



Demonstrating real advantage of machine learning-enhanced Monte Carlo for combinatorial optimization

Luca Maria Del Bono^{ab}, Federico Ricci-Tersenghi^{abc,1} , and Francesco Zamponi^{a,1} 

Affiliations are included on p. 9.

Edited by David Reichman, Columbia University Department of Chemistry, New York, NY; received November 30, 2025; accepted April 9, 2026

Combinatorial optimization problems are central to both practical applications and the development of optimization methods. While classical and quantum algorithms have been refined over decades, machine learning-assisted approaches are comparatively recent and have not yet consistently outperformed simple, state-of-the-art classical methods. Here, we focus on a class of Quadratic Unconstrained Binary Optimization problems, specifically the challenge of finding minimum energy configurations in three-dimensional Ising spin glasses. We use a Global Annealing Monte Carlo algorithm that integrates standard local moves with global moves proposed via machine learning. We show that local moves play a crucial role in achieving optimal performance. Benchmarking against Simulated Annealing and Population Annealing, we demonstrate that Global Annealing not only surpasses the performance of Simulated Annealing but also exhibits greater robustness than Population Annealing, maintaining effectiveness across problem hardness and system size without hyperparameter tuning. These results provide clear and robust evidence that a machine learning-assisted optimization method can exceed the capabilities of classical state-of-the-art techniques in a combinatorial optimization setting.

disordered systems | machine learning | Monte Carlo | optimization | QUBO problems

Combinatorial optimization problems appear in a variety of real-world applications as well as in fundamental theoretical studies. They consist of finding an optimal state, specified by N discrete variables, that minimizes an objective function over a finite, but exponentially large in N , set. They include, but are not limited to, maximum-satisfiability (MAX-SAT) (1), graph coloring (2), MAX-CUT (3), maximum independent set (4), job scheduling (5), set cover (6, 7), and the traveling salesman problem (8).

Many different algorithms have been introduced in the course of the years to study combinatorial optimization. Exact deterministic solvers are available, but their applicability is limited to moderately large sizes (9–14) due to an exponential increase in the computational cost. Hence, state-of-the-art scalable solvers for combinatorial optimization are instead based on simple local stochastic rules, in which one or a few variables are updated at each step. Starting from a random assignment of variables, local moves are proposed by some heuristics, and accepted according to the achieved gain in the objective function. A prominent example is Simulated Annealing (SA) (15), but other effective heuristics exist (16–21). Quantum algorithms such as Quantum Annealing (22–26) and Quantum Approximate Optimization Algorithms (27, 28) have also been proposed (29). However, these algorithms often fail to find the best solution and can only find approximate ones, often with large relative errors. To date, they do not seem to outperform classical ones (30).

Recently, it has been proposed to use machine learning (ML) methods as a way to generate better heuristics for stochastic search algorithms, in particular because such methods can propose global moves in which most of the variables are updated in a single step (31–38).

Yet, until now, most results have been limited to moderately large sizes and do not seem to outperform classical algorithms. Claims of superiority (39) have been contested (40, 41). Some works have focused in particular on the task of finding the minimum energy configuration of spin glass models (35, 42–44). Also in this case, claims of superiority (36) have been contested (45, 46). Therefore, it is still unclear whether a novel ML-based method can outperform or even perform comparably with state-of-the-art classical algorithms.

To give a clear answer to this question, in this work, we consider a hard benchmark for combinatorial optimization, namely the three-dimensional Edwards-Anderson spin

Significance

In this work, we address a question that has attracted intense interest in recent years: whether machine learning-assisted algorithms can genuinely outperform classical approaches in challenging combinatorial optimization problems. While numerous studies have explored this possibility (particularly using generative or reinforcement learning models), clear, reproducible evidence of a true advantage has remained elusive. Our paper provides a robust demonstration that a machine learning-assisted Monte Carlo algorithm can surpass state-of-the-art classical solvers under controlled and well-benchmarked conditions. By carefully comparing the machine learning-assisted Global Annealing algorithm to state-of-the-art Simulated and Population Annealing on a prototypically hard Quadratic Unconstrained Binary Optimization problem, we establish that the inclusion of learning-assisted global moves leads to a consistent and significant performance gain.

Copyright © 2026 the Author(s). Published by PNAS. This article is distributed under [Creative Commons Attribution-NonCommercial-NoDerivatives License 4.0 \(CC BY-NC-ND\)](#).

¹To whom correspondence may be addressed. Email: federico.ricci@uniroma1.it or francesco.zamponi@uniroma1.it.

This article contains supporting information online at <https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2534768123/-/DCSupplemental>.

Published May 8, 2026.

glass model that provides instances of Quadratic Unconstrained Binary Optimization (QUBO). We perform a fair comparison of an ML-assisted Global Annealing (GA) algorithm with two state-of-the-art solvers, namely SA and Population Annealing (PA) (Fig. 1). We present robust evidence that an ML-assisted algorithm can outperform state-of-the-art solvers under controlled conditions.

More specifically, our main results are as follows.

- We show (Fig. 2) that the GA procedure requires a combination of ML-assisted global moves and simple local moves to be effective, thus confirming intuition from previous theoretical works (47, 48).
- We compare the runtimes of SA, PA, and GA, which can all be implemented in a similar way using the torch (49) environment, thus allowing for a fair comparison of wall-clock times. We show that, for large systems of $N = 10^3$ variables, GA consistently outperforms SA (Fig. 2). Moreover, while it performs worse than PA on easy instances of the problem, it appears to perform on par or better on harder instances, thus showing improved robustness with respect to instance-to-instance hardness fluctuations (Figs. 3 and 4).
- Once scaled to even larger systems of $N = 14^3 = 2,744$ spins, we show that GA consistently outperforms PA (Fig. 5). We note that this is achieved by using the same hyperparameters as in the $N = 10^3$ case, hence showing greater robustness of GA to changes in problem specification. We also point out that these sizes approach the limit for what can be solved with state-of-the-art algorithms.

We stress once again that our comparison is performed using one of the hardest QUBO benchmarks, with problems of large size, under comparable implementations, and using wall-clock times. Hence, we believe that our analysis is fair, and we can make a robust claim of the superiority of ML-assisted algorithms. We also provide some tentative explanations for why GA outperforms state-of-the-art solvers by examining how the combinatorial space is explored along the annealing process (Fig. 6).

The paper is organized as follows. In *Algorithms*, we define precisely the optimization problem under study and the algorithms we consider. In *Results*, we present the main numerical results for our study. In particular, we present extensive results for $N = 10^3$ spins and additional results for $N = 14^3$ spins. In *Discussion*, we discuss our results and present possible future directions for research. Finally, in *Materials and Methods*, we give a detailed description of the algorithms used and of the details of our simulations. Additional data can be found in *SI Appendix*.

Algorithms

Combinatorial Optimization. A wide range of combinatorial optimization problems belongs to the QUBO class (50), where one wants to find $x^* = \operatorname{argmin}_x E(x)$ for

$$E(x) = - \sum_{i < j} Q_{ij} x_i x_j - \sum_i b_i x_i, \quad [1]$$

where $x = (x_1, \dots, x_N)$, with $x_i \in \{0, 1\}$, being a set of N Boolean variables. In statistical physics language, the QUBO class is equivalent to Ising optimization (51), where one wants to find a configuration of N binary Ising variables, $\sigma = (\sigma_1, \dots, \sigma_N)$ with $\sigma_i \in \{-1, +1\}$, called spins, that minimizes an energy function

$$H(\sigma) = - \sum_{i < j} J_{ij} \sigma_i \sigma_j - \sum_i h_i \sigma_i, \quad [2]$$

hence

$$\sigma^* = \operatorname{argmin}_\sigma H(\sigma). \quad [3]$$

We note that, in general, there can be multiple degenerate minima, i.e., solutions of Eq. 3. Many of the previously cited optimization problems can be recast as Ising optimization problems (51).

The energy function, Eq. 2, also plays a very important role in statistical physics, where it is associated with the Gibbs-Boltzmann (GB) distribution,

$$\rho_{\text{GB}}(\sigma) \propto e^{-\beta H(\sigma)}, \quad [4]$$

which defines the probability distribution of configurations at thermal equilibrium with temperature $T = \beta^{-1}$. Depending on the specific choice of the couplings J_{ij} and the fields h_i , Eq. 4 can describe a variety of different systems and phenomena, such as the paramagnetic–ferromagnetic transition at the Curie temperature (52). A particularly interesting choice is considering random symmetric coupling $J_{ij} = J_{ji} \sim \mathcal{N}(0, 1)$ between nearest neighbors on a d -dimensional square lattice. This choice defines the Edwards-Anderson (EA) model (53) for spin glasses, whose energy landscape is particularly complex and hard to optimize for $d \geq 3$. In particular, it has been shown that finding the minimum energy configuration of the system, i.e., solving Eq. 3, is NP-hard for any $d \geq 3$ (36, 54). In this paper, we are interested in studying the $d = 3$ EA model without external magnetic fields ($h_i = 0$), as it provides a set of hard benchmarks in the QUBO class.

Sampling Solvers. Apart from the methods mentioned in the Introduction, energy optimization can also be performed using sampling algorithms, that is, techniques whose broader goal is to obtain configurations distributed according to Eq. 4. Indeed, when $\beta \rightarrow \infty$, the probability distribution in Eq. 4 concentrates on the configuration(s) of minimum energy. Therefore, if one can sample correctly from $\rho_{\text{GB}}(\sigma)$ at all temperatures, then one can also find the minimum energy configuration of the system by slowly decreasing the temperature down to zero. This idea was first implemented by the famous SA algorithm (15). Following this idea, several improved sampling algorithms have been developed, such as Simulated Tempering (55), Parallel Tempering (56) and its modifications (57, 58), and PA (59–61). For completeness, we mention that other classical algorithms based on dynamical systems (62, 63), or tensor-networks (64, 65) exist.

More recently, following the machine learning revolution, it was proposed to use generative modeling architectures to assist sampling (66). This led to a variety of proposals, using architectures such as autoregressive networks (67, 68) and transformers (69), normalizing (70–73) and equivariant (74–79) flows, diffusion models (44, 80–82), stochastic interpolants (83, 84), Boltzmann machines (85) and renormalization-group inspired models (86, 87), which were implemented in different algorithms such as Sequential Tempering (88), Adaptive Monte Carlo (47) and variational neural annealing (89, 90). Additional work has been carried out to use nongenerative techniques (91, 92), as well as mixing quantum and machine learning methods (93).

Despite the numerous and diverse ML-assisted algorithms proposed, there is currently no evidence that any of these can be effective in solving challenging optimization problems. A proper benchmarking of a solver based on an ML-assisted sampling solver approach is the scope of this work.

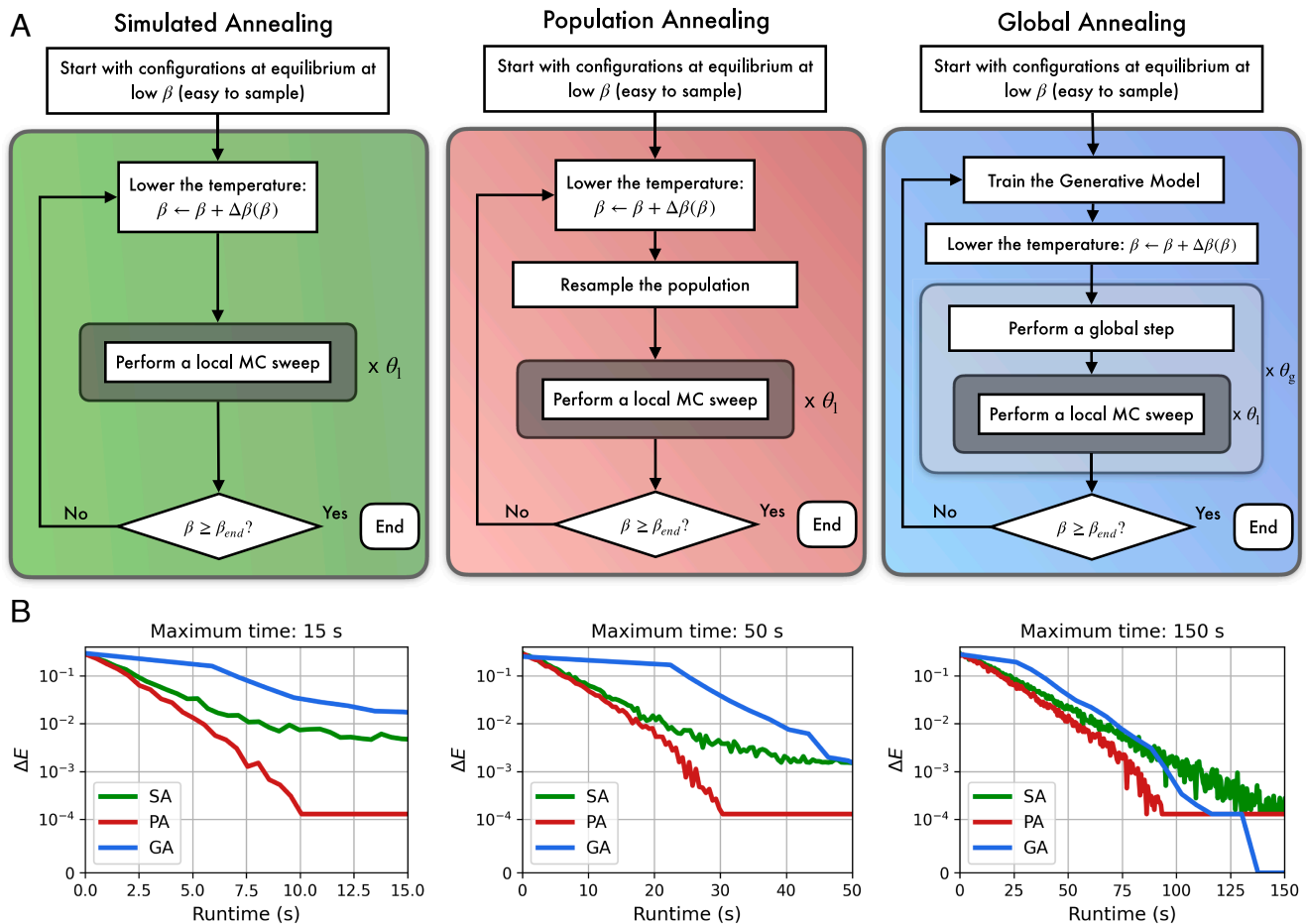


Fig. 1. (A) Schematic description of the three algorithms considered in this work: the well-known Simulated Annealing (SA) and Population Annealing (PA), and the novel machine learning–assisted Global Annealing (GA). θ_1 and θ_g represent the number of local and global steps, respectively. At each annealing step, the temperature is updated according to $\beta \leftarrow \beta + \Delta\beta(\beta)$, where $\Delta\beta(\beta)$ is a (generally not constant) temperature schedule. (B) Example of how the minimum energy decreases during three different runs of various lengths for a given instance of the problem. The difference ΔE between the minimum energy found by the algorithms and the exact one [here found using the Gurobi solver (13)] is plotted as a function of the simulation time for a short (15 s), a medium (50 s), and a long (150 s) run. As the running time increases, the three algorithms manage to progressively find lower energy states. The chosen instance is the same hard instance used for the *Right* panel of Fig. 2 and has $L = 10$, $N = 10^3$.

Details of the Implemented Algorithms. In this work, we compare three different annealing techniques (that is, techniques in which the temperature is monotonously lowered): two classical algorithms, Simulated Annealing and Population Annealing, and the machine learning–assisted Global Annealing. The choice of

SA and PA as benchmarks is motivated by two main reasons: *i*) they are among the algorithms currently achieving the best results, and can therefore be regarded as state-of-the-art solvers for Eq. 3; and *ii*) their implementation is very similar to that of GA, which ensures a fair comparison (Fig. 1). More specifically:

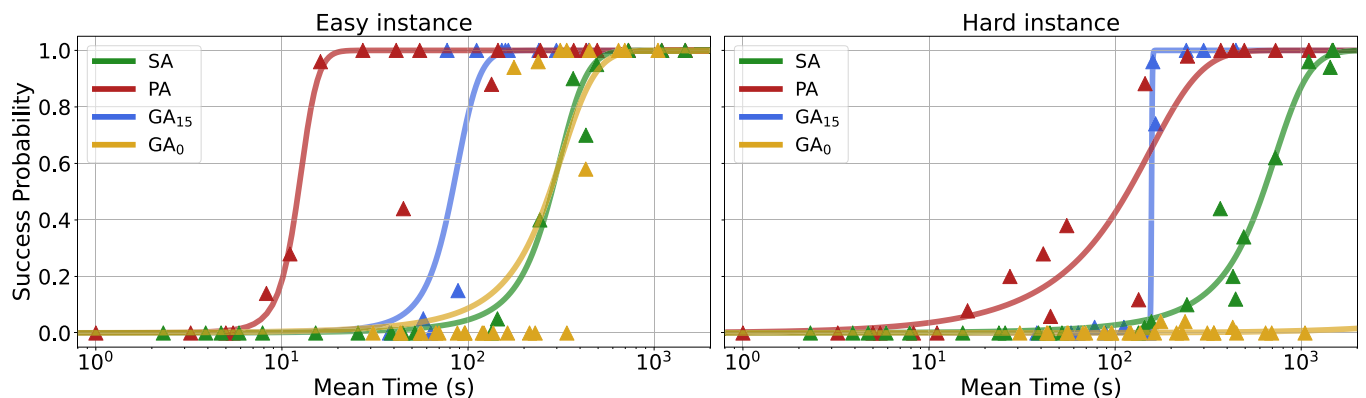


Fig. 2. Success probability as a function of the mean running time for SA, PA, and GA (with and without local moves), together with sigmoidal fits, for easy and hard realizations of the couplings at $N = 10^3$. GA_k identifies the GA algorithm with k local MCSs per global move, so that GA_0 identifies the case in which no local moves are performed.

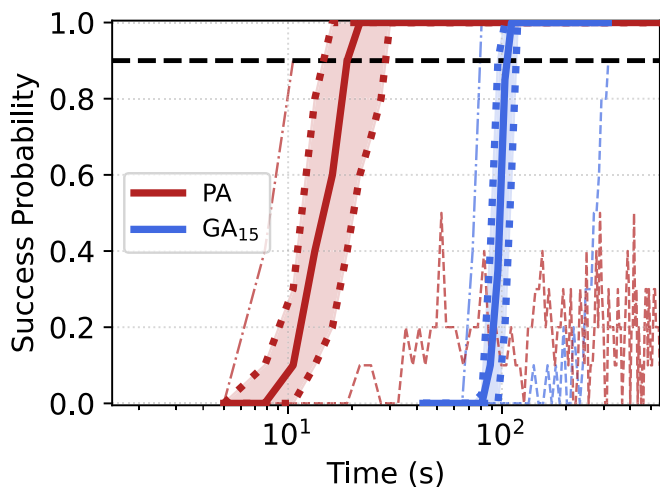


Fig. 3. Median success probability (solid lines) over 200 instances for $N = 10^3$ instances as a function of the running time for PA (red) and GA (blue), together with the 25th and the 75th percentiles (dotted lines, shaded area) and the best (dash-dotted lines) and worst (color dashed) instances. The black dashed line corresponds to a 90% success probability. To reduce computational cost, instances whose runs achieved a 90% success rate were terminated, and a 100% success rate was assumed thereafter for computing the average quantities. While PA tends to outperform GA on the majority of the runs, there are some instances, like the worst case shown here, in which PA performs much worse than GA. For PA, we have performed 10 MCS for every temperature, as in ref. 61. For GA, we performed 5 global moves per temperature, and 15 local MCS per global move. Both algorithms used a logarithmic spacing of the temperatures during annealing.

- **Simulated Annealing (SA)** (15, 94) uses classical local Monte Carlo (MC) moves, in which a single variable σ_i is updated at each time step, by flipping its sign (95). A set of N such moves is referred to as a Monte Carlo Sweep (MCS). The algorithm attempts to sample according to the GB distribution, Eq. 4, at different temperatures: the sampling starts at high temperature, and then T is progressively lowered in small steps. Since Eq. 4 collapses on the minimum energy states when $\beta \rightarrow \infty$, as the temperature is lowered, the typical energy of the states sampled by the local MC progressively decreases. If equilibration can be maintained at all temperatures, one obtains a solution of Eq. 3. Otherwise, one might get stuck at a higher energy, only achieving an approximate solution to the optimization problem.
- **Population Annealing (PA)** (59–61, 96, 97) works similarly to Simulated Annealing, but evolving a whole population of configurations at once. At each step, when the temperature is lowered, the population of configurations is resampled in order to better adapt to the lower temperature. Previous studies (61) found comparable performances with another state-of-the-art classical algorithm, Parallel Tempering (56). Due to its inherently parallel nature, PA is well suited to be implemented on modern GPUs (98).
- **Global Annealing (GA)** (47, 88, 99) works similarly to Simulated Annealing, but instead of performing local, single-spin-flip moves, a generative model is used to propose global moves, in which all the spins are updated at the same time. These global moves are then accepted with a generalized version of the Metropolis criterion (see *Materials and Methods* for the details).* The procedure starts with a population of configurations at high temperature, where sampling from Eq. 4

*Note that, at variance with local MC moves, these global moves require the knowledge of the probability $p_{NN}(\sigma)$ of the configuration σ generated by the model. Therefore, in principle, only models for which $p_{NN}(\sigma)$ can be computed (such as autoregressive models or normalizing flows) fit in this scheme.

is easy. These configurations are used to train a generative model. The temperature is then lowered, and the previously trained network is used to propose moves at this lower temperature. After enough moves, the configurations are used to retrain the generative model. Then, the temperature is lowered once again, and the procedure continues. As in SA and PA, once the temperature is low enough, one considers the minimum energy configuration found as the estimated solution to Eq. 3. Notice that local MC moves can be alternated with the global ones to improve performances (47, 48).

A schematic description of the three algorithms is given in Fig. 14, which shows that they can be implemented in a very similar way, hence allowing for a fair comparison. A more in-depth description is given in *Materials and Methods* for GA and in *SI Appendix* for SA and PA.

We highlight the difference between Global Annealing and another widely used training scheme, Variational Neural Annealing (89). The latter approach aims at training the neural network by minimizing the reverse Kullback–Leibler (KL) divergence between the GB distribution and the distribution obtained via the neural network. This approach does not require samples distributed according to the GB distribution and corresponds to minimizing the free energy of the network distribution. In practice, the network is first trained at high temperature (that is, minimizing the divergence with respect to GB at high T) and then progressively retrained at lower and lower temperatures. Instead, the GA method described above minimizes the direct KL divergence, therefore requires data distributed according to the GB distribution. These data are obtained via a recursive series of Monte Carlo steps, as described above. While there is no direct comparison of the two procedures in the literature, the property of the direct KL to be mode covering (with respect to the reverse

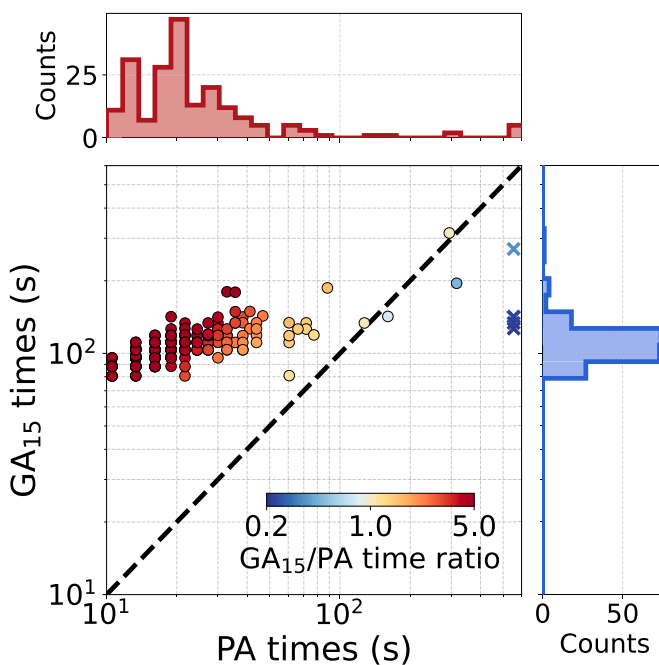


Fig. 4. Scatter plot (in log-log scale) of the times it takes on each instance to reach a 90% success rate with PA and GA₁₅, together with the corresponding histograms. Different colors of the data points highlight different ratios between the runtimes of the algorithms. The dashed line corresponds to equal times. Crosses are points for which PA fails to achieve a 90% success rate within the time limit of about 550 s. Data are the same as those used to obtain Fig. 3.

KL, which is mode seeking)[†] suggests the GA procedure to be better suited for optimization problems.

Fig. 1B presents representative examples of the energy evolution with running time for the algorithms considered in this work. In this case, PA initially attains lower-energy approximate solutions, whereas only GA eventually reaches the exact minimum energy configuration (MEC) (the details on how the MEC is found are reported in *Materials and Methods*). While this example provides a preliminary indication of a potential advantage of GA, it remains anecdotal; in the remainder, we examine this observation systematically through a detailed comparative analysis.

Results

We present a systematic comparison between Global Annealing (GA)—in which local moves alternate with global moves proposed by a generative model—and two state-of-the-art algorithms, Population Annealing (PA) and Simulated Annealing (SA). The minimum energy is found either exactly for smaller system sizes or it is estimated using longer runs for larger system sizes (see *Materials and Methods* for the details). As described in *Algorithms*, in GA simulations, we employed the properly modified Metropolis acceptance criterion for global moves, which accounts for the probability that the generative model produces a given configuration (see *Materials and Methods* for details). We observed that neglecting this correction leads to a severe degradation in GA performance. This emphasizes the importance of using generative architectures for which the probability of generating a specific configuration can be efficiently computed.

Local Moves Are Essential. We start by comparing GA with and without local moves. We call GA_k the GA algorithm with k local MCSs per global move (each sweep being a sequence of N local MC moves), so that GA_0 identifies the case in which no local moves are performed. We consider an easy and a hard[‡] realization of the couplings J_{ij} for a $d = 3$ EA model on a cubic lattice of side $L = 10$ with periodic boundary conditions, hence with $N = 10^3$ variables in total.

In Fig. 2 we plot the success probability as a function of the running time for GA_0 and GA_{15} together with sigmoidal fits. Easy and hard are here defined qualitatively, simply based on the time it takes for the algorithms to find the solution. The runs use a temperature schedule uniformly spaced in logarithmic scale. The running time is varied by changing both the number of global moves and the number of temperatures, and the success probability is computed by running 50 independent runs and then checking which fraction reaches the minimum energy configuration, estimated as discussed in *Algorithms*. We note that the same running time can correspond to different pairs of parameters (number of temperatures and number of local/global moves), which explains the noncomplete monotonicity of the scatter plot. Some longer runs correspond to choices of parameters that take a long time but are not effective at finding the minimum energy configuration.

The figure clearly shows that the addition of local moves is useful in finding the minimum energy configuration of the system.

[†]That is, the direct KL tends to be large for distributions that are zero when the target distribution is nonzero, while the reverse KL tends to be large for distributions that are nonzero where the target distribution is zero. So the former favors distributions that cover all the modes of the target, while the latter favors distributions that are only concentrated on (some of) the modes of the target, even without capturing all of them.

[‡]Here, easy and hard are loosely defined by simply looking at the time it takes for the different algorithms to find the ground state. Other criteria have been studied in the literature (96).

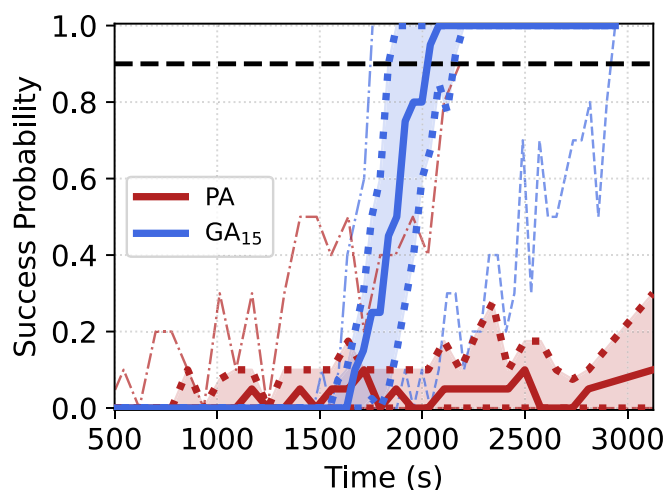


Fig. 5. Same plot as in Fig. 3, here obtained with 10 instances with $N = 14^3$. The choices of the hyperparameters for both algorithms are the same as in Fig. 3. In this case, GA outperforms PA both in the median and in the best/worst case runs.

Not including them leads to heavily deteriorated performances, with GA_0 almost always failing to find the minimum energy configuration in the hard case. We find that while $k = 0$ is not a good choice, performance quickly saturates upon increasing k , hence $k = 15$ is a suitable choice (*SI Appendix*).

Instance-to-Instance Fluctuations. Having assessed the need for local moves, from now on we discard GA_0 and we focus on GA_{15} , which we call GA for simplicity. In Fig. 2 we compare GA with SA and PA. We observe that GA outperforms SA on both instances, hence we will discard SA from subsequent discussions. In contrast, while GA is outperformed by PA on the easy instance, the two methods achieve comparable performance on the hard instance. This seems to suggest that, at this system size, PA performs better than GA on easy instances but on par or worse on harder instances. Yet, we observe an important difference: While PA starts having a nonzero probability of finding the minimum energy configuration before GA, it has a much less sharp transition from low to high success probability. This is an indication that GA is more robust than PA, in the sense that its outcome is more reproducible between distinct runs on the same instance (the success probability jumps very sharply from zero to one). Additional results for other annealing schemes are reported as *SI Appendix*.

In order to obtain a more systematic assessment of the performance of GA versus PA, we consider an ensemble of 200 random independent realizations of the J_{ij} and we repeat the analysis. As we have observed that increasing the number of temperatures improves the performance more than increasing the number of temperatures (especially for PA), here and in the following we vary the runtime by changing the number of temperatures in the annealing while keeping the other parameters fixed. Because random instances of the EA model at the size we are now considering ($N = 10^3$) are typically easy (100), PA tends to outperform GA on the majority of cases. This can be observed in Fig. 3, in which the median success rate over 200 realizations of the couplings J_{ij} , each estimated on 10 different runs, is plotted as a function of the runtime for PA and GA. Remarkably, as evidenced by the worst-case curves, GA tends to perform better on hard instances.

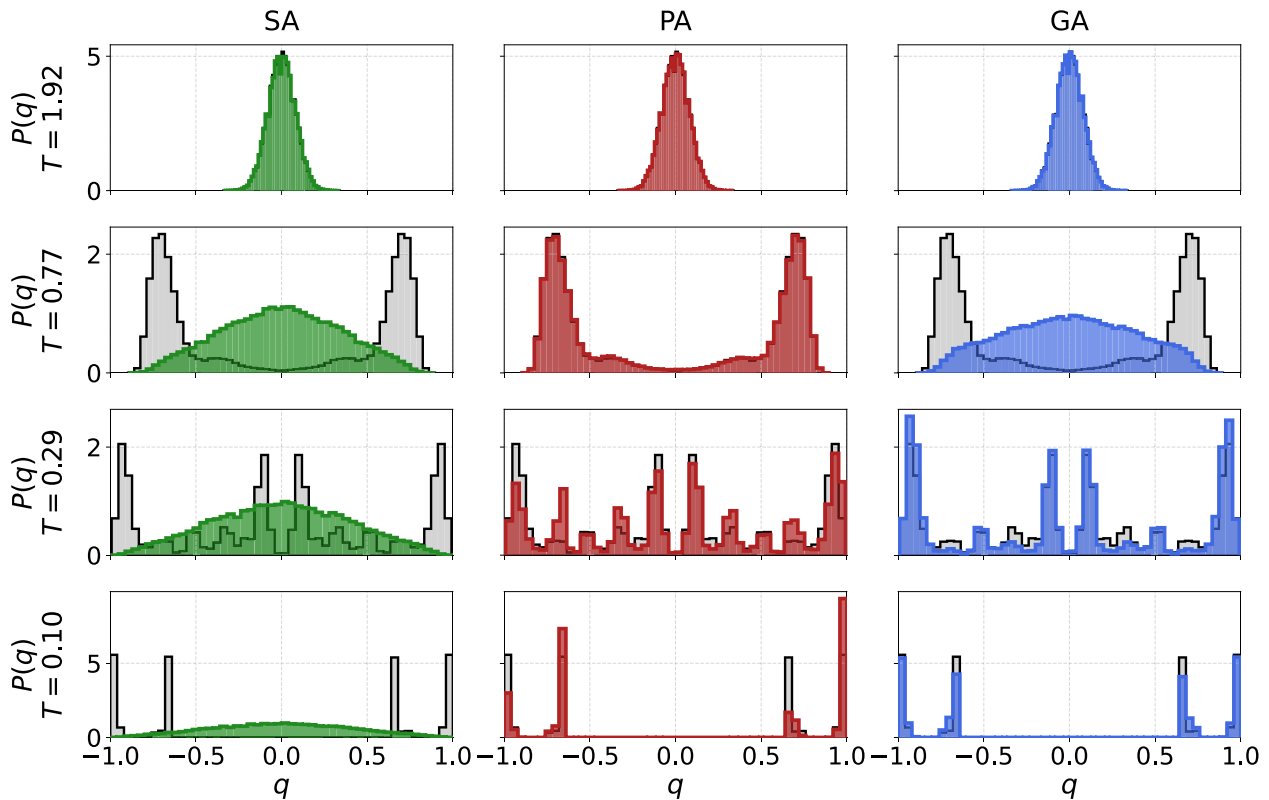


Fig. 6. Probability density of the overlap as obtained by SA, PA, GA (green, red, and blue, respectively) compared to the one obtained by a much longer PA run (gray) at different temperatures for different runs at which the energy minimum is reached. The three different algorithms use runs of different lengths, chosen in order to correspond approximately to the first time at which a 100% success rate is achieved.

This intuition is further confirmed by Fig. 4, in which we compare the time it takes to reach a success rate equal or greater to 90% for the different runs used in Fig. 3. In the majority of instances PA outperforms GA, mainly due to the cost of training the architecture. However, on some harder instances PA is outperformed by GA and in some cases even fails completely to reach a 90% success rates within the given time limit (crosses in Fig. 4). This highlights a greater robustness of the GA algorithm. Moreover, this result also shows the importance of including the training time of the generative model, which is often nonnegligible, to achieve a fair comparison.

Scaling to Larger Sizes. The previous results seem to indicate a greater robustness of the GA algorithm with respect to PA when the problem hardness is increased, with the former not requiring tweaks in the hyperparameters. Yet, at $N = 10^3$ PA remains more efficient in most cases. We thus tested the two algorithms for larger (and therefore harder) instances with $L = 14$, hence $N = 14^3 = 2,744$. In Fig. 5 we consider the success rate as a function of running time for 10 different such instances. We clearly see that in this case GA outperform PA when the same hyperparameters of the $N = 10^3$ case are used. While it is known that for PA one has to increase the population size when N increases (60), this result confirms the better robustness of GA to changes in the problem specification. Moreover, additional tests on PA with a half-as-big or three-times-bigger population did not seem to yield better performances when the total runtime is taken into account.

Fig. 5 presents clear evidence that an algorithm exploiting machine learning techniques can be much more effective than state-

of-the-art classical algorithms. The difference in performance between PA and GA is remarkable, with the worst run of GA taking less than 3,000 s, and being not far from the best run of PA that takes more than 2,000 s.

Overlap Probability Distribution. It is interesting to understand the mechanism that allows SA, PA, and GA to find the minimum energy. To this aim, for each algorithm we consider one successful run on the hard instance of Fig. 2. At each temperature, we consider the set of $M = 2^{17}$ configurations that are annealed in parallel by each algorithm. The overlap (similarity) between two such configurations σ^1 and σ^2 is defined as

$$q = \frac{1}{N} \sigma^1 \cdot \sigma^2 = \frac{1}{N} \sum_{i=1}^N \sigma_i^1 \sigma_i^2, \quad [5]$$

and is a key quantity to verify whether the system has reached thermal equilibrium, i.e. if the distribution over the configurations is given by Eq. 4. In Fig. 6 we compare the probability density of the overlap for SA, PA, and GA during annealing with the equilibrium distribution obtained from a much longer run of PA.

Interestingly, we observe that SA finds the minimum but remains out of equilibrium at all temperatures, producing distributions that differ significantly from the reference equilibrium ones. PA follows the correct distribution more closely across most temperatures, but at the lowest temperatures it fails to reproduce the relative weights of the peaks and even breaks the expected symmetry around $q = 0$. GA shows the opposite trend: It struggles to match the distribution at intermediate temperatures

(mainly due to the large temperature steps and the few training epochs), but at low temperatures it captures the peak weights much more accurately and preserves the symmetry, yielding a closer agreement with the equilibrium reference.

These different mechanisms highlight the distinct ways in which the minimum energy configuration is reached. In SA, the different elements of the population have no means of communicating with one another. As a result, the ensemble as a whole cannot thermalize efficiently. Finding the minimum then becomes essentially a matter of rare events: By chance, one or a few elements of the population may end up in the correct configuration. In contrast, both PA and GA include mechanisms that allow information to be exchanged among different elements of the population. In PA this occurs through the reweighting step, in which lower-energy configurations are preferentially replicated. In this way, population elements that discover good states effectively share this information with others, leading to improved thermalization and facilitating the discovery of lower-energy configurations. However, this requires an increasing population size for larger instances, to maintain sufficient diversity. In GA this information-sharing role is played by the generative model, which extracts information from the entire population and can then propose new moves accordingly.

Discussion

In this work, we studied the application of the machine learning–assisted Global Annealing (also previously called Sequential Tempering) to the optimization of spin glass systems in finite dimension, which provide hard instances in the QUBO class. Specifically, we tested the capabilities of the algorithm on the NP-hard problem of finding the minimum energy configurations of the Edwards-Anderson model in three dimensions. We considered system sizes of $N = 10^3$ and $N = 14^3$, comparable or larger than state-of-the-art studies in the field.

First, we verified the theoretical prediction (48) that standard local Metropolis moves are needed to improve the performance of Global Annealing. An intuitive explanation is as follows. If the generative model proposes moves with the correct GB probability, $\rho_{\text{NN}} \propto e^{-\beta H(\sigma)}$, then global moves become equivalent to temperature-swap moves in Parallel Tempering (PT) (56): Indeed, the acceptance probability described in Eq. 6 of *Materials and Methods* reduces to the temperature-swap acceptance probability of PT. This establishes a direct analogy between GA and PT: In GA, global moves take the role of temperature swaps, while the entire ladder of replicas in temperature is effectively replaced by a single neural network (a substitution that underlies the potential speedup of the GA approach). A key feature of PT is that temperature-swap moves are alternated with local updates. By analogy, this provides a clear motivation for why local moves remain essential in the GA procedure. Additionally, this parallelism hints at the reason why GA is effective: It substitutes the long (and computationally expensive to run) ladder of temperatures of PT with a single generative model.

Moreover, we compared the GA technique with two classical algorithms, Simulated Annealing and Population Annealing. Within a comparable torch implementation, we found that GA consistently outperforms SA. Instead, the comparison between GA and PA is less obvious. While we observed that PA outperforms GA on the majority of the $N = 10^3$ instances, we also noticed that GA is extremely robust and its running time

depends weakly on the hardness of the instance. Finally, for some hard $N = 10^3$ instances and for all of the $N = 14^3$ instances, GA shows a clear superiority over PA, requiring much shorter wall-clock times to reach the energy minimum. As noted in the Introduction, demonstrating the advantage of machine learning methods over classical algorithms in hard optimization problems has been a long-standing goal. Our results provide clear evidence of this advantage.

While the main focus of this work is to compare GA with the state-of-the-art annealing algorithms, it would be interesting to characterize the scaling of resources required by this class of annealing-based algorithms to find the exact minimum energy. This has recently been attempted in optimization problems where the minimum energy is known (101), but in the general case, much more work is needed, and we leave this open question for future work.

We stress that all algorithms have been implemented using the python library torch (49) and run on a single GPU with the same level of optimization, to make the comparison fair. Nonetheless, the algorithms' implementations could be further improved. For instance, SA and PA could benefit from implementations in Cuda C (102) and multispin coding (103), while GA could become faster by the use of jax (104) or torch.compile to achieve a faster training of the generative model. The GA implementation in this work uses the shallow MADE architecture described in *Materials and Methods*. This architecture has a number of parameters scaling as $N^2 = L^6$ in the $d = 3$ case. The use of lighter architectures, such as TwoBo (105), the three-dimensional HAN (106) or 4N (107), could be taken into account to achieve a better scaling. Future experiments could also involve more powerful autoregressive networks, such as transformers (108) or MAMBA (109). While these architectures are probably more expressive and could better fit the target distribution, it is not obvious that they would achieve better computational performance: Indeed, their training could require orders of magnitude more time (e.g., for the transformer architecture introduced in ref. 69), thus severely degrading the overall performance. More advanced architectures could also be developed to incorporate the disorder realization as an additional input, as done, for example, in ref. 110 for the quantum setting. Such architectures would have the advantage of not requiring retraining for each disorder instance, which could in turn improve performance at inference time. Whether these approaches are feasible, applicable to annealing schemes, and more efficient than the one adopted here, especially when one scales the system to progressively larger sizes, remains an open question and a subject for future work.

In this study, we have used GA only for the task of finding the minimum energy. However, GA is much more general, and can also be used for sampling from Eq. 4 at any given temperature. The question of whether GA is effective in sampling is still open (88, 99). The results shown in Fig. 6 suggest that GA can find the minimum energy configurations even if the procedure does not sample equilibrium configurations at the intermediate temperatures. The equilibration (or mixing) times of GA should be more systematically compared to state-of-the-art algorithms.

Finally, in this study, we have considered only one ML-assisted algorithm, Global Annealing, and two classical algorithms, Simulated Annealing and Population Annealing. Many additional algorithms can be tested and compared, as described in the Introduction. Future work should focus on constructing proper benchmarks for a systematic comparison of existing algorithms, to avoid unsubstantiated claims of superiority.

Materials and Methods

General Details. All annealing procedures begin at a high temperature of $T = 1.92$, which is gradually reduced to a low temperature of $T = 0.1$ following a logarithmically spaced schedule.

Local MC updates are carried out in a checkerboard scheme, where spins with odd and even indices are alternately updated in parallel. In this way, a MCS is made of a single odd-indices move followed by a single even-indices move, the two moves combined proposing a flip for all the N spins.

Each algorithm is run with a population of $2^{17} = 131,072$ configurations and is implemented in torch (49). The initial configurations at $T = 1.92$ are assumed to be provided, and they are thermalized through 200 MCS. The runtime of this initial thermalization step is excluded from the reported timings of all algorithms.

Reported runtimes refer to runs on single NVIDIA Tesla V100-SXM2-32GB GPUs for all data in the articles except for the example data in Fig. 1, which were obtained on a NVIDIA Tesla V100S-PCIE-32GB.

General Description of the Global Annealing Procedure. In the GA procedure, one uses a generative model to generate configurations σ' of the system, approximately at equilibrium, i.e., according to Eq. 4 at a temperature β . These configurations are then used as global proposal moves for the MC procedure at $\beta' = \beta + \Delta\beta(\beta) > \beta$, instead of the standard single-spin-flip moves of the local MC algorithm. The move $\sigma \rightarrow \sigma'$ is then accepted with an acceptance probability:

$$\text{Acc}[\sigma \rightarrow \sigma'] = \min \left[1, \frac{\rho_{\text{GB}}(\sigma') \times \rho_{\text{NN}}(\sigma)}{\rho_{\text{GB}}(\sigma) \times \rho_{\text{NN}}(\sigma')} \right], \quad [6]$$

where ρ_{NN} is the probability that the generative model generates a configuration σ . This choice for the acceptance rate guarantees, under ergodicity assumptions, that the distribution over the states is asymptotically given by Eq. 4. The advantage of proposing global moves with the generative model is that all spins are updated simultaneously, making the procedure, in principle, much faster at sampling independent configurations than when only local moves are used. Notice that, if one neglects the ratio $\rho_{\text{NN}}(\sigma)/\rho_{\text{NN}}(\sigma')$ in Eq. 6, detailed balance is no longer satisfied, hence the convergence to Eq. 4 is not guaranteed anymore. Correspondingly, we found that the procedure's performance worsens substantially (not shown).

The general scheme of the Global Annealing procedure is sketched in Fig. 1. It is summarized in Algorithm 1.

Algorithm 1 Global Annealing

b!

- 1: **Input:** Initial inverse temperature β_{start} , final inverse temperature β_{end} , temperature schedule $\Delta\beta(\beta)$, number of configurations M , number of global steps per temperature θ_g , number of local steps per global step θ_l .
- 2: **Initialize:** A set of M equilibrium configurations at β_{start} (sampled e.g. using standard Metropolis MC)
- 3: **while** $\beta < \beta_{\text{end}}$ **do**
- 4: Train a neural network (NN) using the set of M configurations
- 5: Lower the temperature: $\beta \leftarrow \beta + \Delta\beta(\beta)$
- 6: **for** m in $1, \dots, M$ **do**
- 7: Choose the m -th configuration from the set as the initial state
- 8: **for** t in $1, \dots, \theta_g$ **do**
- 9: Propose a new configuration using the NN
- 10: Accept or reject the configuration with probability Eq. 6 at the new temperature $T = 1/\beta$
- 11: **for** t in $1, \dots, \theta_l$ **do**
- 12: Perform a local MC step

Details on the Architecture. In this work, we have used a shallow MADE [Masked Autoencoder for Distribution Estimation, (111)] autoregressive architecture, which is modeled as

$$P(\sigma_i | \sigma_{<i}) = \frac{\exp \left(\sum_{j=1}^{i-1} W_{ij} \sigma_j \right)}{2 \cosh \left(\sum_{j=1}^{i-1} W_{ij} \sigma_j \right)}, \quad [7]$$

where $\sigma_{<i}$ is the set of spins $\sigma_{<i} = \{\sigma_1, \dots, \sigma_{i-1}\}$. We note that the autoregressive approach requires choosing an ordering of the variables. Here, spins are taken in raster order, i.e., from left to right, line by line, and plane by plane.

In practice, the architecture consists of a dense autoregressive layer followed by a sigmoidal activation and has $\mathcal{O}(N^2)$ parameters. This relatively simple design allows us to focus on the annealing procedure rather than on the architectural details. As mentioned in Discussion, alternative choices of the generative architecture could further improve the GA procedure. However, it is not obvious that more sophisticated architectures would necessarily yield better results, since (depending on the type of the architecture) their training could require orders of magnitude more time (as the transformers architecture introduced in ref. 69), which would severely degrade overall performance.

Details on the Training Procedure. Training is performed by minimizing the binary cross-entropy loss (i.e., minimizing the KL divergence between ρ_{NN} and ρ_{GB}). The initial training runs for 40 epochs using the Adam optimizer with learning rate $\eta_0 = 10^{-3}$. We employ an exponential learning-rate schedule that halves the rate every 10 epochs. Early stopping is applied on the training set solely as a plateau detector for the training objective, with a patience of 10 epochs. Each epoch processes the full set of 2^{17} configurations in minibatches of size 256. For retraining at lower temperatures, we perform a single epoch per stage without the previously mentioned regularizations.

Best Estimate of the Minimum Energy Configuration. We consider instances of the $d = 3$ EA model with Gaussian couplings J_{ij} , for which the solution to Eq. 3, i.e., the minimum energy configuration (MEC) or ground state, is unique with high probability. Finding the MEC may require a time scaling superpolynomially in N (as the problem is NP-hard), but we need a fast and reliable estimate of the MEC to perform the comparison among algorithms. For each instance, we define the MEC "best estimate" as the lowest energy state found by all the runs of GA, SA, and PA that we have performed. In many cases, we are confident these algorithms found the exact MEC, while in other cases, they may have found only a very good approximation to the MEC. In any case, the comparison among algorithms is fair and meaningful.

In particular, for the $N = 10^3$ instances shown in Figs. 3 and 4, the MEC has been estimated by performing a set of ten longer runs of GA, with 10 global steps, 30 local steps per global step, and roughly 35 temperatures, each run taking approximately 800 s. For the $N = 14^3$ instances shown in Fig. 5, about 60 temperatures have been used, and each run took about 7,000 s.

For Fig. 2 and for many of the instances in Figs. 3 and 4 we checked that the MEC found with the above criterion coincides with the one obtained by the Gurobi solver (13), run with a 0% gap setting, which guarantees convergence to the exact MEC at the price of an unfavorable scaling of the running time (typically exponential in the problem size). As a further test, in the $L = 10$ case, we solved 2,500 more instances using the same hyperparameters described above (one run per instance). The mean ground state energy found across samples, $-1.6977(5)$, is in excellent agreement with the value $-1.6980(3)$ reported in ref. 61.

For the above reasons, and since for long enough annealing times the best-performing algorithms consistently find the same configuration, we are confident that our best estimate for the MEC is often the exact ground state. We cannot exclude that, in some cases, another lower energy configuration exists, which is never found by any of the algorithms we tested. However, even in these cases, where our best estimate is just a very low energy configuration, the comparison among algorithms is meaningful and fair (which is the aim of the present work).

Data, Materials, and Software Availability. The code and data used in this paper are available at the GitHub repository https://github.com/Laplaxe/MLMC_optimization. (112).

ACKNOWLEDGMENTS. We warmly thank Stefano Bae, Indaco Biazzo, Giulio Biroli, Giuseppe Carleo, Patrick Charbonneau, Marylou Gabrié, and Enzo Marinari, for many interesting discussions related to this work. The research has received financial support from the "National Centre for HPC, Big Data and Quantum Computing," Project CN_00000013, CUP B83C22002940006, NRRP Mission 4 Component 2 Investment 1.4, Funded by the European Union-NextGenerationEU. L.M.D.B. acknowledges funding from the Bando Ricerca Scientifica 2024-Avvio alla Ricerca (D.R. No. 1179/2024) of Sapienza Università di

Roma, project B83C24005280001-MaLeDiSSI. We acknowledge support from the computational infrastructure DARIAH.IT, PON Project code PIRO1_00022, National Research Council of Italy.

Author affiliations: ^aDipartimento di Fisica, Sapienza Università di Roma, Rome 00185, Italy; ^bCNR-Nanotec, Rome unit, Rome 00185, Italy; and ^cIstituto Nazionale di Fisica Nucleare, sezione di Roma1, Rome 00185, Italy

Author contributions: L.M.D.B., F.R.-T., and F.Z. designed research; performed research; analyzed data; and wrote the paper.

The authors declare no competing interest.

This article is a PNAS Direct Submission.

1. C. M. Li, F. Manyà, "Maxsat, hard and soft constraints" in *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications*, A. Biere, H. van Maaren, T. Walsh, Eds. (IOS Press, 2009), vol. 185, pp. 613-631.
2. T. R. Jensen, B. Toft, *Graph Coloring Problems, Wiley-Interscience Series in Discrete Mathematics and Optimization* (Wiley, New York, 1995).
3. M. X. Goemans, D. P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* **42**, 1115-1145 (1995).
4. M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman, New York, 1979).
5. M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems* (Springer, ed. 5, 2016).
6. D. S. Johnson, Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.* **9**, 256-278 (1974).
7. V. Chvátal, A greedy heuristic for the set-covering problem. *Math. Oper. Res.* **4**, 233-235 (1979).
8. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, Eds., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (Wiley, Chichester, 1985).
9. F. Rendl, G. Rinaldi, A. Wiegele, Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Math. Program.* **121**, 307-335 (2010).
10. N. Krislock, J. Malick, F. Roupin, Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. *ACM Trans. Math. Softw.* **43**, 32:1-32:23 (2017).
11. N. Gusmeroli et al., Biqbin: A parallel branch-and-bound solver for binary quadratic problems with linear constraints. *ACM Trans. Math. Softw.* **48**, 24:1-24:29 (2022).
12. J. Charfreitag, M. Jünger, S. Mallach, P. Mutzel, "McSparse: Exact solutions of sparse maximum cut and sparse unconstrained binary quadratic optimization problems" in *2022 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, C. A. Phillips, B. Speckmann, Eds. (SIAM, 2022), pp. 54-66.
13. L. L. C. Gurobi Optimization, Gurobi Optimizer Reference Manual (2024).
14. W. Nakhle, Gta-an atsp method: Shifting the bottleneck from algorithm to ram. arXiv [Preprint] (2025). <http://arxiv.org/abs/2509.13327>. Accessed 22 October 2025.
15. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing. *Science* **220**, 671-680 (1983).
16. E. Marinari, G. Parisi, Effects of changing the boundary conditions on the ground state of ising spin glasses. *Phys. Rev. B* **62**, 11677 (2000).
17. S. Boettcher, "Extremal optimization" in *New Optimization Algorithms in Physics* (Wiley Online Library, 2004), pp. 227-251.
18. M. Alava et al., Circumspect descent prevails in solving random constraint satisfaction problems. *Proc. Natl. Acad. Sci. U.S.A.* **105**, 15253-15257 (2008).
19. U. Benlic, J. K. Hao, Breakout local search for the max-cut problem. *Eng. Appl. Artif. Intell.* **26**, 1162-1173 (2013).
20. M. C. Angelini, F. Ricci-Tersenghi, Monte Carlo algorithms are very effective in finding the largest independent set in sparse random graphs. *Phys. Rev. E* **100**, 013302 (2019).
21. M. Bernaschi et al., How we are leading a 3-xorsat challenge: From the energy landscape to the algorithm and its efficient implementation on gpus. *Europhys. Lett.* **133**, 60005 (2021).
22. T. Kadowaki, H. Nishimori, Quantum annealing in the transverse ising model. *Phys. Rev. E* **58**, 5355-5363 (1998).
23. E. Farhi et al., A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science* **292**, 472-475. See also arXiv:quant-ph/0104129 (2001).
24. G. E. Santoro, R. Martonák, E. Tosatti, R. Car, Theory of quantum annealing of an ising spin glass. *Science* **295**, 2427-2430 (2002).
25. A. Das, B. K. Chakrabarti, Colloquium: Quantum annealing and analog quantum computation. *Rev. Mod. Phys.* **80**, 1061-1081 (2008).
26. L. Fiorini, V. Savona, Entanglement-assisted variational algorithm for discrete optimization problems. *Commun. Phys.* **8**, 438 (2025).
27. E. Farhi, J. Goldstone, S. Gutmann, A quantum approximate optimization algorithm. arXiv [Preprint] (2014). <http://arxiv.org/abs/1411.4028>. Accessed 22 October 2025.
28. L. T. Brady, S. Hadfield, Iterative quantum algorithms for maximum independent set. *Phys. Rev. A* **110**, 052435 (2024).
29. E. Farhi, D. Gamarnik, S. Gutmann, The quantum approximate optimization algorithm needs to see the whole graph: Worst case examples. arXiv [Preprint] (2020). <http://arxiv.org/abs/2005.08747>. Accessed 22 October 2025.
30. V. Bapst, L. Foini, F. Krzakala, G. Semerjian, F. Zamponi, The quantum adiabatic algorithm applied to random optimization problems: The quantum spin glass perspective. *Phys. Rep.* **523**, 127-205 (2013).
31. Y. Bengio, A. Lodi, A. Prouvost, Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.* **290**, 405-421 (2021).
32. H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, L. Song, "Learning combinatorial optimization algorithms over graphs" in *NeurIPS* (2017).
33. W. Kool, H. van Hoof, M. Welling, "Attention, learn to solve routing problems!" in *ICLR* (2019).
34. S. H. Pahng, M. P. Brenner, Predicting ground state configuration of energy landscape ensemble using graph neural network. arXiv [Preprint] (2020). <http://arxiv.org/abs/2008.08227>. Accessed 22 October 2025.
35. A. O. Korol' et al., Calculation of the ground states of spin glasses using a restricted Boltzmann machine. *JETP Lett.* **115**, 466-470 (2022).
36. C. Fan et al., Searching for spin glass ground states through deep reinforcement learning. *Nat. Commun.* **14**, 725 (2023).
37. X. Feng, Z. Yu, Y. Xiong, H. Chen, Sequential stochastic combinatorial optimization using hierarchical reinforcement learning. *Proc. 13th Intl. Conf. Learn. Represent. (ICLR)* (2025).
38. O. Krylova, F. Phillips, Unsupervised learning with gns for qubo-based combinatorial optimization. *EURO J. Comput. Optim.* **13**, 100116 (2025).
39. M. J. Schuetz, J. K. Brubaker, H. G. Katzgraber, Combinatorial optimization with physics-inspired graph neural networks. *Nat. Mach. Intell.* **4**, 367-377 (2022).
40. M. C. Angelini, F. Ricci-Tersenghi, Modern graph neural networks do worse than classical greedy algorithms in solving combinatorial optimization problems like maximum independent set. *Nat. Mach. Intell.* **5**, 29-31 (2023).
41. S. Boettcher, Inability of a graph neural network heuristic to outperform greedy algorithms in solving combinatorial optimization problems. *Nat. Mach. Intell.* **5**, 24-25 (2023).
42. S. Sanokowski, W. Berghammer, J. Kofler, S. Hochreiter, S. Lehner, "One network to approximate them all: Amortized variational inference of ising ground states" in *Machine Learning and the Physical Sciences workshop, NeurIPS 2022* (2022).
43. D. Dobrynin, M. Mohseni, J. P. Strachan, Nonlocal Monte Carlo via reinforcement learning. arXiv [Preprint] (2025). <http://arxiv.org/abs/2508.10520>. Accessed 22 October 2025.
44. S. Sanokowski et al., Scalable discrete diffusion samplers: Combinatorial optimization and statistical physics. arXiv [Preprint] (2025). <http://arxiv.org/abs/2502.08696>. Accessed 22 October 2025.
45. S. Boettcher, Deep reinforced learning heuristic tested on spin-glass ground states: The larger picture. *Nat. Commun.* **14**, 5658 (2023).
46. C. Fan et al., Reply to: Deep reinforced learning heuristic tested on spin-glass ground states: The larger picture. *Nat. Commun.* **14**, 5659 (2023).
47. M. Gabrié, G. M. Rotskoff, E. Vanden-Eijnden, Adaptive Monte Carlo augmented with normalizing flows. *Proc. Natl. Acad. Sci. U.S.A.* **119**, e2109420119 (2022).
48. L. M. Del Bono, F. Ricci-Tersenghi, F. Zamponi, Performance of machine-learning-assisted Monte Carlo in sampling from simple statistical physics models. *Phys. Rev. E* **112**, 045307 (2025).
49. A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library" in *Advances in Neural Information Processing Systems 32*, H. Wallach et al., Eds. (Curran Associates, Inc., 2019), vol. 32, pp. 8024-8035.
50. S. Boettcher, Analysis of the relation between quadratic unconstrained binary optimization and the spin-glass ground-state problem. *Phys. Rev. Res.* **1**, 033142 (2019).
51. A. Lucas, Ising formulations of many np problems. *Front. Phys.* **2**, 5 (2014).
52. L. Onsager, Crystal statistics. I. A two-dimensional model with an order-disorder transition. *Phys. Rev.* **65**, 117 (1944).
53. S. F. Edwards, P. W. Anderson, Theory of spin glasses. *J. Phys. F Met. Phys.* **5**, 965 (1975).
54. F. Barahona, On the computational complexity of ising spin glass models. *J. Phys. A Math. Gen.* **15**, 3241 (1982).
55. E. Marinari, G. Parisi, Simulated tempering: A new Monte Carlo scheme. *Europhys. Lett.* **19**, 451 (1992).
56. K. Hukushima, K. Nemoto, Exchange Monte Carlo method and application to spin glass simulations. *J. Phys. Soc. Jpn.* **65**, 1604-1608 (1996).
57. J. Houdayer, A cluster Monte Carlo algorithm for 2-dimensional spin glasses. *Eur. Phys. J. B. Condens. Matter Complex Syst.* **22**, 479-484 (2001).
58. Z. Zhu, A. J. Ochoa, H. G. Katzgraber, Efficient cluster algorithm for spin glasses in any space dimension. *Phys. Rev. Lett.* **115**, 077201 (2015).
59. K. Hukushima, Y. Iba, Population annealing and its application to a spin glass in AIP Conf. Proc. **690**, 200-206 (2003).
60. J. Machta, Population annealing with weighted averages: A Monte Carlo method for rough free-energy landscapes. *Phys. Rev. E* **82**, 026704 (2010).
61. W. Wang, J. Machta, H. G. Katzgraber, 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).
62. H. Goto et al., High-performance combinatorial optimization based on classical mechanics. *Sci. Adv.* **7**, eabe7953 (2021).
63. E. Ekanayake, N. Shukla, Different paths, same destination: Designing physics-inspired dynamical systems with engineered stability to minimize the ising hamiltonian. *Phys. Rev. Appl.* **24**, 024008 (2025).

64. T. Śmierczalski *et al.*, Spinglasspeps.jl: Tensor-network package for ising-like optimization on quasi-two-dimensional graphs. *SoftwareX* **31**, 102257 (2025).
65. T. Chen, J. Zhang, J. Liu, Y. Deng, P. Zhang, Batchnmc: Efficient sampling of two-dimensional spin glasses using tensor network Monte Carlo. arXiv [Preprint] (2025). <http://arxiv.org/abs/2509.19006>. Accessed 22 October 2025.
66. G. Carleo, M. Troyer, Solving the quantum many-body problem with artificial neural networks. *Science* **355**, 602–606 (2017).
67. D. Wu, L. Wang, P. Zhang, Solving statistical mechanics using variational autoregressive networks. *Phys. Rev. Lett.* **122**, 080602 (2019).
68. S. Wang, Z. Liu, Enhancing the efficiency of variational autoregressive networks through renormalization group. *Phys. Rev. E* **112**, 035310 (2025).
69. S. Bunaiyan, C. Delacour, S. Chowdhury, K. Lee, K. Y. Camsari, Isingformer: Augmenting parallel tempering with learned proposals. arXiv [Preprint] (2025). <http://arxiv.org/abs/2509.23043>. Accessed 22 October 2025.
70. F. Noé, S. Olsson, J. Köhler, H. Wu, Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science* **365**, eaaw1147 (2019).
71. M. Invernizzi, A. Krämer, C. Clementi, F. Noé, Skipping the replica exchange ladder with normalizing flows. *J. Phys. Chem. Lett.* **13**, 11643–11649 (2022).
72. M. Dibak, L. Klein, A. Krämer, F. Noé, Temperature steerable flows and Boltzmann generators. *Phys. Rev. Research* **4**, L042005 (2022).
73. G. Jung, G. Biroli, L. Berthier, Normalizing flows as an enhanced sampling method for atomistic supercooled liquids. *Mach. Learn. Sci. Technol.* **5**, 035053 (2024).
74. J. Köhler, L. Klein, F. Noé, "Equivariant flows: Exact likelihood generative learning for symmetric densities" in *International Conference on Machine Learning* (PMLR, 2020), pp. 5361–5370.
75. G. Kanwar *et al.*, Equivariant flow-based sampling for lattice gauge theory. *Phys. Rev. Lett.* **125**, 121601 (2020).
76. M. S. Albergo, G. Kanwar, P. E. Shanahan, Flow-based generative models for Markov chain Monte Carlo in lattice field theory. *Phys. Rev. D* **100**, 034515 (2019).
77. M. Gerdes, P. de Haan, C. Rainone, R. Bondesan, M. C. Cheng, Learning lattice quantum field theories with equivariant continuous flows. *SciPost Phys.* **15**, 238 (2023).
78. P. de Haan, C. Rainone, M. C. Cheng, R. Bondesan, Scaling up machine learning for quantum field theory with equivariant continuous flows. arXiv [Preprint] (2021). <http://arxiv.org/abs/2110.02673>. Accessed 22 October 2025.
79. C. Schönle, M. Gabrié, T. Lelièvre, G. Stoltz, Sampling metastable systems using collective variables and jarzynski-crooks paths. *J. Comput. Phys.* **527**, 113806 (2025).
80. G. Biroli, M. Mézard, Generative diffusion in very large dimensions. *J. Stat. Mech. Theory Exp.* **2023**, 093402 (2023).
81. S. Bae, E. Marinari, F. Ricci-Tersenghi, Diffusion reconstruction for the diluted Ising model. *Phys. Rev. E* **111**, L023301 (2025).
82. N. T. Hunt-Smith *et al.*, Accelerating Markov chain Monte Carlo sampling with diffusion models. *Comput. Phys. Commun.* **296**, 109059 (2024).
83. M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, Stochastic interpolants: A unifying framework for flows and diffusions. *J. Mach. Learn. Res.* **26**, 1–80 (2025).
84. Y. Chen, E. Vanden-Eijnden, Scale-adaptive generative flows for multiscale scientific data. arXiv [Preprint] (2025). <http://arxiv.org/abs/2509.02971>. Accessed 22 October 2025.
85. A. Decelle *et al.*, The Restricted Boltzmann Machine: from the statistical physics of disordered systems to a practical and interpretative generative machine learning. APS March Meeting Abstracts 2024, W56-004 (2024).
86. T. Marchand, M. Ozawa, G. Biroli, S. Mallat, Multiscale data-driven energy estimation and generation. *Phys. Rev. X* **13**, 041038 (2023).
87. K. Masuki, Y. Ashida, Generative diffusion model with inverse renormalization group flows. arXiv [Preprint] (2025). <http://arxiv.org/abs/2501.09064>. Accessed 22 October 2025.
88. B. McNaughton, M. Milošević, A. Perali, S. Pilati, Boosting Monte Carlo simulations of spin glasses using autoregressive neural networks. *Phys. Rev. E* **101**, 053312 (2020).
89. M. Hibat-Allah, E. M. Inack, R. Wiersma, R. G. Melko, J. Carrasquilla, Variational neural annealing. *Nat. Mach. Intell.* **3**, 952–961 (2021).
90. E. M. Inack, S. Morawetz, R. G. Melko, Neural annealing and visualization of autoregressive neural networks in the Newman-Moore model. *Condens. Matter* **7**, 38 (2022).
91. L. Galliano, R. Rende, D. Coslovich, Policy-guided Monte Carlo on general state spaces: Application to glass-forming mixtures. *J. Chem. Phys.* **161**, 064503 (2024).
92. D. Tzivraïlis, A. Rosso, E. Kawasaki, Uncertainty in ai-driven Monte Carlo simulations. arXiv [Preprint] (2025). <http://arxiv.org/abs/2506.14594>. Accessed 22 October 2025.
93. G. Scriver, E. Costa, B. McNaughton, S. Pilati, Accelerating equilibrium spin-glass simulations using quantum annealers via generative deep learning. *SciPost Phys.* **15**, 018 (2023).
94. S. Caracciolo, A. Hartmann, S. Kirkpatrick, M. Weigel, "Simulated annealing, optimization, searching for ground states" in *Spin Glass Theory and Far Beyond: Replica Symmetry Breaking After 40 Years* (World Scientific, 2023), pp. 1–20.
95. M. E. Newman, G. T. Barkema, *Monte Carlo Methods in Statistical Physics* (Clarendon Press, 1999).
96. C. Amey, J. Machta, Analysis and optimization of population annealing. *Phys. Rev. E* **97**, 033301 (2018).
97. G. Jung, M. Ozawa, G. Biroli, L. Berthier, Numerical investigation of the equilibrium kuzmann transition in a two-dimensional atomistic glass. arXiv [Preprint] (2025). <http://arxiv.org/abs/2507.03590>. Accessed 22 October 2025.
98. L. Y. Barash, M. Weigel, M. Borovský, W. Janke, L. N. Shchur, Gpu accelerated population annealing algorithm. *Comput. Phys. Commun.* **220**, 341–350 (2017).
99. S. Ciarella, J. Trinquier, M. Weigt, F. Zamponi, Machine-learning-assisted Monte Carlo fails at sampling computationally hard problems. *Mach. Learn. Sci. Technol.* **4**, 010501 (2023).
100. F. Martínez-García, D. Porras, Problem hardness of diluted ising models: Population annealing vs simulated annealing. arXiv [Preprint] (2025). <http://arxiv.org/abs/2501.07638>. Accessed 22 October 2025.
101. M. Angelini *et al.*, Algorithmic thresholds in combinatorial optimization depend on the time scaling. *Phys. Rev. X* **16**, 011045 (2026).
102. NVIDIA Corporation, CUDA C Programming Guide (NVIDIA), (2023) Version 12.3.
103. L. Jacobs, C. Rebbi, Multi-spin coding: A very efficient technique for Monte Carlo simulations of spin systems. *J. Comput. Phys.* **41**, 203–210 (1981).
104. J. Bradbury *et al.*, Jax: Composable transformations of python+numpy programs. GitHub repository. <http://github.com/google/jax>. Accessed 23 April 2026.
105. I. Biazzo, D. Wu, G. Carleo, Sparse autoregressive neural networks for classical spin systems. *Mach. Learn. Sci. Technol.* **5**, 025074 (2024).
106. P. Bialas *et al.*, Hierarchical autoregressive neural networks in three-dimensional statistical system. *Comput. Phys. Commun.* **318**, 109892 (2026).
107. L. M. Del Bono, F. Ricci-Tersenghi, F. Zamponi, Nearest-neighbors neural network architecture for efficient sampling of statistical physics models. *Mach. Learn. Sci. Technol.* **6**, 025029 (2025).
108. A. Vaswani *et al.*, Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30**, e3295349 (2017).
109. A. Gu, T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces" in *First Conference on Language Modeling* (2024).
110. L. L. Viteritti *et al.*, Quantum spin glass in the two-dimensional disordered heisenberg model via foundation neural-network quantum states. arXiv [Preprint] (2025). <http://arxiv.org/abs/2507.05073>. Accessed 22 October 2025.
111. M. Germain, K. Gregor, I. Murray, H. Larochelle, "Made: Masked autoencoder for distribution estimation" in *International Conference on Machine Learning* (PMLR, 2015), pp. 881–889.
112. L. M. Del Bono, F. Ricci-Tersenghi, F. Zamponi, MLMC_optimization. GitHub. https://github.com/Laplace/MLMC_optimization. Deposited 22 October 2025.