One Network to Approximate Them All: Amortized Variational Inference of Ising Ground States

Sebastian Sanokowski[1] **Wilhelm Berghammer**[1] **Johannes Kofler**[1]

Sepp Hochreiter[1,2] **Sebastian Lehner**[1,2]

[1] ELLIS Unit Linz and LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Austria
[2] Institute of Advanced Research in Artificial Intelligence (IARAI), Vienna, Austria
sanokowski@ml.jku.at

Abstract

For a wide range of combinatorial optimization problems, finding the optimal solutions is equivalent to finding the ground states of corresponding Ising Hamiltonians. Recent work shows that these ground states are found more efficiently by variational approaches using autoregressive models than by traditional methods. In contrast to previous works, where for every problem instance a new model has to be trained, we aim at a single model that approximates the ground states for a whole family of Hamiltonians. We demonstrate that autoregregressive neural networks can be trained to achieve this goal and are able to generalize across a class of problems. We iteratively approximate the ground state based on a representation of the Hamiltonian that is provided by a graph neural network. Our experiments show that solving a large number of related problem instances by a single model can be considerably more efficient than solving them individually.

1 Introduction

Ising models are of broad interest since they exhibit a remarkably wide range of interesting phenomena in statistical mechanics and provide a universal model of classical spin physics [1; 2]. Besides their popularity in physics, they are frequently applied in disciplines like e.g. molecular biology, operations research, and neuroscience.
The energy of an Ising system with N spins is given by the following Hamiltonian:

$$H = -\sum_{i<j} J_{ij}\sigma_i\sigma_j - \sum_{i=1}^{N} h_i\sigma_i, \tag{1}$$

where $\sigma_i, \sigma_j \in \{-1, 1\}$ are the spins, i.e. the system's degrees of freedom which are located at lattice sites indexed by $i, j \in \{1, \dots, N\}$. The interactions among these spins are given by the couplings $J_{ij} \in \mathbb{R}$ in the first term. The second term represents couplings to external fields $h_i \in \mathbb{R}$. For the problems considered in this work, this term vanishes, because external fields can be absorbed into the coupling term. A specific instance of an Ising model is thus in general defined by the couplings and the external fields. The problem of finding the spin configuration with the lowest energy, i.e the ground state, is known to be equivalent to numerous NP-hard problems including combinatorial optimization (CO) problems, like Max-Cut and the Traveling Salesperson problem [3]. Consequently, while the problem of finding or approximating the ground states of Ising Hamiltonians is of general interest, under the commonly held $\mathbf{P} \neq \mathbf{NP}$ assumption, it is excluded that solutions to all problem instances can be found efficiently. Nevertheless, this does

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

not imply that it is impossible that an optimal solution or an approximation might be efficiently computable for many practically occurring problem instances. Traditionally, methods like simulated annealing (SA) [4] are used to approximate ground states. More recently, variational methods using autoregressive neural networks were shown to yield better approximations of the system's Boltzmann distribution (or of the wave function in the corresponding quantum setting) of Ising-type problems and CO problems [5; 6; 7; 8; 9; 10; 11]. An attractive feature of these methods is that they are general, i.e. they do not rely on a combination with other algorithms or problem-specific heuristics and do not require ground truth data, which is typically problematic [12; 13]. The work of [14] employs meta-learning methods to similar variational problems to obtain more efficient optimization procedures for individual problem instances. However, also here each problem instance requires dedicated fine-tuning. In time and computationally constrained scenarios, however, these approaches may be infeasible since they require a potentially lengthy variational optimization procedure for each new problem instance. We, therefore, aim at a model that approximates solutions to a family of CO problems with a fixed set of model parameters.

Since many CO problems naturally admit a representation as graphs, numerous Graph Neural Network (GNN) [15] applications to CO problems have been demonstrated. As pointed out in a recent review article [16], most of these applications rely either on the availability of ground truth solutions or on the combination with traditional algorithms or handcrafted cost functions. Notable exceptions are the GNN-based approaches in [17; 18]. Similar to the variational approaches discussed above, these require an individual optimization procedure for new problem instances as well. This limitation does not apply to the GNN-based approaches in [19; 20]. However, here the learned distribution over states are assumed to factorize with respect to individual state variables and thus neglect correlations among them. As pointed out in [5], these mean-field approximations are in general inferior compared to the autoregressive neural networks used in aforementioned variational approaches and in this work.

The contribution of the present work is threefold. First, we extend the variational approaches by introducing the additional goal of generalization over families of problem instances. This aspect is analogous to amortized variational inference where the solution to a variational optimization problem is directly approximated by the output of a learned function and not the result of a dedicated optimization procedure [21]. We will refer to our method as Amortized Variational Annealing (AVA). Second, we investigate how the size of the models affect the achievable solution quality and how the model generalises to different problem sizes. Finally, we define a new measure of problem instance difficulty which appears to be a useful indicator for the hardness of CO problem instances.

2 Problem Setting and Methods

We consider the problem of approximating the ground state $\boldsymbol{\sigma}_0 \in \{-1, +1\}^N$ of Ising Hamiltonians with N spins as defined in (Eq. 1). We take a variational approach to this problem and optimize the variational parameters θ of a probability distribution $p_\theta(\boldsymbol{\sigma})$ over the states $\boldsymbol{\sigma} \in \{-1, +1\}^N$ such that a high probability mass is assigned to $\boldsymbol{\sigma}_0$. Once a sufficiently good approximation $\hat{\theta}$ of the optimal variational parameters is found, the ground state is likely to be obtained by sampling states from $p_{\hat{\theta}}(\boldsymbol{\sigma})$ and choosing the sample with the lowest energy. Instead of optimizing $p_\theta(\boldsymbol{\sigma})$ for an individual Hamiltonian H_τ we are interested in constructing a parameterization of $p_\theta(\boldsymbol{\sigma}|H_\tau)$ that adapts to the Hamiltonian of interest without requiring an optimization of θ. That is, we aim at finding a distribution that is conditioned on Hamiltonians which are sampled from a distribution $p_H(H_\tau)$.

In order to prevent our algorithm from being stuck in local minima, we use the Variational Classical Annealing (VCA) approach, where an entropy term is added to the minimization objective and annealed from high to low temperature T (A.5.3). The resulting task is to minimize the following variational free energy with respect to θ:

$$\min_\theta \mathop{\mathbb{E}}_{H_\tau \sim p_H} \mathop{\mathbb{E}}_{\boldsymbol{\sigma} \sim p_\theta(\boldsymbol{\sigma}|H_\tau)} \big[H_\tau(\boldsymbol{\sigma}) + T \log p_\theta(\boldsymbol{\sigma}|H_\tau) \big]. \tag{2}$$

Our model AVA (see Fig. 1) is built up of three components (A.4). The first part consists of a GNN that maps the graph of an Ising Hamiltonian to a node-wise embedding. The embedding of the Ising Hamiltonian is obtained by mapping its couplings on the edges of a graph. We then use a GNN with several message passing steps [22] to obtain an embedding $e_i(H_\tau)$ for each node. Then, an LSTM [23] is used which iteratively processes these embeddings along with the previously generated spin value σ_{i-1}. Finally, an MLP maps the hidden state of the LSTM to the parameter of a Bernoulli distribution for the state at node i. A spin configuration is generated by sampling these distributions

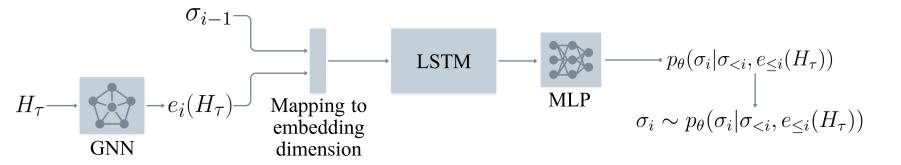

Figure 1: An illustration of our architecture. We first run a forward pass through a GNN in order to obtain a Hamiltonian embedding $e_i(H_\tau)$ for each node. Then, we iterate in an autoregressive manner over each node in the graph, where the LSTM receives the GNN embedding of the current node i and the previously generated spin value $i-1$ as an input. Here, i denotes the index of the currently visited site and $<i$ denotes indices of already visited nodes.

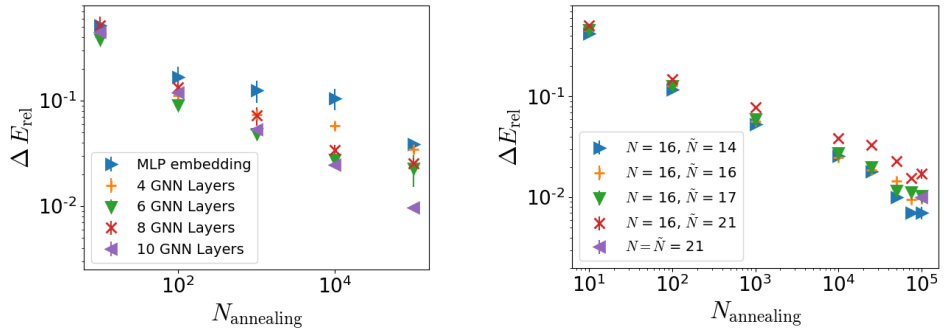

Figure 2: Left: ΔE_{rel} over number of annealing steps for AVA with different types of Hamiltonian embeddings. The network is evaluated on 100 SK-16 test Hamiltonians. Right: ΔE_{rel} over the number of annealing steps for different system sizes in the training and test set. The model is trained on N spins and evaluated on $\tilde{N}$ spins. ΔE_{rel} was evaluated on 300 test Hamiltonians and 50 sampled spin configurations. Error bars in both plots indicate the standard deviation over three independent runs.

autoregressively. The corresponding variational distribution of our autoregressive model can therefore be written as

$$p_\theta(\boldsymbol{\sigma}|H_\tau) = \prod_{i=1}^{N} p_\theta(\sigma_i|\sigma_{<i}, e_{\leq i}(H_\tau)). \tag{3}$$

3 Experiments

In the following, we report results for two classes of Hamiltonians. One class are the Sherrington-Kirkpatrick spin-glass Hamiltonians (A.1.1) with N spins, which we denote as SK-N, and the other class of Hamiltonians represents the Max-Cut problem (A.1.2).

Expressivity of the Embedding Network: To investigate how the model performance depends on the expressivity of the Hamiltonian embedding, we plot in Fig. 2 (left) the relative energy difference ΔE_{rel} (A.1.3) over the number of annealing steps for different numbers of GNN layers on SK-16. The model tends to perform better, the more GNN layers are used. To characterize the inductive bias of the GNN, we compare the GNN embedding with an embedding generated by a comparable MLP (A.4.3). The results indicate that the GNN embeddings allow for more efficient annealing.

Generalisation to other System Sizes: Another advantage of the GNNs compared to the MLPs is that once they were trained on some system size they can in principle be applied to systems of any size. To test the generalization capability to other system sizes, we train the GNN-based model on SK-16. Then, we make a prediction for different system sizes and evaluate ΔE_{rel} using 300 Hamiltonians per system size. The results in Fig. 2 (right) show that the prediction on smaller systems

becomes slightly better and the prediction on larger systems becomes slightly worse compared to the performance on SK-16 itself. The model that is trained on SK-16 and applied on SK-21 (red star) is only slightly worse than the model that is directly trained on SK-21 (purple triangle, shown only for $N_{\text{annealing}} = 10^5$).

Amortisation: Since AVA learns a conditional distribution over a distribution of Hamiltonians while in Variational Neural Annealing (VNA) [8] each model is only trained on one specific Hamiltonian, the task of VNA is considerably easier. Thus, it is not unexpected that for a given amount of annealing steps, VNA performs better than AVA (see direct comparison of decay rates in A.3). On the other hand, an important advantage of our method is that our model is able to approximate ground states of i.i.d. Hamiltonians at inference time, whereas in VNA a new model hast to be trained from scratch for every new Hamiltonian. Therefore, given a certain target ΔE_{rel} we estimate in Fig. 3 (left) at which number of Hamiltonians it is faster to use AVA compared to VNA (A.3).

Hardness Measure: Since not all problem instances from a certain CO problem class are equally hard, [24] propose a simplicity criterion which indicates whether a given CO problem instance can be solved by simply diagonalizing the corresponding Ising Hamiltonian H_τ and discretizing the state $\mathbf{v}^\tau_{\text{min}} \in \mathbb{R}^N$ that minimizes the energy. Based on this simplicity criterion, we introduce for each problem instance H_τ a problem hardness measure $\Upsilon(H_\tau) \in [0, 1]$ that is based on the Hamming distance d between a discretization of $\mathbf{v}^\tau_{\text{min}}$ and the whole set of all k ground states $\{\boldsymbol{\sigma}_{0,1}, ..., \boldsymbol{\sigma}_{0,k}\}$. We define our hardness measure accordingly as

$$\Upsilon(H_\tau) = \frac{1}{N} \min_{\boldsymbol{\sigma} \in \{\boldsymbol{\sigma}_{0,1}, ..., \boldsymbol{\sigma}_{0,k}\}} \Big[d\big(\text{sgn}(\mathbf{v}^\tau_{\text{min}}), \boldsymbol{\sigma}\big) \Big], \tag{4}$$

where larger values indicate harder instances. In Fig. 3 (center), we plot the ground state probability $p_\theta(\boldsymbol{\sigma}_0|H_\tau)$ and ΔE_{rel} from our model over our hardness measure $\Upsilon(H_\tau)$. The latter clearly correlates with the performance of our model, which indicates that this measure is a suitable to determine the hardness of problem instances.

Maximum Cut: In addition, we conduct experiments on Hamiltonians representing the Max-Cut problem of size $N = 50$ and with connection density 0.5 (A.1.2). We compare our method to the Goemans-Williamson (GW) algorithm [25], which is a polynomial-time algorithm that achieves an approximation ratio of ≈ 0.878. This means that even for the worst-case Max-Cut instance GW yields a fairly good expected solution quality [26]. In Fig. 3 (right), we compare GW with AVA by computing for each of 100 test Hamiltonians the Max-Cut ratio (A.1.4) of 50 sampled states. Here, AVA often generates states with a better Max-Cut ratio than the GW algorithm. On average, our method achieves a Max-Cut ratio of ≈ 0.988 which is better than the average Max-Cut ratio of GW of ≈ 0.971 (see A.1.4 for further comparisons).

4 Conclusion and Outlook

In this paper, we use GNNs combined with LSTMs to approximate ground states for a familiy of Ising Hamiltonians without adapting the model to individual Hamiltonians. Our experiments show that longer annealing and more expressive GNNs improve the solution quality considerably. We demonstrate that our method generalises to larger system sizes without any fine tuning.
For the problem sizes studied in this work, exact solutions are still routinely obtained by traditional methods. Future work could investigate how our method can be extended to larger systems and how it performs on real-world data sets. Additionally, it might be interesting to investigate whether a our models can be used as a starting point for problem instance specific fine tuning. Furthermore, we aim to extend the evaluation of our hardness measure on further algorithms and problem classes.

5 Broader Impact

In this work, we develop and characterize a novel approach to combinatorial optimization problems which aims to approximate solutions for problem classes without any instance-specific retraining, i.e.

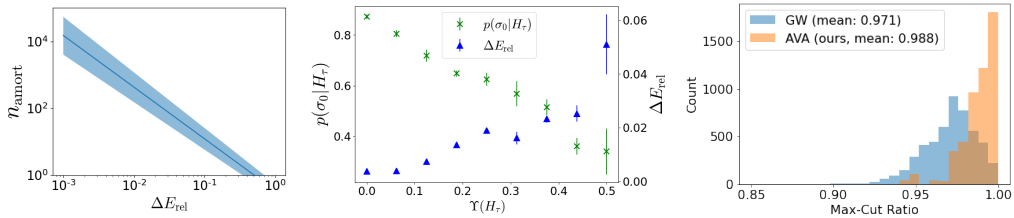

Figure 3: Left: The estimated number of Hamiltonians n_{amort} above which it is faster to use AVA instead of VNA in terms of annealing steps, while reaching the same relative energy difference. Error bars indicate errors from the fitted decay rates (A.3). For VNA we use the same architecture as for AVA (A.5). Center: ΔE_{rel} and probability of the ground state plotted over the hardness measure defined in Eq. 4. Here, due to spin flip symmetry of the SK model, the hardest instances have a hardness of 0.5. The model was evaluated on SK 16 and the error bars indicate the standard error over three independent runs. Right: Comparison of the Max-Cut ratio of 50 sampled states from the AVA and the GW algorithm on the Max-Cut-50 problem with density 0.5. Our AVA model is trained for $2.5 \cdot 10^4$ annealing steps. The evaluation is done on 100 Hamiltonians.

with a fixed set of model parameters. We expect that this method will be useful for many real-world problems in operations research and eventually for applications in material design.

Acknowledgments and Disclosure of Funding

We thank Giuseppe Carleo for his useful comments to this work.

The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (LIT-2018-6-YOU-212), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), AIRI FG 9-N (FWF-36284, FWF-36235), ELISE (H2020-ICT-2019-3 ID: 951847). We thank Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, Software Competence Center Hagenberg GmbH, TÜV Austria, Frauscher Sensonic and the NVIDIA Corporation.

References

[1] A.J. Berlinsky and A.B. Harris. *Statistical Mechanics: An Introductory Graduate Course*. Graduate Texts in Physics. Springer International Publishing, 2019. ISBN 9783030281878.

[2] Gemma De las Cuevas and Toby S Cubitt. Simple universal models capture all classical spin physics. *Science*, 351(6278):1180–1183, 2016.

[3] Andrew Lucas. Ising formulations of many np problems. *Frontiers in physics*, page 5, 2014.

[4] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[5] Dian Wu, Lei Wang, and Pan Zhang. Solving statistical mechanics using variational autoregressive networks. *Physical review letters*, 122(8):080602, 2019.

[6] Or Sharir, Yoav Levine, Noam Wies, Giuseppe Carleo, and Amnon Shashua. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Physical review letters*, 124(2): 020503, 2020.

[7] Mohamed Hibat-Allah, Martin Ganahl, Lauren E Hayward, Roger G Melko, and Juan Carrasquilla. Recurrent neural network wave functions. *Physical Review Research*, 2(2):023358, 2020.

[8] Mohamed Hibat-Allah, Estelle M Inack, Roeland Wiersema, Roger G Melko, and Juan Carrasquilla. Variational neural annealing. *Nature Machine Intelligence*, 3(11):952–961, 2021.

[9] Shoummo Ahsan Khandoker, Jawaril Munshad Abedin, and Mohamed Hibat-Allah. Supplementing recurrent neural networks with annealing to solve optimization problems. *arXiv preprint arXiv:2207.08189*, 2022.

[10] Tianchen Zhao, Giuseppe Carleo, James Stokes, and Shravan Veerapaneni. Natural evolution strategies and variational monte carlo. *Machine Learning: Science and Technology*, 2(2):02LT01, 2020.

[11] Joseph Gomes, Keri A McKiernan, Peter Eastman, and Vijay S Pande. Classical quantum optimization with neural network quantum states. *arXiv preprint arXiv:1910.10675*, 2019.

[12] Gal Yehuda, Moshe Gabel, and Assaf Schuster. It's not what machines can learn, it's what we cannot teach. In *International conference on machine learning*, pages 10831–10841. PMLR, 2020.

[13] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[14] Tianchen Zhao, James Stokes, Oliver Knitter, Brian Chen, and Shravan Veerapaneni. Meta variational monte carlo. *arXiv preprint arXiv:2011.10614*, 2020.

[15] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[16] Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*, 2021.

[17] Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33:6659–6672, 2020.

[18] Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.

[19] Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph neural networks for maximum constraint satisfaction. *Frontiers in artificial intelligence*, 3:580607, 2021.

[20] Haoran Sun, Etash K Guha, and Hanjun Dai. Annealed training for combinatorial optimization on graphs. *arXiv preprint arXiv:2207.11542*, 2022.

[21] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[22] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] Kirill P. Kalinin and Natalia G. Berloff. Computational complexity continuum within ising formulation of np problems. *Communications Physics*, 5:1–10, 2022.

[25] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[26] Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for max cut. *Random Structures & Algorithms*, 20(3):403–440, 2002.

[27] David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, 35(26): 1792, 1975.

[28] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[29] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[30] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[31] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[32] Filippo Vicentini, Damian Hofmann, Attila Szabó, Dian Wu, Christopher Roth, Clemens Giuliani, Gabriel Pescia, Jannes Nys, Vladimir Vargas-Calderón, Nikita Astrakhantsev, et al. Netket 3: Machine learning toolbox for many-body quantum systems. *SciPost Physics Codebases*, page 007, 2022.

[33] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. *Version 0.2*, 5:14–24, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We support our claims in Section 3.
   (b) Did you describe the limitations of your work? [Yes] In Section 4 we discuss the limitations of our work.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We did not yet include code because this work is in progress. However, a public code repository is under preparation.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section A.5 and also A.4 for information about hyperparameters and their corresponding grid searches. For more information on how the data was sampled see A.1.1 and A.1.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars with respect to multiple random seeds in all of our figures.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In section A.5.5 we state on which GPUs the models for different system sizes were be trained on. I addition we report the number of update steps in the training for each of our results.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section A.7.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

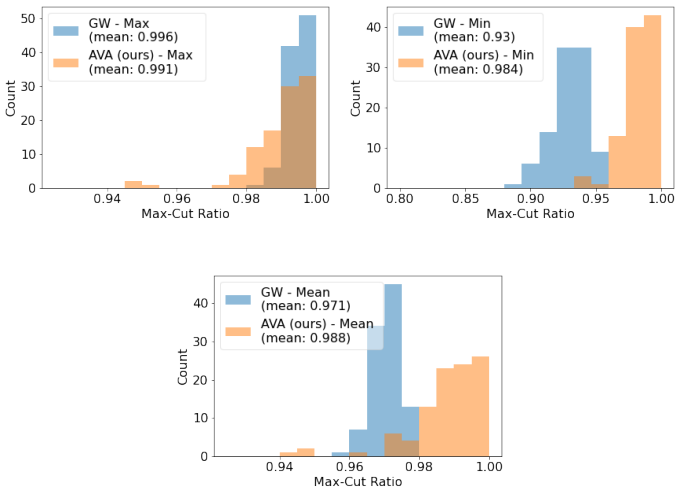

Figure 4: Comparison of the histograms of the maximum (top, left), minimum (top, right) and mean (bottom) Max-Cut ratios over 50 states on 100 test Hamiltonians between AVA and the GW algorithm.

A Appendix

A.1 Definitions of Optimization Problems

A.1.1 Sherrington-Kirkpatrick model

The Sherrington-Kirkpatrick [27] spin-glass model (SK) is defined as

$$H(\boldsymbol{\sigma}) = -\frac{1}{\sqrt{N}} \sum_{i>j} J_{ij}\, \sigma_i\, \sigma_j, \tag{5}$$

where the couplings J_{ij} are sampled independently from a standard Gaussian.

A.1.2 Max-Cut

The maximum cut (MC) of an unweighted graph is defined as

$$\mathrm{MC} = \max \Big\{ \sum_{(i,j)\in\mathcal{E}} \frac{(1-\sigma_i\sigma_j)}{2} \Big\} = \max_{\boldsymbol{\sigma}\in\{-1,1\}^N} \Big\{ \frac{1}{4}\, \boldsymbol{\sigma}^T L\, \boldsymbol{\sigma} \Big\}, \tag{6}$$

where spins $\sigma_{i,j}$ are associated with the graph nodes and $\mathcal{E}$ is the set of all edges. The graph Laplacian $L = D - A$ is defined in term of the degree matrix D and the adjacency matrix A. Since D is diagonal, $\boldsymbol{\sigma}^T D \boldsymbol{\sigma}$ is constant and we can drop this part and just minimise $\boldsymbol{\sigma}^T A \boldsymbol{\sigma}$, which is equivalent to maximizing the MC as defined in Eq. 6. Instead of minimizing $\boldsymbol{\sigma}^T A \boldsymbol{\sigma}$ we minimize $\frac{1}{N}\boldsymbol{\sigma}^T A \boldsymbol{\sigma}$, so that hyperparameters are more easily transferable to larger system sizes. In our experiments, we sample connected graphs with a density of 0.5 and make sure that each node can be reached by any other node.

A.1.3 Relative Energy Difference

We define the relative energy difference of a model with parameters θ with respect to a set of N_H Hamiltonians as

$$\Delta E_\mathrm{rel}(\theta) = \frac{1}{N_\mathrm{H}} \sum_{\tau=1}^{N_\mathrm{H}} \frac{1}{N_\sigma} \sum_{i=1}^{N_\sigma} \frac{H_\tau(\boldsymbol{\sigma}_i) - H_\tau(\boldsymbol{\sigma}_{\tau,0})}{H_\tau(\boldsymbol{\sigma}_{\tau,0})}, \tag{7}$$

where N_σ is the number of sampled states for each Hamiltonian, $\boldsymbol{\sigma}_{\tau,0}$ is the true ground state of Hamiltonian H_τ, and $\boldsymbol{\sigma}_i \sim p_\theta(\boldsymbol{\sigma}|H_\tau)$.

A.1.4 Max-Cut Ratio

We define the Max-Cut ratio as

$$R_\mathrm{MC}(\tau, \boldsymbol{\sigma}) = \frac{\mathrm{MC}_\tau(\boldsymbol{\sigma})}{\mathrm{MC}_\tau(\boldsymbol{\sigma}_{\tau,0})}, \tag{8}$$

where the relation between the approximation ratio (AR) and the Max-Cut ratio is given by

$$\mathrm{AR} = \inf_{\tau} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim p_{\mathrm{Algo}}} \big[R_{\mathrm{MC}}(\tau, \boldsymbol{\sigma})\big], \tag{9}$$

where p_{Algo} refers to the probability distribution stemming from an algorithm from which the configurations $\boldsymbol{\sigma}$ are sampled, and the infimum is taken over the set of all possible graphs.

In Fig. 4 we provide further comparisons of Max-Cut ratios between GW and AVA on MC-50. We compare the maximum (top, left), minimum (top, right) and mean (bottom) Max-Cut ratios of both algorithms. While in most cases our method samples better states than the GW algorithm, the maximum Max-Cut ratio averaged over 100 Hamiltonians of GW of ≈ 0.996 is better than our method which achieves a value of ≈ 0.991.

A.2 Free Energy Minimisation

The Free Energy minimisation objective for one Hamiltonian H_τ is given by

$$F_\tau = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim p_\theta(\boldsymbol{\sigma}|H_\tau)} \Big[H_\tau(\sigma) + T\log(p_\theta(\boldsymbol{\sigma}|H_\tau))\Big]. \tag{10}$$

The gradient for the weight update is then given by

$$\nabla_\theta F_\tau = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim p_\theta(\boldsymbol{\sigma}|H_\tau)} \Big[\Big(H_\tau(\boldsymbol{\sigma}) + T\log(p_\theta(\boldsymbol{\sigma}|H_\tau)) - b\Big)\nabla_\theta \log(p_\theta(\boldsymbol{\sigma}|H_\tau))\Big], \tag{11}$$

where we use $b = \mathbb{E}_{\sigma\sim p_\theta(\boldsymbol{\sigma}|H_\tau)}(H_\tau + T\log(p_\theta(\boldsymbol{\sigma}|H_\tau)))$ for variance reduction (see [8; 5] for details).

In order to obtain the amortised optimisation objective, we calculate the expectation of Eq. 11 over a batch of Hamiltonians.

A.3 Amortisation

Beyond a certain number of problem instances, denoted by n_{amort}, it pays off to use a single AVA model. We estimate this number by dividing the number of annealing steps $N^{\mathrm{AVA}}_{\mathrm{anneal}}$, which AVA needs to obtain a certain ΔE_{rel}, and divide it by the number of annealing steps $N^{\mathrm{VNA}}_{\mathrm{anneal}}$ that VNA takes in order to obtain the same relative energy difference.

The equation for n_{amort} is therefore given by

$$n_{\mathrm{amort}}(\Delta E_{\mathrm{rel}}) = \frac{N^{\mathrm{AVA}}_{\mathrm{anneal}}}{N^{\mathrm{VNA}}_{\mathrm{anneal}}}\Big|_{\Delta E_{\mathrm{rel}}}. \tag{12}$$

In order to estimate this amortization number of Hamiltonians n_{amort}, we fit the relative energy difference as a function of annealing steps as $\Delta E_{\mathrm{rel}} = \Delta E_0 / N^k_{\mathrm{anneal}}$. We fit ΔE_0 and k for both methods on results for SK-16 and obtain fit parameters of $\Delta E_0^{(\mathrm{VNA})} = 10.72 \pm 3.28$, $k_{\mathrm{VNA}} = 1.45 \pm 0.12$ for VNA, and $\Delta E_0^{(\mathrm{AVA})} = 1.29 \pm 0.05$, $k_{\mathrm{AVA}} = 0.45 \pm 0.01$ for AVA. For AVA we have used the results on 100 Hamiltonians and for VNA we have evaluated the method on 36 Hamiltonians. Since the decay rate of VNA is larger than for AVA, VNA will achieve a lower ΔE_{rel} for the same number of annealing steps.

The amortization number for a specific relative energy difference can then be calculated as

$$n_{\mathrm{amort}}(\Delta E_{\mathrm{rel}}) = (\Delta E_0^{(\mathrm{AVA})})^{\frac{1}{k_{\mathrm{AVA}}}} \, (\Delta E_0^{(\mathrm{VNA})})^{-\frac{1}{k_{\mathrm{VNA}}}} \, \Delta E_{\mathrm{rel}}^{-\frac{1}{k_{\mathrm{AVA}}}+\frac{1}{k_{\mathrm{VNA}}}}. \tag{13}$$

In Fig. 3 (left), the shaded region indicates the errors from the fits, using the following upper and lower estimates of the fit parameters. The parameters of the lower bound are estimated by $\Delta E_0^{(\mathrm{AVA})} = 1.29 + 0.05$, $\Delta E_0^{(\mathrm{VNA})} = 10.72 - 3.28$, $k_{\mathrm{AVA}} = 0.45 - 0.01$, $k_{\mathrm{VNA}} = 1.45 + 0.12$ and the parameters of the upper bound are $\Delta E_0^{(\mathrm{AVA})} = 1.29 - 0.05$, $\Delta E_0^{(\mathrm{VNA})} = 10.72 + 3.28$, $k_{\mathrm{AVA}} = 0.45 + 0.01$, $k_{\mathrm{VNA}} = 1.45 - 0.12$.

A.4 Architecture details

A.4.1 LSTM

In our SK experiments, we use three LSTM layers with 120 hidden neurons each. In our Max-Cut experiments we additionally use dilated LSTMs, where the l-th ($l \in \{0, 1, 2\}$) LSTM layer at site i receives the hidden states from the $i - 2^l$-th site. In order to obtain probabilities out of our model, we apply a Probability MLP (ProbMLP) on the hidden state of the LSTM.

A.4.2 Graph Embedding Network

To obtain a graph embedding of the Hamiltonian, we first pass its couplings J_{ij} and possibly also external fields F_i through an Encoder MLP, with which we calculate edge embeddings $E_{ij}^0 = \mathrm{MLP}_{\mathrm{Enc},J}(J_{ij})$ and node embeddings $N_i^0 = \mathrm{MLP}_{\mathrm{Enc},F}(F_i)$.

Afterwards, we use GNNs to apply several message passing updates, which can be written as

$$N_i^{l+1} = \ln(\Phi(N_i^l, \square_{j \in N(i)} \Psi(N_i^l, N_j^l, E_{ij}^l)) + W N_i^l). \tag{14}$$

Here, neighbouring nodes N_i^l, N_j^l and their correspoinding edges E_{ij}^l at layer l are first processed by a message MLP denoted as Φ. The resulting messages are then aggregated by a permutation invariant aggregation $\square_{j \in N(i)}$, which is in our case the mean aggregator. Afterwards, the node update function Ψ is applied on the aggregated messages together with the current node embedding N_i^l. Finally, we use a skip connection, where a weight matrix W is applied on N_i^l and apply layer normalization (ln) [28]. For simplicity, in this work we always employ fully connected graphs where the edge weight denotes whether there is an edge or not.

Similarly, we update edges with

$$E_{ij}^{l+1} = \ln(\Psi_{\mathrm{E}}(N_i^l, N_j^l, E_{ij}^l)) + W_{\mathrm{E}} E_{ij}^l), \tag{15}$$

where Ψ_E is the edge update MLP and W_E is a weight matrix which is used for skip connections.

Our GNN architecture is similar to the Full GN block architecture introduced in [22] but without global graph attributes.

Afterwards, we apply our decoder MLP on the output nodes of the GNN N_i^L to obtain the node-wise Hamiltonian embedding $e_i(H_\tau)$.

Our graph embedding network uses three-layer MLPs with hidden size of [128,96,64] (encoder), [64,64,64] (processor) and [64, 96, 128] (decoder). The decoder MLP, encoder MLP, Φ and Ψ are always three-layer MLPs. Within MLPs we always use Relu [29] activation functions and apply layer normalization after every layer. For Experiments in Fig. 2 (left) and 3 (left) we determined the best learning rate with a grid search in order to obtain good hyperparameters of each setting.

A.4.3 MLP Embedding Network

In order to obtain an MLP embedding, the couplings of the Ising Hamiltonian are concatenated to a vector with fixed ordering. We then use an L-Layer MLP with $h_l \times N$ Neurons, where N is the number of spins and h_l are the number of hidden neurons per spin in layer l. The output $e(H_\tau)$ of the MLP is then reshaped into a matrix e_{ij} of dimension $h_{\mathrm{out}} \times N$, where the j-th column is used as the Hamiltonian embedding at node j.
For the result in Fig. 2 (left) we did a grid search over $\mathrm{lr} \in \{10^{-3}, 5 \cdot 10^{-4}, 10^{-4}\}$, $h_L \in \{64, 80\}\, \forall L$ and over $L \in \{3, 4, 5\}$ in order to find the best hyperparameters for the MLP embedding. $L = 4$, $h_{1,2,3,4} = 64$ and a lr of $1 \cdot 10^{-3}$ worked best in our experiments. The network with the MLP embedding network from Fig. 2 (left) is with $\approx 3.8 \cdot 10^6$ parameters, approximately 3.4 times larger than the network with the graph embedding network with 10 GNN layers, which has $\approx 1.1 \cdot 10^6$ parameters.

A.5 Hyperparemeters

As an optimizer, we use the RAdam optimizer [30]. All other hyperparameters can be taken from Tab. 1. In the following, we provide further details on the Learning Rate Schedule, Annealing Schedule and other implementation choices.

A.5.1 Learning Rate Schedule

In all of our architectures we use a Cosine decay learning rate schedule with restarts [31]. During the warmup phase we use a cosine schedule with restarts of frequency two, where $\mathrm{lr}_{\mathrm{start}}$ is increased by a factor of two and then decreased by the same factor. During Temperature annealing we use restarts with a frequency of 10 while decreasing the learning rate with a cosine schedule down to $\mathrm{lr}_{\mathrm{end}}$. In our experiments we always choose $\mathrm{lr}_{\mathrm{start}} = 10^{-1} \cdot \mathrm{lr}_{\mathrm{end}}$.

A.5.2 Gradient Clipping

In order to use LSTMs we observed that it is necessary to use gradient clipping at a gradient norm of 1.0.

Table 1: Table with Hyperparameters on different problem instances.

Problem Instance	SK				MC
n GNN Layers	4	6	8	10	10
$\mathrm{lr}_{\mathrm{start}}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
N_{warmup}	10^4	10^4	10^4	10^4	$2 \cdot 10^3$
N_{states}	50	50	50	50	50
N_σ	16	16	16	16, 21	50
Figure	2 (left)	2 (left),	2 (left)	2 (left), 2 (right), 3 (center)	3 (right), 4
N_H	50	50	50	50	50
$\mathrm{T}_{\mathrm{start}}$	2.	2.	2.	2.	0.2
ProbMLP	[128,96,2]	[128,96,2]	[128,96,2]	[128,96,2]	[128,2]

A.5.3 Annealing Schedule

We adapt the temperature annealing schedule from [8], where first a warmup phase of N_{warmup} is applied and then the temperature is decreased linearily for N_{anneal} steps from $T = T_{\mathrm{start}}$ to $T = 0$. Each step in N_{warmup} and N_{anneal} contains a cycle of $N_{\mathrm{equil}} = 5$ gradient update steps.

A.5.4 VNA Hyperparameters

For VNA experiments for Fig. 3 (left) we have used the same architecture as for AVA and made a hyperparameter search over the number of GNN layers $L \in \{3, 5, 10\}$ and the $\mathrm{lr} \in \{10^{-4}, 5 \cdot 10^{-4}\}$. Here, the lr of 10^{-4} with 5 GNN layers performed best. In case of VNA, we run $2 \cdot 10^3$ warmup steps. All other hyperparameters were chosen as for SK-16 in Tab. 1.

A.5.5 Computational Resources

All Models for SK-16 can be trained on GTX 1080Ti (11GB) GPUs. For Max-Cut 50 distributed training on two of such GPUs is necessary.

A.6 Evaluation Data

For small system sizes, we use exact diagonalization by reformulating the classical spin system into the quantum setting. This procedure is justified as the (quantum) Ising Hamiltonian with no external fields,

$$H = -J_{ij} \sum_{i>j} \sigma_i^z \otimes \sigma_j^z, \tag{16}$$

is diagonal in the computational basis. Here, $\sigma^z = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$ is the Pauli-z operator, and $H \in \mathbb{C}^{2^N \times 2^N}$. The eigenvectors with the smallest eigenvalues are the ground states $|\Psi_0\rangle \in \mathbb{C}^{2^N}$ and the corresponding eigenvalue is the ground state energy. Since we work with randomly sampled real-valued couplings, there is typically only one ground state solution $|\boldsymbol{\sigma}_0\rangle$ and its spin-flipped counterpart $|\bar{\boldsymbol{\sigma}}_0\rangle$. These are products of computational basis states, i.e., each individual spin is either up ($|\uparrow\rangle = (1,0)^T$) or down ($|\downarrow\rangle = (0,1)^T$), e.g., $|\boldsymbol{\sigma}_0\rangle = |\uparrow\rangle|\downarrow\rangle \cdots |\downarrow\rangle$. There is a trivial equivalence between the quantum states $|\sigma_0\rangle$ and the classical vectors $\boldsymbol{\sigma}_0 \in \{-1, +1\}^N$ as the k-th spin in the product state (up or down) corresponds to the k-th entry of the classical spin vector (-1 or $+1$). Any superposition $c_1|\boldsymbol{\sigma}_0\rangle + c_2|\bar{\boldsymbol{\sigma}}_0\rangle$ (with $|c_1|^2 + |c_2|^2 = 1$) of these ground states is a ground state as well. In the main text and Fig. 3 (center), we use the ground state probability in brief notation $p(\boldsymbol{\sigma}_0)$, which is computed as $p(|\boldsymbol{\sigma}_0\rangle + p(|\bar{\boldsymbol{\sigma}}_0\rangle)$.

For exact diagonalization, we make use of implementation in Netket [32]. For larger system sizes we use the spinglass server http://spinglass.uni-bonn.de/.

A.7 Packages

The code is written in Jax [33]. For the GW algorithm we use the package cvxgraphalgs https://pypi.org/project/cvxgraphalgs/.