left[ \min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t) \right]
\]

To train the value function estimator, a squared error is used:

\[
L_{\text{VF}}(\theta) = \hat{E}_t \left[ (V_{\pi_{\theta}}(s_t) - V_{\pi_{\theta_{\text{old}}}}(s_t))^2 \right]
\]

and to encourage exploration, an entropic regularization functional \( S \) is used. This all amounts to a three-term cost function

\[
L_{\text{PPO}}(\theta) = \hat{E}_t \left[ L_{\text{CLIP}}(\theta) - c_1 L_{\text{VF}}(\theta) + c_2 S(\pi_{\theta}(s_t)) \right]
\]

where \( c_1 \) and \( c_2 \) are hyperparameters.

Policy network architecture. The neural network is composed of two parts: a convolutional feature extractor and a recurrent network to capture the temporal characteristics of the problem (Fig. 2). The feature extractor comprises two parallel two-dimensional (2D) convolutional layers. The first convolutional layer has \( N_k \) kernels of size \( 1 \times L \), and aggregates along the replicas dimension, enabling the collection of spin-wise statistics across the replicas. The second convolutional layer has \( N_k \) kernels of size \( N_{\text{rep}} \times 1 \) and slides along the spin dimension, enabling the aggregation of replica-wise statistics across the spins. The outputs of these layers are flattened, concatenated and fed into a dense layer of size \( N_{\theta} \) hidden nodes. This operates as a latent space encoding for input to a recurrent neural network (a long short-term memory, or LSTM, module\(^{49}\)), used to capture the sequential nature of our application. The latent output of the LSTM module is of size \( N_{\theta} \). For simplicity, we set \( N_k = N_k = N_d = N_L = 64 \). All activation functions are hyperbolic tangent (tanh) activations. Because \( \phi \) can assume a continuum of real values, this task is referred to as having a continuous action space, and thus standard practice is for the network to output two values corresponding to the first and second moments of a normal distribution, which can be sampled to produce predictions.

Reward. At the core of RL is the concept of reward engineering, that is, developing a reward scheme to inject a notion of success into the system. As we only care about reaching the ground state by the end of a given episode, we use a sparse reward scheme, with a reward of zero for every time step before the terminal step, and a reward equal to the negative of the minimum energy as the reward for the terminal step:

\[
R_t = \begin{cases} 
0, & t < N_{\text{steps}} \\
- \min_k H(\phi_k(s_t)), & t = N_{\text{steps}} 
\end{cases}
\]

where \( k \in [1, N_{\text{rep}}] \) and

\[
\phi_k(s_t) \in \{-1, 1\}^{1 \times L},
\]

is an indexing function that returns the binary spin values for the kth replica of state \( s_t \). This reward function is agnostic to system size; as the system size increases, the correlation time will also
We present an analysis of two classes of Hamiltonians. The first, a hyperparameters. When optimizing the neural network, we use a hyperparameter search to find the best value for each parameter. The second class is a set of spin glasses. For each class, we generate a set of instances and use them to train our model. We compare the performance of our model against a baseline algorithm, which is a standard method for solving these types of problems. The results show that our model outperforms the baseline on all instances. Furthermore, we investigate how the hyperparameters influence the performance of the model. We find that there is an optimal set of hyperparameters that maximizes the performance of the model. Finally, we demonstrate the use of our model on a real-world problem, and show that it is able to generalize to new instances.
In all cases, the RL schedule obtains a better (lower) $n_{ea}$ value, meaning far fewer episodes are required for us to be confident that the ground state has been observed. Furthermore, the $n_{ea}$ value exhibits much better scaling with respect to the system size (that is, the number of optimization variables). In Fig. 4e–k, we plot some of the schedules that the RL algorithm produces. In many cases, we see initial heating, followed by cooling; although, in the case of the larger models (Fig. 4i–k) we see much more complex, but still successful, behaviour. In all cases, the variance of the policies with respect to time (shown as the shaded regions in Fig. 4e–k) indicates that the agent is using information from the provided state to make decisions, and not just basing its decisions on the elapsed time using the internal state of the LSTM module. If schedules were based purely on some internal representation of time, there would be no variance between episodes.

Comparing easy and difficult instances. The learned strategy of the RL agent is relatively simple in concept: increase the temperature to a sufficiently high value and then use the remaining time to cool the system as seen in the average policies in Fig. 4e–k. In this section, we demonstrate the degree to which the performance improvement can be attributed to the ability of the RL agent to base its decisions on the various dynamics in the system.

We divide the instances in the $10 \times 10$ test set into two subsets, which we label ‘easy’ and ‘difficult’ based on the success of the classic SA baseline. This results in 14 difficult instances in which classic SA succeeds in only 3% of anneals, and 86 easy instances in which classic SA succeeds in more than 3% of anneals.

We compare three temperature scheduling methods on 100 episodes of each instance in both subsets: (1) classic (linear) SA; (2) the RL agent; (3) an RL agent (not yet discussed) that does not include a recurrent LSTM module. As shown in Fig. 5a, linearly scheduled classic SA solves the easy instances in 19% of anneals, whereas the RL agent manages to solve the same instances with a 53% success probability. With the difficult instances, the difference is more extreme; classic SA manages only 1% success, whereas RL performs substantially better with 29% success.

A variant of the agent without an LSTM module performs more poorly, but still better than classic SA. This agent is simply provided with a floating point representation of the episode step concatenated to the state vector, but without a recurrent network, it has no mechanism to capture the time dependence (history) of the problem. It therefore can only use the current observation in making decisions, and evidently does so more poorly than the agent with access to an LSTM module. For our formulation of the environment, an LSTM module is theoretically important to achieve a well-defined Markov decision process.

In Fig. 5b we plot the average action taken and in Fig. 5c we plot the average inverse temperature of the system at each step in the test episodes driven by the RL agent, averaged over the easy and difficult instances separately. There is no notable difference in the average schedules of the two subsets. This fact, combined with the considerable magnitude of the standard deviation (plotted as a shaded region for difficult instances and vertical bars for easy ones) suggests that the RL agent is adaptive to the specific instantiation of each Hamiltonian. Some of these dynamics can be seen in the successful schedules randomly selected for plotting in Fig. 5f.

We then take the average schedules plotted in Fig. 5b,c and use them as if they were RL-designed general heuristic schedules, removing the necessity to conduct observations during the evaluation procedure. Both the difficult and easy average schedules perform very poorly on both the difficult and easy subsets, succeeding in fewer than 10% of episodes. This is strong evidence of the specificity of the RL agent’s actions to the particular dynamics of each
episode and refutes the hypothesis that a single, average policy, even if trained by RL, is a good case for generic instances.

We repeated the previous analysis with subsets based on the performance of the RL agent, arriving at identical conclusions (Fig. 5e).

**Destructive observation.** A key element of the nature of quantum systems is the collapse of the wavefunction when a measurement of the quantum state is made. When dealing with quantum systems, one must make control decisions based on quantum states that have evolved through an identical policy but have never before been measured. We model this restriction on quantum measurements by allowing any replica observed in the anneal to be consumed as training data for the RL algorithm only once. We simulate this behaviour by keeping track of the policy decisions (the changes in inverse temperature) in an action buffer as we play through each episode. When a set of $N_{\text{steps}}$ replicas are measured, they are consumed and the system is reset to a new set of initial conditions, as if it was a new episode. The actions held in the buffer are replayed on the new replicas.

In this situation, the agent cannot base its decision on any replica-specific temporal correlations between given measurements; this should not be a problem early in each episode, as the correlation timescale of a hot system is very short, and the system, even under non-destructive observation, would have evolved sufficiently in the time window between steps to be uncorrelated. However, as the system cools, the correlation timescale increases exponentially, and destructive observation prevents the agent from relying on temporal correlations of any given replica.

We evaluate an agent trained in this ‘quantum-inspired’ way and plot its performance alongside the non-destructive (that is, classical) case in Fig. 4d. In the case of destructive observation, the agent performs marginally less well than the non-destructive case, but still performs better than SA in all cases. As it is a more complicated task to make observations when the system is temporally uncorrelated, it is understandable that the performance would be inferior to the non-destructive case. Nonetheless, RL is capable of outperforming SA in both the destructive and non-destructive cases.

![Diagram](attachment:diagram.png)

**Fig. 4** An RL policy learns to anneal spin glass models. a, An example ($L = 4^2$) lattice. b, c, Plots of the acceptance ratios over time for the $L = 8^2$ (b) and $L = 12^2$ (c) lattices. d, Comparison of the scaling of the RL policy with respect to system size and comparison to classic SA. We plot the $n_{\text{steps}}$ value (the number of anneals required to be 99% certain of observing the ground state; in the case of 100% success, $n_{\text{steps}}$ is undefined and plotted as zero) as a function of system size for both the RL and the best linear simulated annealing schedule we observed. The 95% confidence interval is shown as a shaded region. For all system sizes investigated, the learned RL policy is able to reach the ground state in significantly fewer runs. The destructive observation results are also plotted; these also outperform the linear schedules. We note that the destructive observation requires far more Monte Carlo steps per episode to simulate the destructive measurements; this plot should not be interpreted as a comparison of run time with regard to the destructive observation result. e–k, Example inverse temperature schedules (solid lines) and average inverse temperature schedules (for all testing episodes) (dashed lines) for lattice sizes of $L = 4^2$ (e), $6^2$ (f), $8^2$ (g), $10^2$ (h), $12^2$ (i), $14^2$ (j) and $16^2$ (k). The shaded regions denote the standard deviation. In this work, we use $N_{\text{steps}} = 40$ episode steps.

The relative performance in terms of computational demand between destructive observation and SA alludes to an important future direction in the field of RL, especially when applied to physical systems where observation is destructive, costly and altogether difficult. With destructive observations, $N_{\text{steps}}$ systems must be initialized and then evolved together under the same policy. Each copy is consumed one by one, as observations are required for decision making, thus incurring an unavoidable $N_{\text{steps}}^2/2$ penalty in the destructive case. In this sense, it is difficult to consider RL to be superior; prescheduled SA simply does not require observation. However, if the choice to observe were to be incorporated into the action set of the RL algorithm, the agent would choose when observation would be necessary.

For example, in the systems presented in this work, the correlation time of the observations is initially small; the temperatures are high and frequent observations are required to guide the system through phase space. As the system cools, however, the correlation time grows exponentially, and the observations become much more similar to each previous observation; in this case, it would be beneficial to forgo some expensive observations, as the system would not be evolving substantially. With such a scheme, RL stands a better chance at achieving greater performance.

**Policy analysis.** To glean some understanding into what the RL agent is learning, we train an additional model on a well-understood Hamiltonian, the ferromagnetic Ising model of size $16 \times 16$. In this case, the temperatures are initialized randomly (as in the WSC model). This model is the extreme case of a spin glass, with all $J_{ij} = 1$. In Fig. 6a, we display the density of states $g(M, E)$ of the Ising model, plotted in phase space, with axes of magnetization per spin ($M/L$) and energy per spin ($E/L$). The density of states is greatest in the high-entropy $M = E = 0$ region and lowest in the low-entropy ‘corners’. We show the spin configurations at the three corners (chequerboard, all spin-up and all spin-down) for clarity. The density of states is obtained numerically using Wang–Landau sampling\cite{Wang1996}. Magnetization and energy combinations outside of the dashed ‘triangle’ are impossible.
**Fig. 5** | We separate the $10 \times 10$ spin glass instances in the test set into two subsets (easy and difficult), depending on the success of classic SA in finding their ground states. **a.** Plot of the performances of three different temperature scheduling approaches on the easy and difficult subsets. RL exhibits superior performance over classic SA in both subsets; however, it demonstrates dramatic superiority in the case of the difficult instances. RL without an LSTM module still performs better than classic SA; it can still dynamically modify the schedule and is not constrained to a constant temperature change at each step, so is more akin to a traditional heuristic temperature scheduling approach. **b, c.** Plots of the average RL actions (b) and schedule (c), respectively, for both the difficult and easy instance subsets. The standard deviations of the policies are plotted with error bars (easy instances) and shaded regions (difficult instances). The average difficult policy is very similar to the average easy policy, both having a large standard deviation, suggesting a high degree of specificity of the policy to each individual episode (see f). **d.** The performance when we apply the average actions in b as a static policy. The average policies perform even more poorly than classic SA. This is further evidence that the RL agent’s ability to observe the system is crucial to its high performance. One might object to the method used to split the instances into the difficult and easy subsets; we have explicitly chosen to split the subsets at a boundary that makes classic SA perform poorly on the difficult instances. **e.** We thus consider a difficult (easy) instance as one that the RL agent performs poorly (well) on, and the story remains unchanged. **f.** Plots of several successful schedules; each schedule is quite different from the others, but each results in a successful episode.

**Fig. 6** | We train an agent on a special case of the spin glass Hamiltonians: the $16 \times 16$ ferromagnetic Ising model where all couplings $J_{i} = 1$. **a.** We plot the density of states $\log(g(M, E))$ for the $16 \times 16$ Ising model in the phase space of energy and magnetization, sampled numerically using the Wang-Landau algorithm, and indicate the location of four of the novel high- and low-energy spin configurations: two ‘chequerboard’ configurations, all spin-up and all spin-down. **b.** For the trained model, we plot the average of the learned value function $V(s_i)$ for each possible energy-magnetization pair. Additionally, we plot the trajectories of the first replica for three episodes of annealing to demonstrate the path through phase space the algorithm learns to take. **c, d.** Enlarged views of two high-value regions of interest. **e.** Plot of the average action taken at each point in phase space, as well as the same trajectories plotted in b.
In Fig. 6b, we plot a histogram of the average value function \( V(s) \) on the phase plane, as well as three trajectories. Note that because each observation \( s_i \) is composed of \( N_{obs} \) replicas, we count each observation as \( N_{obs} \) separate points on the phase plot when computing the histogram, each with an identical contribution of \( V(s_i) \) to the average. As expected, the learned value function trends higher toward the two global energy minima. The lowest values are present in the initialization region (the high-energy band along the top). We expand two regions of interest in Fig. 6c,d. In Fig. 6d, we can see that the global minimum is assigned the highest value; this is justifiable in that if the agent reaches this point, it is likely to remain here and reap a high reward so long as the agent keeps the temperature low for the remainder of the episode.

In Fig. 6c, we identify four noteworthy energy–magnetization combinations, using asterisks. These four energy–magnetization combinations have identical energies, with increasing magnetization for the remainder of the episode. We analyse the value function that the agent learns and see that it attributes an intuitive representation of value to specific regions in action space; that is, the agent can traverse from the small-band configuration to the ground state in fewer spin flips than if traversing from the wide-band configurations.

In Fig. 6c, we plot a histogram of the average action taken at each point in phase space. The upper high-energy band exhibits more randomness in the actions chosen, as this is the region in which the system lands upon initialization. When initialized, the temperature is at a randomly drawn value, and sometimes the agent must first heat the system to escape a local minimum before then cooling, and thus the first action is, on average, of very low magnitude. As the agent progresses toward the minimum, the agent becomes more aggressive in cooling the system, thereby thermally trapping itself in lower energy states.

Scaling and time to solution. Figure 4d indicates that both non-destructive and destructive RL perform substantially better, not only in absolute terms, but also in terms of scaling. It is important to note that we have specifically chosen a neural network architecture (convolutional) that scales linearly with system size, and have trained each model for the same number of episodes, each consisting of the same number of sweeps. The computation time for each sweep scales linearly with the system size, and thus the training time of our RL models scales linearly with system size. Using RL does indeed impose an additional inference cost, as the observation must be processed by the neural network; on the \( L = 10^2 \) system, inference takes one-third the amount of time as does each episode step. However, this cost has not been optimized and could be lowered substantially through optimization of the neural network inference or even by offloading the policy network onto specialized hardware designed for inference.

Conclusion

In this work, we show that reinforcement learning is a viable method for learning dynamic control schemes for the task of SA. We show that, on a simple spin model, the RL agent is capable of devising a temperature control scheme that can consistently escape a local minimum and then anneal to the ground state. It arrives at a policy that generalizes to a range of initialization temperatures; in all cases, it learns to cool the system. However, if the initial temperature is too low, the RL agent learns to first increase the temperature to provide sufficient thermal energy to escape the local minimum. It achieves this without being provided explicit knowledge of the temperature.

We then demonstrate that the RL agent is capable of learning a policy that can generalize to an entire class of Hamiltonians and that the problem need not be restricted to a single set of couplings. By training multiple RL agents on increasing numbers of variables (increasing lattice sizes), we investigate the scaling of the RL algorithm and find that it outperforms a classic SA schedule both in absolute terms and in terms of its scaling.

Our technique is not limited to the system sizes we present in this work; larger system sizes are also within its reach. At some point, as the size of the system increases, correlation times in the underlying Metropolis–Hastings simulation become larger than the intervals between observations, and the number of sweeps must be increased. Additionally, we have specifically chosen a neural network architecture that scales linearly with system size (convolutional neural networks) as opposed to traditional multilayer perceptron networks that scale exponentially. In fact, the entire procedure scales at most polynomially with system size.

We analyse the value function that the agent learns and see that it attributes an intuitive representation of value to specific regions of phase space.

We discuss the nature of RL in the physical sciences, specifically in situations where observing systems is destructive (destructive observation) or costly (for example, performing quantum computations where observations collapse the wavefunction or conducting chemical analysis techniques that destroy a sample material). We demonstrate that our implementation of RL is capable of performing well in a destructive observation situation, albeit inefficiently. We propose that the future of physical RL (that is, RL in the physical sciences) will be one of ‘controlled observation’, where the algorithm can choose when an observation is necessary, minimizing the inherent costs incurred when observations are expensive, slow or difficult.

Data availability

The test datasets necessary to reproduce these findings are available at https://doi.org/10.5281/zenodo.3897413.

Code availability

The code necessary to reproduce these findings is available at https://doi.org/10.5281/zenodo.3897413.

Received: 4 March 2020; Accepted: 5 August 2020;
Published online: 7 September 2020

References

1. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. Science 220, 671–680 (1983).
2. Barahona, F. On the computational complexity of Ising spin glass models. J. Phys. A 15, 3241–3253 (1982).
3. Sherrington, D. & Kirkpatrick, S. Solvable model of a spin-glass. Phys. Rev. Lett. 35, 1792–1796 (1975).
4. Ising, E. Beitrag zur Theorie des Ferromagnetismus. Zeitschrift Phys. 31, 253–258 (1925).
5. Onsager, L. Crystal statistics. I. A two-dimensional model with an order-disorder transition. Phys. Rev. 65, 117–149 (1944).
6. Ferdinand, A. E. & Fisher, M. E. Bounded and inhomogeneous Ising models. I. Specific-heat anomaly of a finite lattice. Phys. Rev. 185, 832–846 (1969).
7. Lucas, A. Ising formulations of many NP problems. Front. Phys. 2, 1–14 (2014).
8. Hastings, B. Y. W. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57, 97–109 (1970).
9. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of state calculations by fast computing machines. J. Chem. Phys. 21, 1087–1092 (1953).
10. Kirkpatrick, S. Optimization by simulated annealing: quantitative studies. J. Stat. Phys. 34, 975–986 (1984).
11. van Laarhoven, P. J. M. & Aarts, E. H. L. Simulated Annealing: Theory and Applications (Springer, 1987).
12. Starler, J. & Silverman, B. W. Temperature schedules for simulated annealing. Stat. Comput. 4, 21–32 (1994).
13. Heim, B., Rønnow, T. F., Isakov, S. V. & Troyer, M. Quantum versus classical annealing of Ising spin glasses. Science 348, 215–217 (2015).
14. Bounds, D. G. New optimization methods from physics and biology. *Nature* **329**, 215–219 (1987).
15. Farhi, E. et al. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science* **292**, 472–475 (2001).
16. Hen, I. & Young, A. P. Solving the graph-isomorphism problem with a quantum annealer. *Phys. Rev. A* **86**, 042310 (2012).
17. Boixo, S. et al. Evidence for quantum annealing with more than one hundred qubits. *Nat. Phys.* **10**, 218–224 (2014).
18. Bian, Z. et al. Discrete optimization using quantum annealing on sparse Ising models. *Front. Phys.* **2**, 1–10 (2014).
19. Venturelli, D., Marchand, D. J. J. & Rojo, G. Quantum annealing implementation of job-shop scheduling. Preprint at https://arxiv.org/pdf/1906.08479.pdf (2015).
20. Ray, P., Chakrabarti, B. K. & Chakrabarti, A. Sherrington–Kirkpatrick model in a transverse field: absence of replica symmetry breaking due to quantum fluctuations. *Phys. Rev. B* **39**, 11828–11832 (1989).
21. Martoňák, R., Santoro, G. E. & Tosatti, E. Quantum annealing by the path-integral Monte Carlo method: the two-dimensional random Ising model. *Phys. Rev. B* **66**, 094203 (2002).
22. Santoro, G. E. Theory of quantum annealing of an Ising spin glass. *Science* **295**, 2427–2430 (2002).
23. Finnila, A., Gomez, M., Sebenik, C., Stenson, C. & Doll, J. Quantum annealing: a new method for minimizing multidimensional functions. *Chem. Phys. Lett.* **219**, 343–348 (1994).
24. Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse Ising model. *Phys. Rev. E* **58**, 5355–5363 (1998).
25. Harris, R. et al. Experimental demonstration of a robust and scalable flux qubit. *Phys. Rev. B* **81**, 134510 (2010).
26. Harris, R. et al. Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor. *Phys. Rev. B* **82**, 024511 (2010).
27. Johnson, M. W. et al. Quantum annealing with manufactured spins. *Nature* **473**, 194–198 (2011).
28. McGeoch, C. C. & Wang, C. Experimental evaluation of an adiabatic quantum system for combinatorial optimization. In *Proceedings of the ACM International Conference on Computing Frontiers*, CF ’13, Vol. 1, 1–11 (ACM, 2013).
29. Ikeda, K., Nakamura, Y. & Hume, T. S. Application of quantum annealing to nurse scheduling problem. *Sci. Rep.* **9**, 12837 (2019).
30. Dickson, N. G. et al. Thermally assisted quantum annealing of a 16-qubit problem. *Nat. Commun.* **4**, 1903 (2013).
31. Okada, S., Ohzeki, M. & Tanaka, K. Efficient quantum and simulated annealing of Potts models using a half-hot constraint. *J. Phys. Soc. Jpn* **89**, 094081 (2020).
32. Battaglia, D. A., Santoro, G. E. & Tosatti, E. Optimization by quantum annealing: lessons from hard satisfiability problems. *Phys. Rev. E* **71**, 066707 (2005).
33. Tsukamoto, S., Takatsuki, M., Matsubara, S. & Tamura, H. An accelerator architecture for combinatorial optimization problems. *Fujiitsu Sci. Technical J.* **53**, 8–13 (2017).
34. Inagaki, T. et al. A coherent Ising machine for 2,000-node optimization problems. *Science* **354**, 603–606 (2016).
35. Lebleu, T., Yamamoto, Y., McMahon, P. I. & Aihara, K. Destabilization of local minima in analog spin systems by correction of amplitude heterogeneity. *Phys. Rev. Lett.* **122**, 040607 (2019).
36. Tsunov, E. S., Ulanov, A. E. & Lvovsyuk, A. I. Annealing by simulating the coherent Ising machine. *Opt. Express* **27**, 10288–10295 (2019).
37. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. Preprint at https://arxiv.org/pdf/1411.4028.pdf (2014).
38. Farhi, E., Goldstone, J., Gutmann, S. & Zhou, L. The quantum approximate optimization algorithm and the Sherrington–Kirkpatrick model at infinite size. Preprint at https://arxiv.org/pdf/1910.08187.pdf (2019).
39. Sutton, R. & Barto, A. *Reinforcement Learning: An Introduction* 2nd edn (MIT Press, 2018).
40. Berner, C. et al. Dota 2 with large scale deep reinforcement learning. Preprint at https://arxiv.org/pdf/1912.06680.pdf (2019).
41. Zhang, Z. et al. Hierarchical reinforcement learning for multi-agent MOBA Game. Preprint at https://arxiv.org/pdf/1901.08004.pdf (2019).
42. Vinyals, O. et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **575**, 350–354 (2019).
43. Mnih, V. et al. Playing Atari with deep reinforcement learning. Preprint at https://arxiv.org/pdf/1312.5602.pdf (2013).
44. Silver, D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
45. Silver, D. et al. Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
46. Silver, D. et al. A general reinforcement learning algorithm that masters chess, shogi and Go through self-play. *Science* **362**, 1140–1144 (2018).
47. Agostinelli, F., McAlleer, S., Shmakov, A. & Baldi, P. Solving the Rubik’s cube with deep reinforcement learning and search. *Nat. Mach. Intell.* **1**, 356–363 (2019).
48. Akkaya, I. et al. Solving Rubik’s cube with a robot hand. Preprint at https://arxiv.org/pdf/1910.07113.pdf (2019).
49. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. Preprint at https://arxiv.org/pdf/1707.06347.pdf (2017).
50. Hill, A. et al. Stable Baselines (2018); https://github.com/hill-a/stable-baselines.
51. Brockman, G. et al. OpenAI Gym. Preprint at https://arxiv.org/pdf/1606.01540.pdf (2016).
52. Schulman, J., Levine, S., Abbeel, P., Jordan, M. & Moritz, P. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning* 1889–1897 (ICML, 2015).
53. Kakade, S. & Langford, J. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning* 267–274 (ICML, 2002).
54. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
55. Aramon, M. et al. Physics-inspired optimization for quadratic unconstrained problems using a Digital Annealer. *Front. Phys.* **7** (2019); https://doi.org/10.3389/fphy.2019.00048
56. Bunyk, P. I. et al. Architectural considerations in the design of a superconducting quantum annealing processor. *IEEE Trans. Appl. Superconductivity* **24**, 1–10 (2014).
57. Liers, F., Jünger, M., Reinelt, G. & Rinaldi, G. in *New Optimization Algorithms in Physics*, Vol. 4, 47–69 (Wiley, 2005).
58. Jünger, M. Spin glass server; https://informatik.uni-koeln.de/spinglass/.
59. Wang, F. & Landau, D. P. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.* **86**, 2050–2053 (2001).

**Acknowledgements**

I.T. acknowledges support from NSERC. K.M. acknowledges support from Mitacs. We thank B. Krayenhoff for valuable discussions in the early stages of the project and thank M. Bucyk for editing and managing the manuscript.

**Author contributions**

All authors contributed to the idea and design of the research. K.M. developed and ran the computational experiments and wrote the initial draft of the manuscript. P.R. and I.T. jointly supervised this work and revised the manuscript.

**Competing interests**

The authors declare no competing interests.

**Additional information**

Correspondence and requests for materials should be addressed to K.M., P.R. or I.T. 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.

© Crown 2020