

Searching for spin glass ground states through deep reinforcement learning

Received: 18 December 2022

Accepted: 25 January 2023

Published online: 09 February 2023

 Check for updatesChangjun Fan^{1,8}, Mutian Shen^{2,8}, Zohar Nussinov^{2,3,4}, Zhong Liu¹,
Yizhou Sun⁵  & Yang-Yu Liu^{6,7} 


Spin glasses are disordered magnets with random interactions that are, generally, in conflict with each other. Finding the ground states of spin glasses is not only essential for understanding the nature of disordered magnets and many other physical systems, but also useful to solve a broad array of hard combinatorial optimization problems across multiple disciplines. Despite decades-long efforts, an algorithm with both high accuracy and high efficiency is still lacking. Here we introduce DIRAC – a deep reinforcement learning framework, which can be trained purely on small-scale spin glass instances and then applied to arbitrarily large ones. DIRAC displays better scalability than other methods and can be leveraged to enhance any thermal annealing method. Extensive calculations on 2D, 3D and 4D Edwards-Anderson spin glass instances demonstrate the superior performance of DIRAC over existing methods. The presented framework will help us better understand the nature of the low-temperature spin-glass phase, which is a fundamental challenge in statistical physics. Moreover, the gauge transformation technique adopted in DIRAC builds a deep connection between physics and artificial intelligence. In particular, this opens up a promising avenue for reinforcement learning models to explore in the enormous configuration space, which would be extremely helpful to solve many other hard combinatorial optimization problems.

The Ising spin glass is a classical disordered system that has been studied for decades^{1,2}. Its spectacular behaviors have attracted considerable interests in several branches of science, including physics, mathematics, computer science, and biology. The endogenous nature of quenched disorder in spin glasses results in the fact that, it is hard to find out the ground state of such a system due to the frustrations (i.e., the impossibility of simultaneously minimizing all the interactions), despite its seemingly

simple Hamiltonian³:

$$\mathcal{H} = - \sum_{(i,j)} J_{ij} \sigma_i \sigma_j. \quad (1)$$

In general, this Hamiltonian can be defined on arbitrary graphs. Here, we will focus on the most heavily studied lattice realization of the nearest neighbor Ising spin glass, where the sites lie on a D -dimensional

¹College of Systems Engineering, National University of Defense Technology, 410073 Changsha, China. ²Department of Physics, Washington University in St. Louis, Campus Box 1105, 1 Brookings Drive, St. Louis, MO 63130, USA. ³Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Oxford OX1 3PU, UK. ⁴LPTMC, Sorbonne Université, Paris 75006, France. ⁵Department of Computer Science, University of California, Los Angeles, CA 90024, USA. ⁶Channing Division of Network Medicine, Department of Medicine, Brigham and Women's Hospital and Harvard Medical School, Boston, MA 02115, USA. ⁷Center for Artificial Intelligence and Modeling, The Carl R. Woese Institute for Genomic Biology, University of Illinois at Urbana-Champaign, Champaign, IL 61820, USA. ⁸These authors contributed equally: Changjun Fan, Mutian Shen  e-mail: yzsun@cs.ucla.edu; yyl@channing.harvard.edu

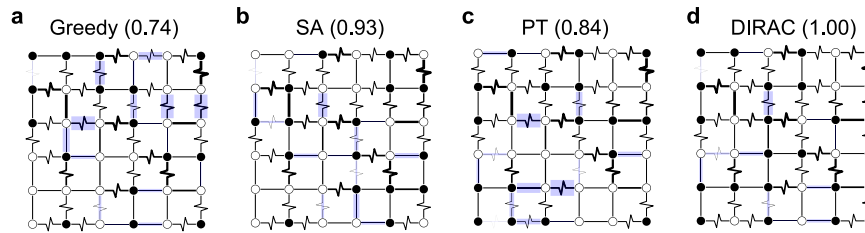


Fig. 1 | Case study comparison. We applied four algorithms: (a) Greedy, (b) Simulated annealing (SA), (c) Parallel tempering (PT), and (d) DIRAC (more precisely, DIRAC^{CR}) to compute the ground state of a randomly generated 6×6 Edwards-Anderson (EA) spin glass instance with fixed boundary conditions and couplings J_{ij} sampled from the Gaussian distribution $\mathcal{N}(0,1)$. The ferromagnetic bonds ($J_{ij} > 0$) are shown with straight lines, while the anti-ferromagnetic bonds ($J_{ij} < 0$) are shown with zigzag lines. The width of the lines are proportional to $|J_{ij}|$. Nodes are filled/hollow if the spin values $\sigma_i = +1/-1$, respectively. If the energy of a

bond, $-J_{ij}\sigma_i\sigma_j$, is positive, namely not satisfied, we draw a light blue shaded rectangle around the bond, with width proportional to $|J_{ij}|$. This way a smaller total shaded area of the image corresponds to a lower system energy. We also showed the approximate ratio (prediction/ground truth, where the numerator is the energy of the predicted ground state computed by each method and the denominator is the exact ground state energy computed by Gurobi, a branch-and-bound based exact solver) in brackets. Note that in this small case DIRAC has actually achieved the exact ground state.

hypercubic lattice with $N = L^D$ sites (see Fig. 1 for 2D instances) and $\sigma_i = \pm 1$ represents the binary Ising spin value at site i . The coupling J_{ij} is a Gaussian random variable that represents the interaction strength between two neighboring spins i and j . In the literature, this is often referred to as the Edwards-Anderson (EA) spin glass model. The EA model aims at capturing the quintessential character of real, physically occurring, spin glasses⁴. Comparing to other short range models such as the mean-field Bethe Lattice⁵, the EA model seems more challenging in the sense that there exists vast amounts of short loops that will lead to much more frustrations.

There are at least three strong motivations to find the ground states of spin glasses. First of all, finding the spin glass ground states is a key to the mysteries behind the strange and complex behaviors of spin glasses (and many other disordered systems), such as its glassy phase⁶ and ergodicity breaking⁷. In particular, ground-state energies in different boundary conditions can be used to compute the stiffness exponent of spin glasses, which can help us ensure the existence of a spin glass phase at finite temperatures^{8,9}. Second, finding ground states of Ising spin glasses in three or higher dimensions is a non-deterministic polynomial-time (NP) hard problem¹⁰, which is closely related to many other hard combinatorial optimization problems¹¹. For example, all of Karp's 21 NP-complete problems and many NP-hard problems (such as the max-cut problem, the traveling salesman problem, the protein folding problem, etc.) have Ising spin glass formulations¹¹⁻¹³. Therefore, finding the Ising spin glass ground states may help us solve many other NP-hard problems. Finally, the celebrated Hopfield model¹⁴ and other pioneering models of neural networks drew deep connections with Ising magnets¹⁵ (and spin glasses, in particular^{16,17}) on general networks. The study of spin glasses and their ground states has led to (and will continue lead to) the development of powerful optimization tools such as the cavity method and Belief Propagation that will further shed new light on computational complexity transitions^{2,18}.

Given the NP-hard nature of finding the spin glass ground states in three or higher dimensions, the exact branch-and-bound approach can only be used for very small systems¹⁹. For two-dimensional lattices with periodic boundary conditions in at most one direction (or planar graphs in general), the Ising spin glass ground states can be calculated by mapping to the minimum-weight perfect matching problem, which can be exactly solved in polynomial time^{20,21}. However, for general cases with large system sizes, we lack a method with both high accuracy and high efficiency. We used to rely on heuristic methods. In particular, Monte Carlo methods based on thermal annealing, e.g., simulated annealing (SA)²², population annealing²³ and parallel tempering (PT)²⁴⁻²⁷, have been well studied in the statistical physics community.

Recently, reinforcement learning (RL) has proven to be a promising tool in tackling many combinatorial optimization problems, such as the minimum vertex cover problem²⁸, the minimum independent set problem²⁹, the network dismantling problem³⁰, the travelling salesman problem³¹, the vehicle routing problem³², etc. Compared to traditional methods, RL-based algorithms are believed to achieve a more favorable trade-off between accuracy and efficiency. We note that RL was recently used to devise a smart temperature control scheme of simulated annealing in finding ground states of the 2D spin glass system, which enabled small systems to better escape local minimum and reach their ground states with high probability³³. However, this RL-enhanced simulated annealing still fails in finding ground states for larger spin glass systems in three or higher dimensions.

In this work, we introduce DIRAC (Deep reinforcement learning for spin-glass ground-state Calculation), a RL-based framework that can directly calculate spin glass ground states. DIRAC has several advantages. First, it demonstrates superior performances (in terms of accuracy) over the state-of-the-art thermal annealing methods, especially when the gauge transformation (GT) technique is adopted in DIRAC. Second, it displays better scalability than other methods. Finally, it can be leveraged to enhance any thermal annealing method and offer much better solutions.

Results

Reinforcement learning formulation

Following many other RL formulations in solving combinatorial optimization problems^{31,34-36}, DIRAC considers the spin glass ground state search as a Markov decision process (MDP), which involves an agent interacting with its environment (i.e., the input instance), and learning an optimal policy that sequentially takes the long-sighted action so as to accumulate its maximum rewards. To better describe this process, we first define state, action and reward in the context of Ising spin glass ground state calculation. State: a state s represents the observed spin glass instance, including both the spin configuration $\{\sigma_i\}$ and the coupling strengths $\{J_{ij}\}$, based on which the optimal action will be chosen. The terminal state s_T is met when the agent has tried to flip each spin once. Action: an action $a^{(i)}$ means to flip spin i . Reward: the reward $r(s, a^{(i)}, s')$ is defined as the energy change after flipping spin i from state s to get a new state s' , i.e., $r(s, a^{(i)}, s') = 2 \sum_{j \in \partial i} J_{ij} \sigma_i \sigma_j$, where ∂i represents the set of nearest neighbors of spin i .

Through the RL formulation, we seek to learn a policy $\pi_{\Theta}(a^{(i)}|s)$ that takes any observed state s and produces the action $a^{(i)}$ corresponding to the optimal spin flip that maximizes the expected future cumulative rewards. Here $\Theta = \{\Theta_{\mathcal{E}}, \Theta_{\mathcal{D}}\}$ represents a collection of learnable encoding parameters $\Theta_{\mathcal{E}}$ and decoding parameters $\Theta_{\mathcal{D}}$, which will be updated through RL.

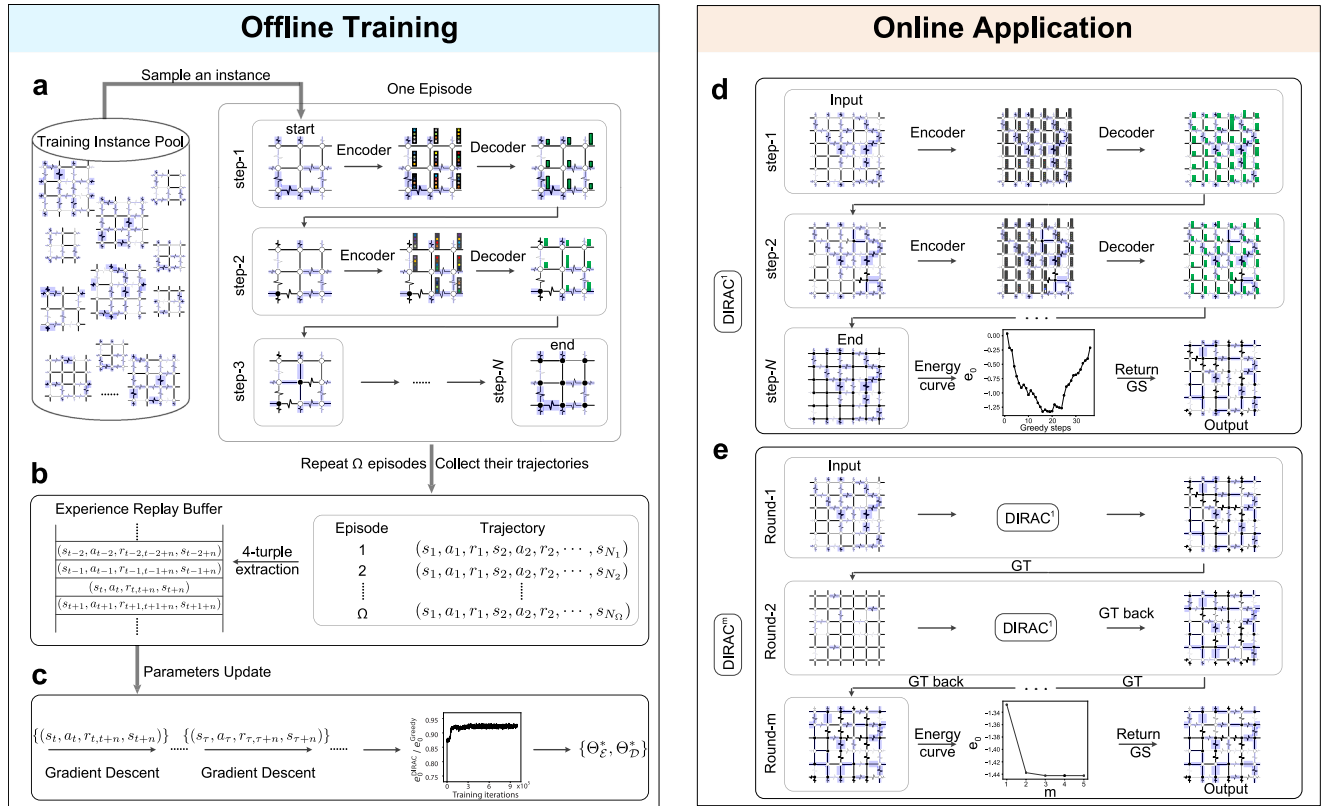


Fig. 2 | Overview of DIRAC. The DIRAC framework consists of two phases: offline training and online application. (*Left*) During training, we first generate random small EA spin glass instances with couplings sampled from Gaussian distribution, as the training data. **a** For each episode, we sample a random instance of size N , and let DIRAC learn to find its ground state. During each episode, the agent starts from the all-spins-up configuration and ends at the all-spins-down configuration, with each spin flipped only once. For the next episode, we sample another training instance. To determine the right action to take, DIRAC first adopts an encoder to represent each node as an embedding vector (shown as a color bar), and then decodes a Q -value (shown as a green bar with heights proportional to its value) for each node that predicts its long-term gain. **b** When one episode ends, we collect the trajectory $(s_1, a_1, r_1, \dots, s_N)$ generated during this process, extract the 4-tuple transitions, i.e., $(s_t, a_t, r_{t,t+n}, s_{t+n})$, where $r_{t,t+n} = \sum_{k=t}^{t+n} r_k$, and push them into the experience replay buffer \mathcal{B} , which is a queue that maintains S_{buffer} most recent n -step

transitions. **c**, To update parameters $\Theta = \{\Theta_E, \Theta_D\}$, we randomly sample mini-batch transitions from the buffer and perform gradient descents over Eq. (3). Repeat this process until the number of training episodes reaches $\Omega = 10^6$. The best model is selected with achieving the highest validation performance, which is measured by the approximation ratio $(e_0^{\text{DIRAC}}/e_0^{\text{Greedy}})$ on the validation data. Here e_0^{DIRAC} and e_0^{Greedy} are the energy densities computed by DIRAC and by Greedy respectively. (*Right*) During application, we have two basic DIRAC strategies: DIRAC¹ and DIRAC^m. **d**, For an input instance, DIRAC¹ (with the optimized parameters $\{\Theta_E^*, \Theta_D^*\}$) starts from $\{\sigma_i = +1\}$, and greedily flips the highest- Q spins till $\{\sigma_i = -1\}$. The spin configuration of the lowest energy encountered during this process is returned as the predicted ground state. **e**, DIRAC^m refers to a sequential running of m iterations of DIRAC¹ connected by GTs. For each iteration, GT converts the lowest-energy configuration from last iteration to be $\{\sigma_i = +1\}$ for DIRAC¹, and convert it back as the output of the current iteration.

DIRAC architecture

We design DIRAC to learn the policy π_Θ automatically. As shown in Fig. 2, DIRAC consists of two phases: offline training and online application. For offline training, the DIRAC agent is self-taught on randomly generated small-scale EA spin glass instances. For each instance, the agent interacts with its environment through a sequence of states, actions and rewards (Fig. 2a). Meanwhile, the agent gains experiences to update its parameters, which enhances its ability in finding the ground states of EA spin glasses (Fig. 2b,c). For online application, the well-trained DIRAC agent can be used either directly (DIRAC¹, Fig. 2d) or iteratively (DIRAC^m, Fig. 2e) or just as a plug-in to a thermal annealing method (DIRAC-SA and DIRAC-PT), on EA spin glass instances with much larger sizes than the training ones.

DIRAC’s success is mainly determined by the following two key issues: (1) How to represent states and actions effectively? (2) How to leverage these representations to compute a Q -value, which predicts the long-term gain for an action under a state. We refer to these two questions as the encoding and decoding problem, respectively.

Encoding. Since a hypercubic lattice can be regarded as a special graph, we design an encoder based on graph neural networks^{37–41},

namely SGNN (Spin Glass Neural Network), to represent states and actions. As shown in Fig. 3, to capture the coupling strengths $\{J_{ij}\}$, which are crucial to determine the spin glass ground states, SGNN performs two updates at each of the K iterations: the edge-centric update and the node-centric update, respectively. Here, the hyper-parameter K represents the number of message-passing steps in SGNN, and we set $K = 5$ in our calculations. The edge-centric update (Fig. 3b, Fig. S1a) aggregates edge embedding vectors, which are initialized as edge input features (SI Sec. IA), from its adjacent nodes. The node-centric update (Fig. 3c, Fig. S1b) aggregates node embedding vectors, which are initialized as node input features (SI Sec. IA), from its adjacent edges. Both updates concatenate the self embedding and the neighborhood embedding and are then subjected to a non-linear transformation (e.g., rectified linear unit, $\text{ReLU}(z) = \max(0, z)$). Traditional graph neural networks architectures often carry only node-centric updates^{37–39,41}, with edge weights taken as node’s neighborhood if needed. Yet this would fail in our case where edge weights play vital roles, and lead to unsatisfactory performances (see ablation study in SI Sec. IF and Fig. S2).

SGNN repeats K iterations of both edge-centric and node-centric updates, and finally obtains an embedding vector for each node (or

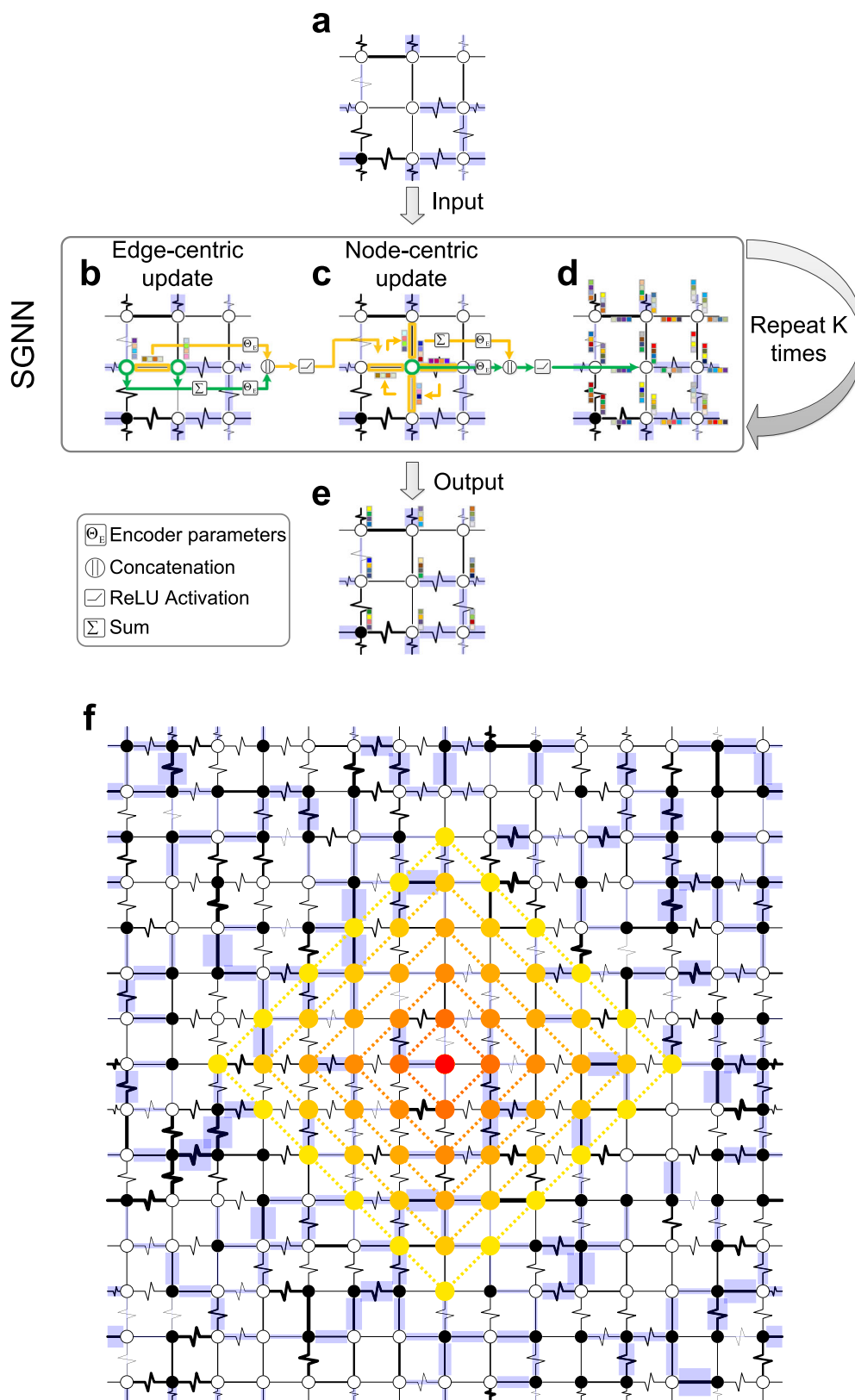


Fig. 3 | The encoder Spin Glass Neural Network (SGNN). SGNN encodes (a) the input spin glass instance (with any arbitrary spin configuration) into a (e) low-dimensional space, where each spin is associated with an embedding vector (shown as a color bar). **b** SGNN first updates edge embedding vectors based on the edge itself and its adjacent nodes, and then (c) updates node embedding vectors based on the node itself and its adjacent edges. Note that the node-centric updates take place only when the edge-centric updates finish for all edges at each layer. Both updates are followed by a non-linear transformation operator (e.g., ReLU) with

learnable parameters. The edge features are initialized by edge weights, and the node features are initialized by its coordinates in the hypercubic coordinate system. **d**, Each node or edge updates its embedding vector in one layer. Repeating several layers, we obtain an embedding vector for each node (e) that reflects its informative features. **f**, Each layer of updates increases the long-range couplings with one more hop's neighbors (dashed lines) for a given spin. For example, for the central spin (colored in dark red), its final embedding vector after $K = 5$ layers captures both its position and its long-range couplings with neighbors within $K = 5$ hops.

spin) (Fig. 3e). Essentially, each node’s embedding vector after K iterations captures both its position in the lattice and its long-range couplings with neighbors within K hops (see Fig. 3f for an example of $K = 5$). In our RL setting, each node is subject to a potential action, thus we also call the embedding vector of node i , denote as z_i , its action embedding. Collectively, we denote $z_a = \{z_i\}$, which includes embedding vectors for all the nodes $i = 1, \dots, N$. To represent the whole lattice (i.e., the state in our setting) and obtain the state embedding, denote as z_s , we sum over all node embedding vectors⁴¹, which is a straightforward and empirically effective way for graph-level encoding. (SI Sec. IA and Algo. S3 describe more details about SGNN.)

Decoding. Once the action embeddings z_a and state embedding z_s have been computed, DIRAC will leverage these representations to compute the state-action pair value function $Q(s, a^{(i)}; \Theta)$, which predicts the expected future cumulative rewards if taking action $a^{(i)}$ under state s , and following the policy $\pi_{\Theta}(a^{(i)}|s)$ till the end of the episode (i.e., till all the spins have been flipped once). Hereafter, we will refer to this function as the Q -value of spin- i . Specifically, we concatenate the embeddings of state and action, and apply a neural network with non-linear transformations to map the concatenation $[z_s, z_i]$ to a scalar value. In theory, any neural network architecture can be used. Here for the sake of simplicity, we adopt the classical multilayer perceptron (MLP) with ReLU activation. (see SI Sec. IB for more details):

$$Q(s, a^{(i)}; \Theta) = \text{MLP}([z_s, z_i]; \Theta). \tag{2}$$

Note that here $\Theta = \{\Theta_{\mathcal{E}}, \Theta_{\mathcal{D}}\}$, $\Theta_{\mathcal{E}}$ are the SGNN encoder parameters (see SI Eq. 1–Eq. 2), $\Theta_{\mathcal{D}}$ are the MLP decoder parameters (see SI Eq. 3).

Offline training. We will adopt the above Q function to calculate the spin glass ground state. Prior to that, we first need to optimize the Q function to predict a more accurate future gain.

We define the n -step Q -learning loss as:

$$\mathcal{L} = \mathbb{E}_{(s_t, a_t, r_{t:t+n}, s_{t+n}) \sim \mathcal{B}} \left[\left(r_{t:t+n} + \gamma \max_{a_{t+n}} Q(s_{t+n}, a_{t+n}; \hat{\Theta}) - Q(s_t, a_t; \Theta) \right)^2 \right], \tag{3}$$

and we perform mini-batch gradient descent to update parameters Θ over large amounts of experience transitions, which are represented by the 4-tuple transitions $(s_t, a_t, r_{t:t+n}, s_{t+n})$ in the DIRAC framework. The transitions are randomly sampled from the experience replay buffer \mathcal{B} , s_t and a_t denote the state and action at time step t , respectively. $r_{t:t+n} = \sum_{k=0}^{n-1} \gamma^k r(s_{t+k}, a_{t+k}, s_{t+k+1})$ represents the n -step accumulated reward, the discount factor γ is a hyper-parameter that controls how much to discount future rewards. $\hat{\Theta}$ is the target parameter set, which will only be updated with Θ every a certain number of episodes (see SI Sec. IC and Algo. S4 for more details on training).

Online application. DIRAC is trained over a large number of small random spin glass instances. Once the training phase is finished, we will perform the optimized Q -based ground state search. Traditional Q -based strategy greedily takes the highest- Q action each step till the end. Here we adopt the batch nodes selection strategy³⁰, i.e., at each step we flip a top fraction (e.g., 1%) of the spins with highest Q -values. Similar to the training phase, we start from the all-spins-up configuration, end at the all-spins-down configuration, and each spin is flipped only once. Hereafter we refer to this process as DIRAC¹. The spin configuration of the lowest energy encountered during this process is returned as the predicted ground state. Note that starting from the same configuration forces the agent to learn a strategy with the same starting point, which drastically reduces the potential trajectory space and thus requires less data for training. Ending at the same configuration makes the agent always finish the MDP within finite steps. This

finite-horizon MDP forces the agent to pick the right move without allowing too many regrets.

We emphasize that the vanilla strategy DIRAC¹ has several limitations. First, it can only handle one single uniform initialization $\{\uparrow, \dots, \uparrow\}$, rather than multiple random initializations. This drastically hinders DIRAC’s performance as significant performance improvements would be achieved by simply taking the best solution found across multiple initializations. Second, starting from the all-spins-up configuration and ending at the all-spins-down configuration (with each spin flipped only once) is certainly not ideal. An ideal way is to let the agent “revisit” its earlier decisions so as to search for an ever-improving solution. After all, due to the inherent complexity of combinatorial optimization problems, a policy that produces only one single “best-guess” solution is often sub-optimal. However, most of the existing RL algorithms are unable to revisit their earlier decisions in solving combinatorial optimization problems, because they often use a greedy strategy to construct the solution incrementally, adding one element at a time. To solve this issue, many recent attempts designed complex neural network architectures with lots of tedious and ad hoc input features^{42,43}. Yet, their presented results are far from satisfactory, they could not achieve the exact ground truth on even small instances, and they lack the validations on large instances.

To resolve the limitations of DIRAC¹, we employ the technique of GT in Ising systems⁴⁴. The GT between one spin glass instance $\{\sigma_i, J_{ij}\}$ and another instance $\{\sigma'_i, J'_{ij}\}$ are given by^{45,46}:

$$J'_{ij} = J_{ij} t_i t_j, \sigma'_i = \sigma_i t_i, \tag{4}$$

where $t_i = \pm 1$ are independent auxiliary variables so that σ'_i can take a desired Ising spin value. This technique is able to switch the spin glass system between any two configurations while keeping the system energy invariant (since $J_{ij} \sigma'_i \sigma'_j = J_{ij} \sigma_i \sigma_j$), which also means any input configuration can be gauge transformed to the all-spins-up configuration. In this way, DIRAC¹ is able to handle any random input spin configuration. Note that if there exists external fields h_i , we only need to make $h'_i = h_i t_i$ so that GT still works.

With the aid of GT, we can design a more powerful strategy beyond DIRAC¹, referred to as DIRAC^m hereafter. As the name suggests (also shown in Fig. 2e), DIRAC^m repeats m iterations of DIRAC¹. During each iteration, DIRAC¹ starts from an instance with all-spins-up configuration, which is obtained by gauge transforming the lowest-energy configuration found in the previous iteration, until the system energy no longer decreases. (As shown in SI Fig. S3, initializing from the lowest-energy configuration found in the previous iteration is much better than from a random one.) Notably, GT allows DIRAC^m to revisit the earlier decisions by computing a new set of Q -values (so as to re-evaluate the states and actions) at each iteration. The Q -value can be seen as a function of $\{J_{ij}, \sigma_i\}$, i.e., $Q(J_{ij}, \sigma_i)$. DIRAC will generate different Q -values for different instances, as long as they have different bond signs and spin values (even if those instances are connected by GTs and hence share the same physics). This also explains why GT only works for DIRAC, but fails for any other energy-based methods, such as Greedy, SA or PT. Those methods only consider the energy of each bond, while GT does not change the energy of each bond at all: $(-J_{ij} \sigma_i \sigma_j = -J'_{ij} \sigma'_i \sigma'_j)$. (see SI Sec. ID for more details on DIRAC^m).

Another prime use of GT is the so-called gauge randomization⁴⁷, where one may execute many runs (or randomizations) of DIRAC (either DIRAC¹ or DIRAC^m) for an input spin glass instance, with each run the instance is randomly initialized with a different spin configuration. The configuration of the lowest energy among these runs is then returned as the predicted ground state of the input instance.

DIRAC can also serve as a plug-in to Monte-Carlo based methods, such as SA and PT. The key ingredient of these methods is the so-called

Metropolis-Hastings criterion:

$$P(\Delta E; \beta) = \min(1, e^{-\beta \Delta E}), \quad (5)$$

which means that the probability of accepting a move with energy change ΔE at β (indicates the inverse temperature, $1/T$) is the minimum of 1 and $e^{-\beta \Delta E}$. The move is usually referred to as a small perturbation of the system, and in our case it just means a single-spin flip. At high temperatures, the Metropolis-Hastings criterion tends to accept all possible moves (including “bad” moves with $\Delta E > 0$). However, at low temperatures it is more likely to accept those “good” moves that could lower the energy (i.e., with $\Delta E < 0$), rendering the move-and-accept iteration more like a greedy search. We refer the process of using the Metropolis-Hastings criterion to accept moves as the energy-based Metropolis-Hastings (EMH) procedure hereafter. The art of these Monte-Carlo based methods, in some sense, is the balance between exploration at high temperatures and exploitation (energy descents) at low temperatures.

The general idea of using DIRAC as a plug-in to Monte-Carlo based methods is to replace the EMH procedure with DIRAC. (Later in this paper we will demonstrate the long-sighted greediness of DIRAC with respect to a purely energy-based greediness.) Specifically, we design a DIRAC-based Metropolis-Hastings (DMH) procedure. At each iteration, we let the systems or replicas choose randomly between DMH and EMH for a more effective configuration search. The DMH procedure uses one DIRAC¹ (with the assistance of GT) to lower the system energy. When the system energy reaches a local minimum (i.e., $\Delta E = 0$), DMH will perturb the spin configuration by flipping each spin with a temperature-dependent probability (SI Eq. 6). When applying this plug-in idea to SA and PT, we obtain DIRAC-SA and DIRAC-PT, respectively. See SI Sec. IH, Algo. S6, Algo. S7 and Fig. S4 for more details about these two DIRAC-enhanced algorithms.

Performance of finding the ground state

To demonstrate the power of DIRAC in finding the ground states of Ising spin glasses, we first calculated the probability P_0 of finding the ground states of small-scale EA spin glass instances as a function of the number of initial spin configurations (denoted as n_{initial}) (see Fig. 4(a–c)). Here we want to point out the difference between the concepts of initial configuration and run. Usually initial configuration can be seen as the same as sweep, referring to N spin-flip (attempts). A run refers to a complete process of an algorithm, and it contains a certain number of initial configurations. For example, PT consists of N_e epochs and N_r replicas, in each epoch, a replica will do a single sweep, namely N spin-flip attempts, so a run of PT contains $N_e \times N_r$ initial configurations; DIRAC^m contains m iterations of DIRAC¹, and each DIRAC¹ contains N spin-flips, so each run of DIRAC^m is counted as m initial configurations. Those instances were chosen to be small so that their exact ground states can be calculated by the branch-and-bound based solver Gurobi within tolerable computing time. For each given n_{initial} , we empirically computed P_0 as the fraction of 1000 random instances for which the ground state is found by DIRAC (and confirmed by Gurobi⁴⁵). We found that DIRAC enables a much faster finding of ground states than the Greedy, SA, and PT algorithm. In fact, all DIRAC variants (DIRAC¹, DIRAC^m, DIRAC-SA and DIRAC-PT) facilitate the finding of ground states. For example, in the case of $D=2$ and $L=10$ (Fig. 4a), n_{initial}^* (the minimum value of n_{initial} where P_0 reaches 1.0) of PT is 322,800, while n_{initial}^* of DIRAC^m is only 600. In fact, for DIRAC^m the ground states can be found with only one gauge randomization for some instances.

To systematically compare those algorithms in terms of their ability of finding the ground states, we investigated the system size scaling of n_{initial}^* . As shown in Fig. 4(d–f), DIRAC’s superior performances of facilitating the finding of ground states is persistent across

different systems with varying sizes, rather than only for the three sizes presented in Fig. 4(a–c).

We notice that the $P_0 - n_{\text{initial}}$ curve of DIRAC-SA contains very few scatter points, and its n_{initial}^* is almost independent of the system size N . This is because in our implementation, each result of SA or DIRAC-SA is calculated using 5000 initial configurations (one run), and for these small systems, one run of DIRAC-SA is able to reach their ground states. Due to the NP-hard nature of this problem, we believe the n_{initial}^* of DIRAC-SA will eventually grow exponentially with N for large N .

We also notice that in Fig. 4(b,c) the performances of SA and PT seem to be worse than the simple Greedy algorithm. We suspect this is because for those small systems, the simple Greedy algorithm, which greedily flips the highest-energy-drop spin, could reach the ground states much faster than SA or PT. Indeed, those annealing-based algorithms often require multiple energetically-unfavorable spin-flips in order to ultimately reach a lower energy state. For large systems, the Greedy algorithm would easily get stuck in the local minimum and thus need more initial configurations to reach the ground state, as shown in Fig. 4(a,d–f).

Performance of minimizing the energy

For larger systems, it is hard to compute the probability of finding the ground states for any algorithm, because we need to confirm if the calculated “ground state” is the true ground state obtained by an exact solver, and even the best branch-and-bound solver could not calculate the ground states of very large instances within acceptable time. In this case, a more practical choice of benchmarking various algorithms is to compare the energy of their predicted “ground state”, denoted as E_0 , which is not necessarily the true ground state energy, but the lowest energy provided by each algorithm for each particular instance. In particular, we are interested in the disorder averaged “ground-state” energy per spin, denoted as e_0 , which is computed as E_0/N averaged over many instances. In Fig. 5, we demonstrate e_0 as a function of n_{initial} on several large systems. Up to our knowledge, some of these systems, such as $D=3, L=20$, have never been considered in previous studies. Moreover, we have never seen results on the 4D systems in the literature.

From Fig. 5 we made several following observations. First, DIRAC-SA reaches the lowest e_0 for all cases. In some cases, DIRAC-SA is very close to the reported ground state in existing studies. For example, for $D=3$ and $L=10$, Ref. ⁴⁹ reported $e_0 = -1.6981$ (with $n_{\text{initial}} = 3.2 \times 10^7$, obtained by PT), while DIRAC-SA obtained $e_0 = -1.6906$ (with $n_{\text{initial}} = 2 \times 10^4$, fewer than one thousandth of the number of initial configurations used in Ref. ⁴⁹) (Fig. 5d). Second, the performance of SA in minimizing the system energy is surprisingly good, which is comparable, sometimes even has a better performance than PT (up to $n_{\text{initial}} = 2 \times 10^4$). PT has long been considered as the state-of-the-art algorithm for the spin glass ground state problem^{27,49,50}. However, our observation suggests that we should revisit the potential of SA in solving this problem. Third, DIRAC as a general plug-in could greatly improve annealing-based Monte-Carlo methods, such as SA and PT. For the nine systems studied in Fig. 5, DIRAC-SA computes an average 0.79% energy lower than SA, and DIRAC-PT calculates an average 2.01% energy lower than PT. Statistical tests indicate that these improvements are not marginal, but statistically significant: p value $< 10^{-4}$ for most cases, and < 0.05 for all the cases (Wilcoxon signed-ranked test, see SI, Fig. S5). Finally, there is a clear performance gap between DIRAC¹ and DIRAC^m in Fig. 5. This is simply because DIRAC^m (as a sequential running of m iterations of DIRAC¹ connected by GTs) can jump out of local minimum and finally reaches a much lower energy state than DIRAC¹.

Efficiency

Besides the effectiveness, DIRAC is also computationally efficient. For example, during the application phase, at each step DIRAC¹

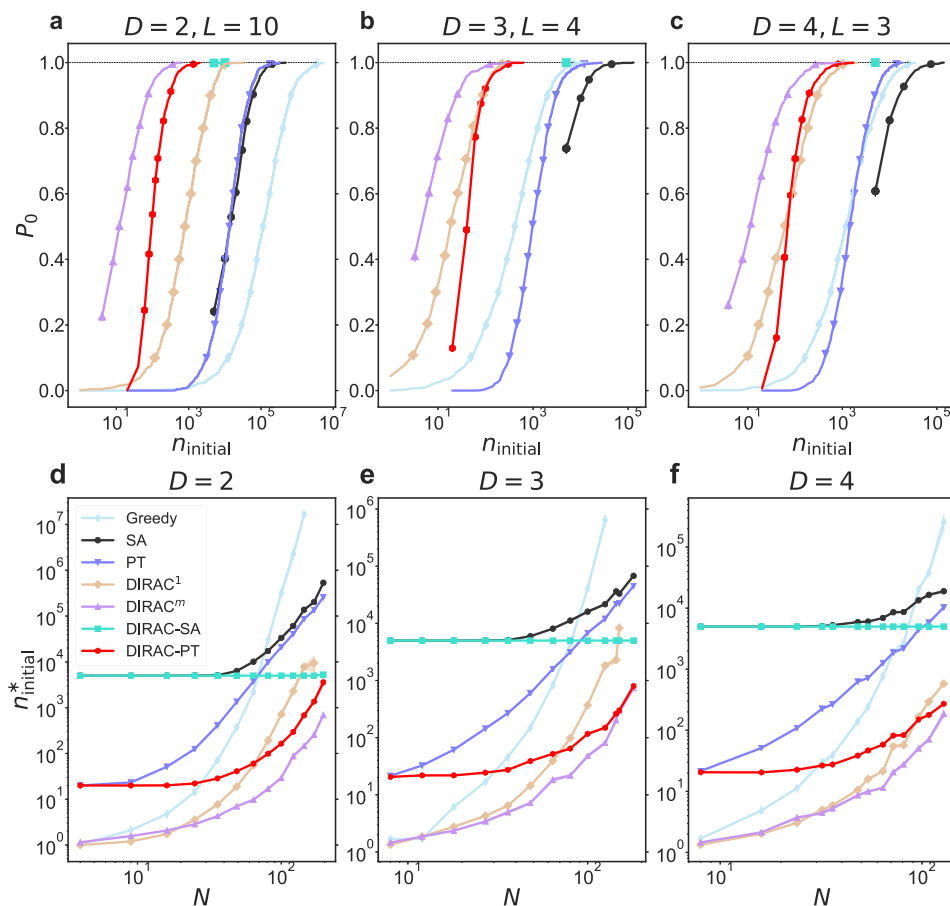


Fig. 4 | Performance of different methods in finding the ground state. To compare the ability of finding the exact ground states, we evaluated different methods on small spin glass systems for which the true ground state can be computed by the branch-and-bound-based solver Gurobi⁴⁹. In (a–c), the quantity we chose to compare is the probability of finding the ground state, denoted as P_0 , which is empirically calculated as the fraction of 1000 random instances for which the ground state is found (and confirmed by Gurobi). We computed P_0 as a function of the number of different initial configurations, denoted as n_{initial} . In (d–f), we plot n_{initial}^* , i.e., the minimum value of n_{initial} where P_0 reaches 1, as a function of system size N . To compute n_{initial}^* , we generated 100 random instances for each size, and performed 10 independent runs for each instance. The minimum number of initial configurations that is required by each method to obtain the true ground state was returned as the result of one run. We averaged n_{initial}^* over these 1000 independent

runs, and showed the results of mean and standard error of the mean (SEM) (shaded area, comparable with the line width or data point symbol size in most cases) of different methods on different sizes. Log scale was used for visualization purpose. Note that to have a variety of system sizes, instead of using the same side L for each dimension, we considered different sides for different dimensions. By tuning the sides (L_1, \dots, L_D), we generated a variety of system size $N = L_1 \times \dots \times L_D$. In our implementations, one run of SA and DIRAC-SA needs 5000 initial configurations, and for some of these small systems, one run of DIRAC-SA is able to reach their exact ground states. As a result, the curves of DIRAC-SA are shown to be independent of these presented sizes. Also, the data points for larger N fluctuate more because we calculated fewer samples due to the computational resource limits.

flips a small fraction (e.g., 1%) of the highest- Q spins, rather than just flipping the spin with the highest Q -value as we did in the training phase. In our numerical experiments, we found this batch nodes selection strategy³⁰ reduces the running time significantly without sacrificing much accuracy (Fig. S6). Both time complexity analysis (SI Sec. IE, Tab. S1, Fig. S7) and the performance analysis of finding the ground state (Fig. 4) suggest that DIRAC displays a better scalability than other methods.

We should admit that DIRAC needs to be offline trained while other methods needn't. Yet, we think it is reasonable to compare DIRAC's efficiency without considering its training time, as DIRAC needs to be trained offline only once for each dimension (Fig. S8), and could then be applied infinite times for the systems (of the same dimension) with different sides. Besides, DIRAC's training time is often affordable. For some large systems the total costs required by its training and application are still lower than that of PT. For instance, although DIRAC needs about 2.5 h to finish training on the 3D system, it takes only on average 417 s for DIRAC^m to calculate a random spin glass instance with $D=3, L=20$ ($n_{\text{initial}}=10$). However, to obtain the same

energy, PT needs on average 3 h ($n_{\text{initial}}=5,360$), which is higher than the total time costs of DIRAC^m (about 2.62 h) (Fig. S5f). Note that all the calculations were conducted on a 20-core computer server with 512GB memory and a 16GB Tesla V100 GPU.

Since the biggest computational cost of DIRAC is from the matrix multiplications in SGNN, the graphics processing unit (GPU)-accelerations can be more easily applied on DIRAC than other methods, as the matrix multiplication itself is particularly suitable for paralleling. Still, for the sake of fairness, here we report DIRAC's CPU testing time only, and do not deploy its GPU-accelerations in the application phase. We only utilized GPU to speed up the training process. Hence, the efficiency of DIRAC presented here is rather conservative.

Application on general NP-hard problems

It has been shown that many NP-hard problems, including all of Karp's 21 NP-complete problems (such as the max-cut problem, 3-SAT problem and the graph coloring problem), have Ising formulations¹¹. Hence, we anticipate that DIRAC (with some modifications) could help us solve a wide range of NP-hard

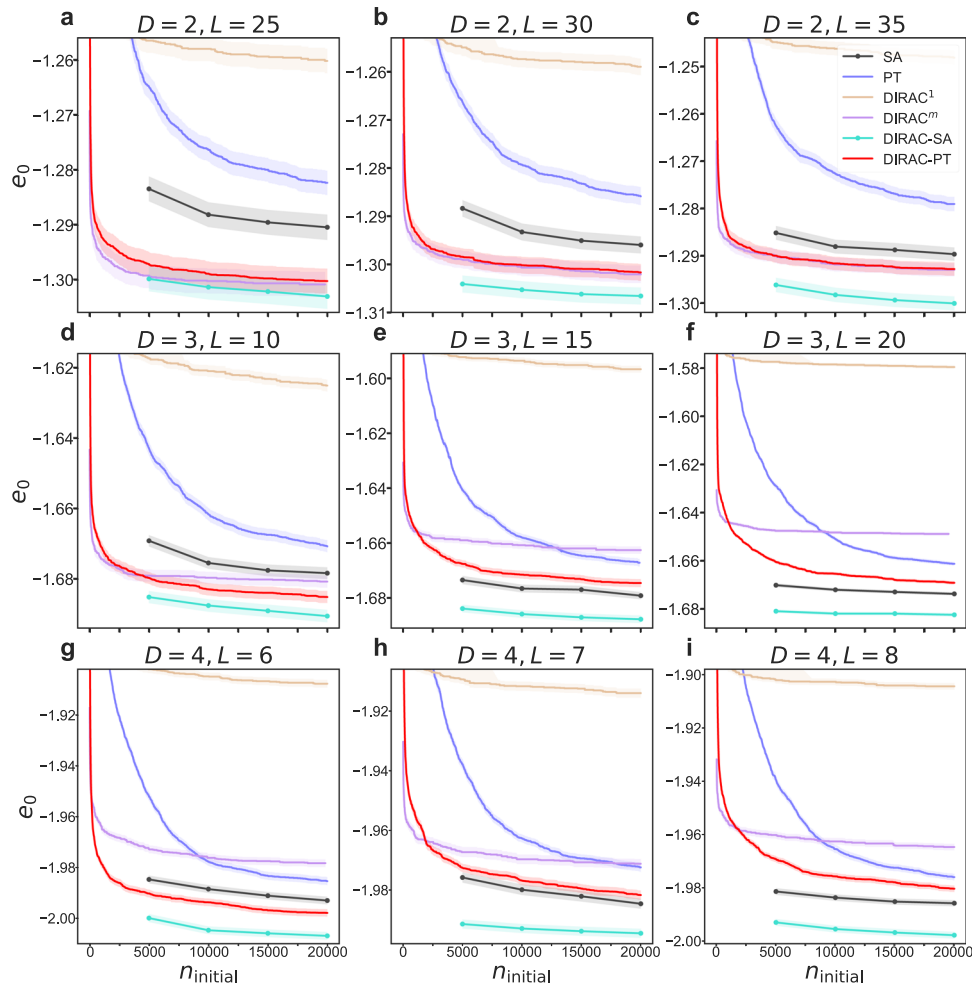


Fig. 5 | Performance of different methods in minimizing the energy. We compared the disorder averaged “ground-state” energy per spin (predicted by each method), denote as e_0 , as a function of the number of initial configurations n_{initial} , to benchmark various methods on large systems. Since the Greedy algorithm performs the worst in minimizing the energy, its results are not shown here so that we can better compare the performance of other methods. We compared on three dimensions ($D = 2, 3, 4$), and for each dimension we consider three different sides (a–i). For each case, we run $n_{\text{initial}} = 2 \times 10^4$ initial configurations for each method. At a given n_{initial} , we chose the lowest energy among all runs for each instance, and averaged the results from 50 independent instances as the final result. We showed their mean and SEM (shaded area) of different algorithms on different sizes. Since

each result of SA and DIRAC-SA is calculated using 5000 initial configurations, the curves of SA and DIRAC-SA actually consist of only four scatter points. Note that here the presented ground state energies reached by our PT implementation are different from the one found in the literature. For example, Ref. ⁵¹ reported an average energy density $e_0 = -1.6981$ (PT) for the 3D-10 ($D = 3, L = 10$) system, while our PT only reached an average $e_0 = -1.6707$ (f). This is simply because different number of initial configurations (n_{initial}) were used. In Ref. ⁵¹, the authors used $n_{\text{initial}} = 3.2 \times 10^7$ initial configurations to reach $e_0 = -1.6981$, while we only used $n_{\text{initial}} = 2 \times 10^4$ initial configurations (i.e., fewer than one thousandth of their n_{initial}) to reach $e_0 = -1.6707$. We know that for those annealing methods, the more initial configurations we explored, the more potential to reach a lower energy.

problems that have Ising spin glass formulations. We emphasize that to make the current DIRAC framework fully compatible with more complex Ising formulations is nontrivial. In the EA spin glass model, we only have pair-wise or two-body interactions, which can be represented by “edges” connecting spins. When the Ising formulation involves k -body interactions (with $k > 2$), we have to leverage the notion of hypergraph and replace the “edge feature” in DIRAC with the “hyperedge feature”⁵¹, which have been heavily studied in the field of hypergraph and hypergraph learning^{52–54}.

We emphasize that GT can be applied to any optimization problem (such as k -SAT⁵⁵ and graph coloring¹¹) with an Ising formulation. Consider a general Ising formulation:

$$\mathcal{H} = - \sum_{a=1}^M J_a \sigma_{a_1} \sigma_{a_2} \cdots \sigma_{a_{k_a}}. \quad (6)$$

Here we have M interactions, and the a -th interaction involves $k_a \geq 1$ Ising spins ($k_a = 1$ corresponds to the external field, $k_a = 2$ corresponds

to the two-body interaction we considered in this paper). Note that GT still works (with $t_i = \pm 1$):

$$\begin{cases} J_a \rightarrow J'_a = J_a t_{a_1} t_{a_2} \cdots t_{a_{k_a}} \\ \sigma_i \rightarrow \sigma'_i = \sigma_i t_i \end{cases} \quad (7)$$

So that the Hamiltonian/energy remains the same:

$$\mathcal{H} \rightarrow \mathcal{H}' = - \sum_{a=1}^M J_a \sigma_{a_1} \sigma_{a_2} \cdots \sigma_{a_{k_a}} t_{a_1}^2 t_{a_2}^2 \cdots t_{a_{k_a}}^2 = - \sum_{a=1}^M J_a \sigma_{a_1} \sigma_{a_2} \cdots \sigma_{a_{k_a}} = \mathcal{H}. \quad (8)$$

As a concrete example, we applied DIRAC to explicitly calculate the max-cut problem (SI Sec. II), a canonical example of the mapping between Ising spin glasses and NP-hard problems¹¹. The results are shown in SI Fig. S9. We found that DIRAC consistently outperforms other competing max-cut solvers.

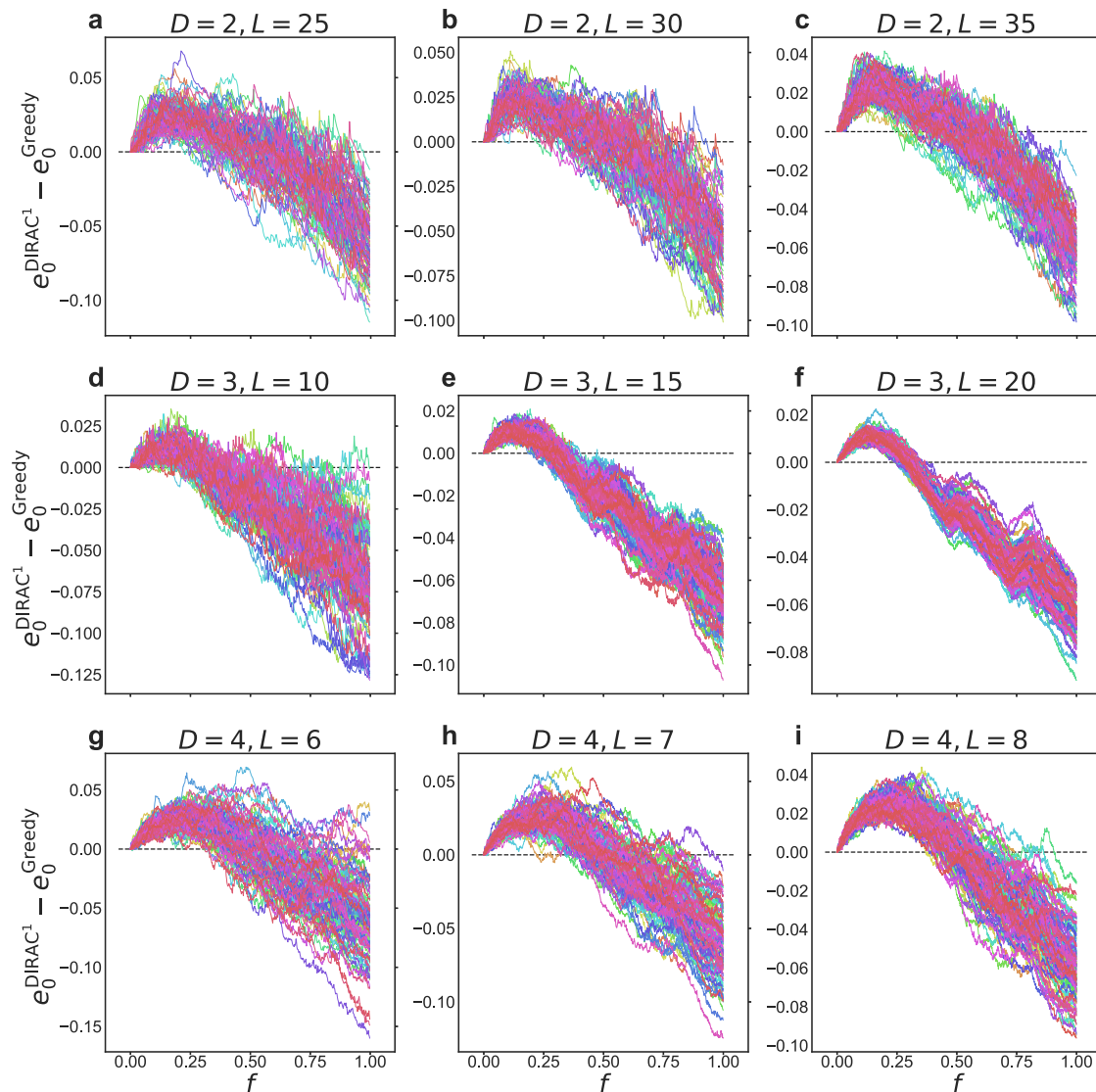


Fig. 6 | A long-sighted greediness. We compared the energy density difference between DIRAC¹ and Greedy ($e_0^{\text{DIRAC}^1} - e_0^{\text{Greedy}}$) at each step. Here e_0 denotes the disorder averaged “ground-state” energy per spin (predicted by each method), and f denotes the fraction of spins that have been flipped. Note that the number of greedy steps of two methods may not be exactly the same, we fill the length of the shorter sequence with its last value, such that the values of these two sequences can be compared one by one. Note that to compare with Greedy step by step more precisely, the results of DIRAC¹ are achieved by flipping only one spin each step. By

contrast, the results of DIRAC¹ in Fig. 5 are achieved by flipping a fraction of top 1% highest- Q -value spins at each step. (a–i) illustrate the energy gaps for different dimensions and different sides. For each size, we tested using 100 randomly generated EA spin glass instances (with couplings sampled from Gaussian distribution $\mathcal{N}(0,1)$), which were represented by 100 curves of different colors. It can be clearly seen that DIRAC¹ always goes through a high-energy state temporarily in the early stage of the greedy process, so as to reach a much lower energy state in the long run.

Interpreting the superior performance of DIRAC

In this section, we offer a systematic interpretation on the superior performances of DIRAC. In Fig. 6, we compare the system’s energy difference between DIRAC¹ and the Greedy algorithm at each greedy step. Note that both methods are performed greedily, the difference is that the former greedily flips the highest- Q -value spin at each step while the latter greedily flips the highest-energy-drop spin at each step. (Note that, for a fair comparison, here in DIRAC¹ at just step we just flip the spin with the highest Q -value, instead of flipping a fraction of spins with highest Q -values.) The (energy-based) Greedy algorithm represents an extremely short-sighted strategy, since it focuses only on each step’s maximal energy drop. Fig. 6 clearly shows that compared to this short-sighted strategy, DIRAC¹ always goes through a high-energy state temporarily for the early steps, so as to reach a much lower energy state in the long run. This result implies that DIRAC has learned to make short-term sacrifices

for long-term gains. In other words, DIRAC has been trained to be mindful of its long-term goals.

In Fig. S10, we demonstrate that, during each iteration of DIRAC^m (which is a sequential running of m iterations of DIRAC¹ connected by GTs), there are two interesting phenomena: (1) the fraction of antiferromagnetic bonds in the gauge transformed instances keeps decreasing (Fig. S10k); and (2) the Q -value distribution becomes more homogeneous (Fig. S10l). In Fig. S3, we show clear evidence that DIRAC^m significantly outperforms m independent DIRAC¹ (where each DIRAC¹ is dealing with a random instance with $\approx 50\%$ antiferromagnetic bonds). These results implies that the superior performance of DIRAC^m is related to the decreasing fraction of antiferromagnetic bonds and a more homogeneous Q -value distribution due to the sequential GTs.

The results shown in Fig. S3 and Fig. S10 prompt us to ask if the superior performance of DIRAC over other methods can be better

visualized in an extreme case, i.e., an anti-ferromagnetic Ising model where $J_{ij} = -1$ for all the bonds. It is well known that this simple model has a ground state with a checkerboard pattern in the spin configuration (as shown in Fig. 7, first row, step = 200, where the red/white sites represent $-1/+1$ spins, respectively). However, for classical energy-based heuristic algorithms (e.g., Greedy, SA and PT), this ground state cannot be found easily. Consider an anti-ferromagnetic Ising model on a 20×20 square lattice with periodic boundary conditions. The last column in Fig. 7 shows the trend of energy vs. the number of steps taken by those heuristic algorithms. For the Greedy algorithm, it ran for 191 steps and got stuck in a local minimum whose energy is significantly higher than the ground state energy. For SA or PT (the coldest replica), it took 21,333 or 10,808 steps to finally reach the ground state, respectively. By contrast, DIRAC only took $200 = 20 \times 20 / 2$ steps (i.e., flipping exactly half of the spins) to reach the ground state for this lattice. In other words, DIRAC did not make any wrong decision in the whole process, which is remarkable.

To further explain why DIRAC is so “smart” in this case, we look at the snapshots shown in Fig. 7. All the different algorithms start from a uniform initial state where all the spin values are set to be +1. Note that in the snapshots, red sites represent spin values -1 . Sites with grayscale colors represent spin values +1, and the grayscale of each site is determined by its Q -value or site-energy. For DIRAC, a darker site corresponds to a higher Q -value; for Greedy, SA and PT, a darker site corresponds to a higher site-energy. All the algorithms always tend to flip a darker site with a higher Q -value or site-energy. But DIRAC differs from other algorithms in the following way. Since the instance composes of purely anti-ferromagnetic bonds, the Q -values of different nodes (spins) are all the same at the beginning. After the first spin is flipped (see the center node in the step = 1 snapshot of DIRAC in Fig. 7), there are two consequences: (1) all its first nearest neighbors’ Q -values are “smartly” decreased, rendering them less likely to be flipped in the future; (2) all its second nearest neighbors’ Q -values are “smartly” increased, rendering them more likely to be flipped in the next step. In a sense, DIRAC has a long-sighted vision that cleverly leverages the nature of a purely anti-ferromagnetic Ising model. As a result, the intermediate snapshots (e.g., in step = 50) display a clear “stripe” pattern that “grows” from the first flipped spin. By contrast, other algorithms are short-sighted, try almost random flips at the beginning, then make incorrect flips and get stuck in the local minimum. The Greedy algorithm got stuck in a local minimum forever. SA and PT can jump out of their local minimum, but it took them very long time to achieve the final ground state.

Taken together, DIRAC seeks to mimic human intelligence in solving the ground state problem. For the spin glass ground state problem, it learns to sacrifice short-term satisfaction for long-term gains. For the anti-ferromagnetic Ising model, it demonstrates a remarkable long-sighted vision by making a smart move every time.

Discussion

This work reports an effective and efficient deep RL-based algorithm, DIRAC, that can directly calculate the ground states of EA spin glasses. Extensive numerical calculations demonstrate that DIRAC outperforms state-of-the-art algorithms in terms of both solution quality and running time. Besides, we also evaluate DIRAC’s superior performances under different scenarios, e.g., different coupling distributions (Gaussian vs. Bimodal vs. Uniform) (Fig. S11); different topological structures (trees vs. loopy trees vs. lattices) (Fig. S12); different hardness regimes (Fig. S13, Fig. S14) and different spin glass models (EA vs. Sherrington-Kirkpatrick) (Fig. S15), see SI Sec. III for more details. Through a pure data-driven way and without any domain-specific guidance, DIRAC smartly mimics the human intelligence in solving the

spin glass ground state problem. In particular, DIRAC enables a much faster finding of ground states than existing algorithms, and it can greatly improve annealing-based methods (e.g., SA and PT) to reach the lowest system energy for all dimensions and sides.

Note that in our implementations of annealing-based methods (e.g., SA and PT), we took the parameters of SA and PT from Ref. ⁵⁰ and Ref. ⁴⁹. We found that our implementations of SA and PT were able to generate similar results as (or, arguably, a slightly better performance than) what were reported in existing works (see SI, Fig. S16 and Fig. S17). We emphasize that even if SA or PT itself can be further improved, we can still use DIRAC as a plug-in to enhance the improved version of SA or PT. Hence, we are not only interested in comparing DIRAC with the state-of-the-art implementation of SA or PT, but also interested in comparing DIRAC-enhanced thermal annealing algorithms with their corresponding vanilla algorithms (as shown in SI Fig. S5).

In the future, advances in deep graph representations may enable us design a better encoder, and developments of RL techniques may help a more efficient training. Both would further improve DIRAC’s performances to find the ground states of Ising spin glasses. The utilization of GT in DIRAC and the way of combining DIRAC and annealing-based methods may also inspire many other physics-guided AI research. Our current framework is just the beginning of such a promising adventure.

Methods

DIRAC

For DIRAC, Tab. S2 lists the values of its hyper-parameters, which were determined by an informal grid search. We only tried to tune a few hyper-parameters, including the discount factor γ , delay reward steps n and the message-passing steps K . The results are shown in Fig. S18, Fig. S19, and Fig. S20. Therefore DIRAC’s performances can be further improved by a more systematic grid search. For example, in Fig. S20, we found that the agent trained using the same K value as that in testing often yields the best performance, and this observation stands for different system sizes. This contradicts our intuition that a larger K should always obtain better performances on large systems due to a better capture of the long-range correlations. We suspect that this may be due to the inconsistency between K and the embedding dimension d (i.e., the size of node embedding vector, which is always set to be $d = 64$ in all our calculations). We anticipate that d should be higher for higher K so that longer correlations can be encoded in the node embedding vector. Systematically testing this idea is beyond the scope of the current study. For more implementation details, please see SI Sec. I.

SA

For SA, we linearly annealed the temperature from a high value to a low one, the number of temperatures is set to be N_t . For each temperature, we performed N_s sweeps of explorations, each sweep contains N (number of spins) random moves. The values of hyper-parameters are determined from Ref. ⁵⁰, i.e., setting the maximal inverse temperature $\beta_{\max} = 5$ and the minimal inverse temperature $\beta_{\min} = 0$. We set $N_t = 100$, which is consistent with the first row in TABLE I in Ref. ⁵⁰. Ref. ⁵⁰ also pointed out that the optimized value of $N_t \times N_s$ should be around 5000. In our case, we set $N_s = 50$ and $N_t = 100$. For more implementation details, please see SI Sec. IV.

PT

For PT, we chose $N_r = 20$ replicas, whose temperatures range from 0.1 to 1.6 with equal interval⁴⁹, initialized with random configurations. Within each epoch, we attempted random flips for N (the number of spins) times. After these random flips, we randomly picked up two replicas and exchanged their spin configurations. The lowest energy and the corresponding spin configuration of all the replicas were

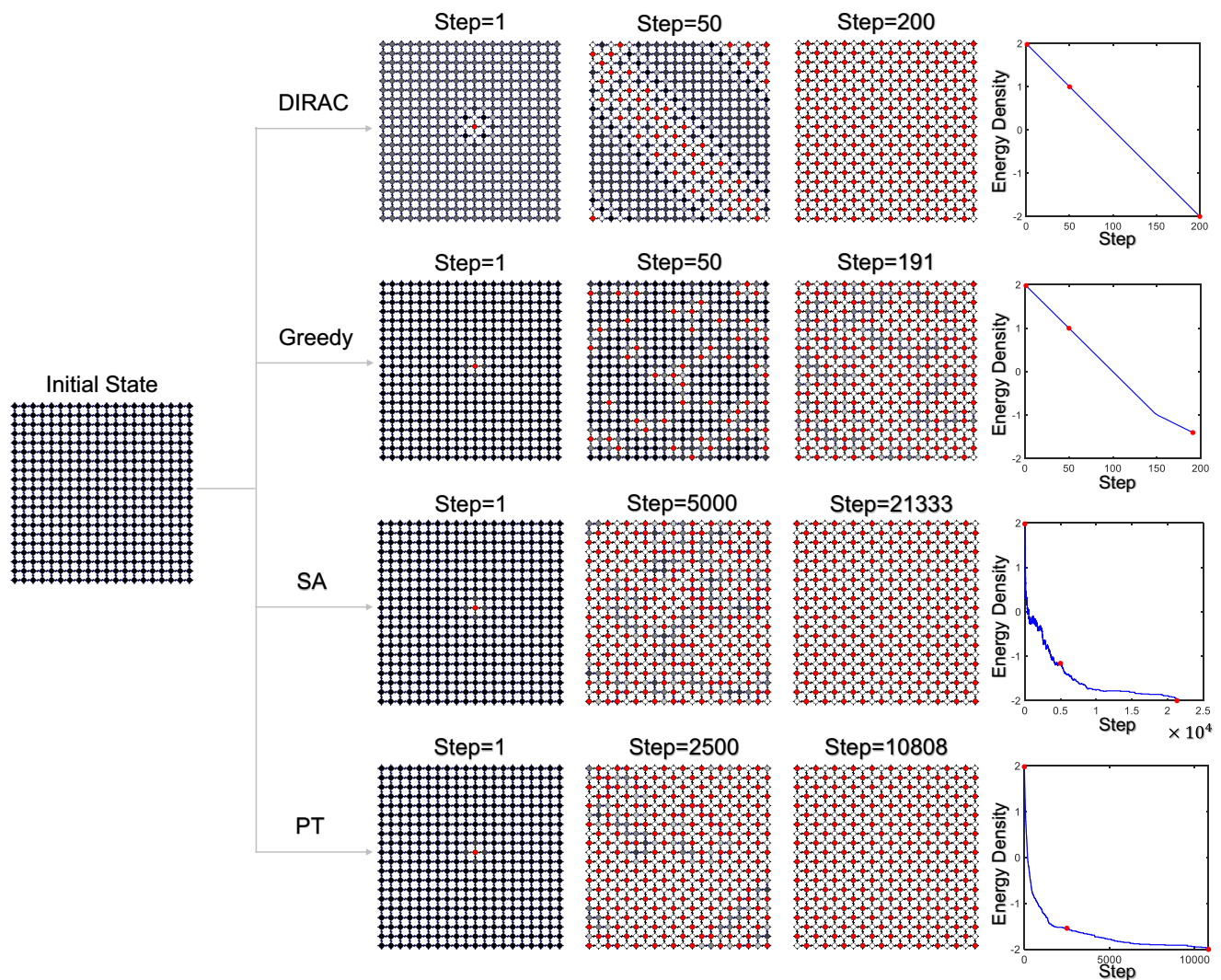


Fig. 7 | DIRAC attempts to mimic human intelligence in finding the ground state of the anti-ferromagnetic Ising model where all couplings $J_{ij} = -1$. Here we demonstrated how different methods, namely DIRAC, Greedy, SA, and PT, reach the ground state in the case of anti-ferromagnetic Ising model where all couplings $J_{ij} = -1$. The PT snapshots and energy trend shown here were chosen from the lowest-temperature replica. Also, for SA, here we set the number of sweeps $N_s = 1$ (instead of 50) because for SA finding the ground state of the anti-ferromagnetic Ising model is a relatively easy task w.r.t finding the spin glass ground state. All the different methods started from a uniform initial state (the leftmost snapshot in the figure), where all the spin values are set to be +1 (black). In the following snapshots, red sites represent sites with spin value -1. The grayscale value was determined as follows: For DIRAC, all the Q -values were rescaled to the range from 0 (the smallest Q -value, white) to 1 (the largest Q -value, black); For Greedy, SA and PT, the site

energies ranging from -4 to +4 are rescaled to the range from 0 (site-energy = -4, white) to 1 (site-energy = +4, black) and take the power of four for clearer visualization. In any case, a darker site is always more likely to be flipped in the next step than a lighter site. In the rightmost column, the trend of the energy density drop is shown. Here the step is defined as each try to flip a spin, so in a given step of SA or PT, the spin is not necessarily flipped, if the try is not accepted in the Metropolis-Hastings procedure. We can see that for the anti-ferromagnetic Ising model, DIRAC demonstrates a remarkable long-sighted vision by making a smart move every time (so that its spin configuration displays a clear checkerboard pattern), implying that it seeks to mimic human intelligence in solving the ground state problem. In supplementary movies, we showed the complete process of how these methods execute on this simple anti-ferromagnetic Ising model.

recorded during the whole process. For more implementation details, please see SI Sec. IV.

Data availability

The data used to reproduce the results in this paper are publicly available through Zenodo⁵⁶ (<https://doi.org/10.5281/zenodo.7562380>).

Code availability

The source code of DIRAC (and its variants), as well as the two baseline methods, SA and PT, are publicly available through Zenodo⁵⁶ (<https://doi.org/10.5281/zenodo.7562380>) or on GitHub (<https://github.com/FFrankyy/DIRAC.git>).

References

1. Binder, K. & Young, A. P. Spin glasses: Experimental facts, theoretical concepts, and open questions. *Rev. Mod. Phys.* **58**, 801 (1986).
2. Mézard, M., Parisi, G. & Virasoro, M. *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications* **9**, (1987).
3. Hartmann, A. K. & Rieger, H. *Optimization algorithms in physics* **2**, (2002).
4. Edwards, S. F. & Anderson, P. W. Theory of spin glasses. *J. Phys. F: Met. Phys.* **5**, 965 (1975).
5. Chayes, J. T., Chayes, L., Sethna, J. P. & Thouless, D. J. A mean field spin glass with short range interactions. *Comm. Math. Phys.* **106**, 41–89 (1986).

6. Ceccarelli, G., Pelissetto, A. & Vicari, E. Ferromagnetic-glassy transitions in three-dimensional ising spin glasses. *Phys. Rev. B* **84**, 134202 (2011).
7. Cugliandolo, L. F. & Kurchan, J. Weak ergodicity breaking in mean-field spin-glass models. *Philos. Mag. B* **71**, 501–514 (1995).
8. Carter, A., Bray, A. & Moore, M. Aspect-ratio scaling and the stiffness exponent θ for ising spin glasses. *Phys. Rev. Lett.* **88**, 077201 (2002).
9. Hartmann, A. K. Scaling of stiffness energy for three-dimensional $\pm J$ ising spin glasses. *Phys. Rev. E* **59**, 84 (1999).
10. Barahona, F. On the computational complexity of ising spin glass models. *J. Phys. A: Math. Gen.* **15**, 3241 (1982).
11. Lucas, A. Ising formulations of many np problems. *Front. Phys.* **2**, 5 (2014).
12. Fortunato, S. Community detection in graphs. *Phys. Rep.* **486**, 75 (2010).
13. Goldstein, R. A., Luthey-Schulten, Z. A. & Wolynes, P. G. Optimal protein-folding codes from spin-glass theory. *Proc. Natl Acad. Sci.* **89**, 4918–4922 (1992).
14. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl Acad. Sci. (USA)* **79**, 2554 (1982).
15. Little, W. A. The existence of persistent states in the brain. *Math. Biosci.* **19**, 101 (1974).
16. Amit, D. J., Gutfreund, H. & Sompolinsky, H. Spin-glass models of neural networks. *Phys. Rev. A* **32**, 1007 (1985).
17. Sompolinsky, H. Statistical mechanics of neural networks. *Phys. Today* **40**, 70 (1988).
18. Mézard, M., Parisi, G. & Zecchina, R. Analytic and algorithmic solution of random satisfiability problems. *Science* **297**, 812–815 (2002).
19. De Simone, C. et al. Exact ground states of ising spin glasses: new experimental results with a branch-and-cut algorithm. *J. Stat. Phys.* **80**, 487–496 (1995).
20. Hartmann, A. K. Ground states of two-dimensional ising spin glasses: fast algorithms, recent developments and a ferromagnet-spin glass mixture. *J. Stat. Phys.* **144**, 519 (2011).
21. Khoshbakht, H. & Weigel, M. Domain-wall excitations in the two-dimensional ising spin glass. *Phys. Rev. B* **97**, 064410 (2018).
22. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).
23. Gubernatis, J. E. The monte carlo method in the physical sciences: celebrating the 50th anniversary of the metropolis algorithm. *The Monte Carlo Method in the Physical Sciences* **690** (2003).
24. Swendsen, R. H. & Wang, J.-S. Replica monte carlo simulation of spin-glasses. *Phys. Rev. Lett.* **57**, 2607 (1986).
25. Geyer, C. J. et al. Computing science and statistics: proceedings of the 23rd symposium on the interface. *American Statistical Association* **156**, (1991).
26. Hukushima, K. & Nemoto, K. Exchange monte carlo method and application to spin glass simulations. *J. Phys. Soc. Jpn.* **65**, 1604–1608 (1996).
27. Earl, D. J. & Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Phys. Chem. Chem. Phys.* **7**, 3910–3916 (2005).
28. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
29. Li, Z., Chen, Q. & Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems* **31**, 539–548 (2018).
30. Fan, C., Zeng, L., Sun, Y. & Liu, Y.-Y. Finding key players in complex networks through deep reinforcement learning. *Nat. Mach. Intell.* **2**, 317–324 (2020).
31. Bello, I., Pham, H., Le, Q. V., Norouzi, M. & Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940* (2016).
32. Nazari, M., Oroojlooy, A., Snyder, L. & Takác, M. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems* **31**, 9861–9871 (2018).
33. Mills, K., Ronagh, P. & Tamblin, I. Finding the ground state of spin hamiltonians with reinforcement learning. *Nat. Mach. Intell.* **2**, 509–517 (2020).
34. Silver, D. et al. Mastering the game of go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
35. Khalil, E., Dai, H., Zhang, Y., Dilkina, B. & Song, L. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems* **30**, 6348–6358 (2017).
36. Mazyavkina, N., Sviridov, S., Ivanov, S. & Burnaev, E. Reinforcement learning for combinatorial optimization: a survey. *Comput. Oper. Res.* **134**, 105400 (2021).
37. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, (2017).
38. Hamilton, W., Ying, Z. & Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems* **30**, 1024–1034 (2017).
39. Velickovic, P. et al. Graph attention networks. In *International Conference on Learning Representations*, (2018).
40. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 1263–1272 (PMLR, 2017).
41. Xu, K., Hu, W., Leskovec, J. & Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, (2018).
42. Barrett, T., Clements, W., Foerster, J. & Lvovsky, A. Exploratory combinatorial optimization with reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* **34**, 3243–3250 (2020).
43. Yao, F., Cai, R. & Wang, H. Reversible action design for combinatorial optimization with reinforcement learning. In *AAAI-22 Workshop on Machine Learning for Operations Research (ML4OR)*, (2021).
44. Wegner, F. J. Duality in generalized ising models and phase transitions without local order parameter. *J. Math. Phys.* **12**, 2259 (1971).
45. Ozeki, Y. Gauge transformation for dynamical systems of ising spin glasses. *J. Phys. A: Math. Gen.* **28**, 3645 (1995).
46. Batista, C. D. & Nussinov, Z. Generalized elitzur’s theorem and dimensional reductions. *Phys. Rev. B* **72**, 045137 (2005).
47. Hamze, F. et al. From near to eternity: spin-glass planting, tiling puzzles, and constraint-satisfaction problems. *Phys. Rev. E* **97**, 043303 (2018).
48. Gurobi Optimization, L. Gurobi optimizer reference manual, (2021).
49. Romá, F., Risau-Gusman, S., Ramirez-Pastor, A. J., Nieto, F. & Vogel, E. E. The ground state energy of the edwards–anderson spin glass model with a parallel tempering monte carlo algorithm. *Phys. A: Stat. Mech. Appl.* **388**, 2821–2838 (2009).
50. Wang, W., Machta, J. & Katzgraber, H. G. Comparing monte carlo methods for finding ground states of ising spin glasses: Population annealing, simulated annealing, and parallel tempering. *Phys. Rev. E* **92**, 013303 (2015).
51. Feng, Y., You, H., Zhang, Z., Ji, R. & Gao, Y. Hypergraph neural networks. In *Proceedings of the AAAI conference on Artificial Intelligence* **33**, 3558–3565 (2019).
52. Yu, C.-A., Tai, C.-L., Chan, T.-S. & Yang, Y.-H. Modeling multi-way relations with hypergraph embedding. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1707–1710 (2018).
53. Gao, Y. et al. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**, 2548–2566 (2020).

54. Pu, L. & Faltings, B. Hypergraph learning with hyperedge expansion. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 410–425 (2012).
55. Ercsey-Ravasz, M. & Toroczkai, Z. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nat. Phys.* **7**, 966–970 (2011).
56. Fan, C. et al. Searching for spin glass ground states through deep reinforcement learning (v1.0.1). *Zenodo* (2023). <https://doi.org/10.5281/zenodo.7562380>.

Acknowledgements

We are grateful to L. Zeng for the valuable discussions. C.F. and Z.L. are supported by National Natural Science Foundation of China (NSFC, 62206303, 62273352, 62073333, 72025405, 72088101), and Ministry of Science and Technology of China (MSTC, 2022YFB3102600).

Author contributions

Y.-Y.L. conceived and designed the project. Y.-Y.L. and Y.S. managed the project. C.F. and M.S. performed all the numerical calculations and analyzed the results, Y.-Y.L., Y.S., Z.N., and Z.L. interpreted the results. C.F., M.S., and Y.-Y.L. wrote the paper, Y.S. and Z.N. edited the paper.

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/s41467-023-36363-w>.

Correspondence and requests for materials should be addressed to Yizhou Sun or Yang-Yu Liu.

Peer review information *Nature Communications* thanks the anonymous, reviewer(s) for their contribution to the peer review of this work. Peer reviewer reports are available.

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

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

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

© The Author(s) 2023