Post-error Correction for Quantum Annealing Processor Using Reinforcement Learning

Tomasz Śmierzchalski[1,2(✉)], Łukasz Pawela[2], Zbigniew Puchała[2,3], Tomasz Trzciński[1,4,5], and Bartłomiej Gardas[2]

[1] Faculty of Electronics and Information Technology, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland
[2] Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Bałtycka 5, 44-100 Gliwice, Poland
tsmierzchalski@iitis.pl
[3] Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, Łojasiewicza 11, 30-348 Kraków, Poland
[4] Faculty of Mathematics and Computer Science, Jagiellonian University, Łojasiewicza 6, 30-348 Kraków, Poland
[5] Tooploox Sp. z o.o., Tęczowa 7, 53-601 Wrocław, Poland

Abstract. Finding the ground state of the Ising model is an important problem in condensed matter physics. Its applications spread far beyond physic due to its deep relation to various combinatorial optimization problems, such as travelling salesman or protein folding. Sophisticated new methods for solving Ising instances rely on quantum annealing, which is a paradigm of quantum computation. However, commercially available quantum annealers are still prone to errors, and their ability to find low energetic states is limited. This naturally calls for a post-processing procedure to correct errors. As a proof-of-concept, this work combines the recent ideas revolving around the DIRAC architecture with the Chimera topology and applies them in a real-world setting as an error-correcting scheme for D-Wave quantum annealers. Our preliminary results show how to correct states output by quantum annealers using reinforcement learning. Our approach exhibits excellent scalability, as it can be trained on small instances. However, its performance on the Chimera graphs is still inferior to Monte Carlo methods.

Keywords: Quantum error correction · Quantum annealing · Deep reinforcement learning · Graph neural networks

1 Introduction

Many significant optimization problems can be mapped onto the problem of finding the ground state of the Ising model. Among them are Karp's 21 NP-complete problems [11], the travelling salesman [10], the protein folding problems [1] and

D. Groen et al. (Eds.): ICCS 2022, LNCS 13353, pp. 261–268, 2022.
https://doi.org/10.1007/978-3-031-08760-8_22

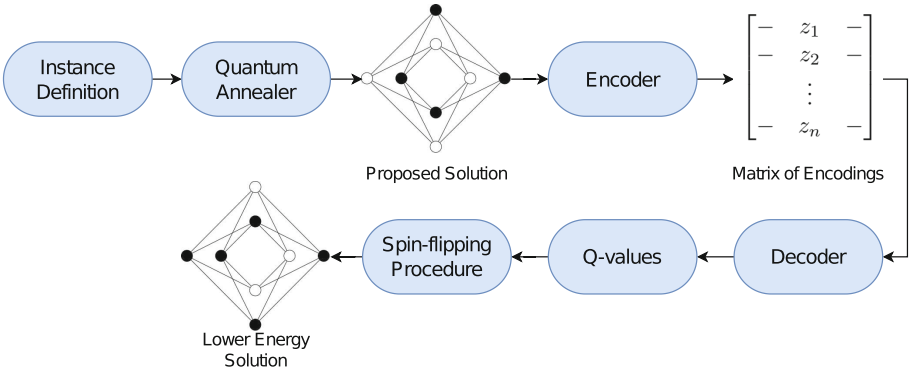

Fig. 1. Overview of our method. Arrows represent subsequent steps. First, we define the Ising instance. Then we obtain the proposed solution from a quantum annealer. White (black) nodes represent spin $\sigma_i = 1$ (-1). Edges represent couplings. In the next step, we encode such an instance using a graph neural network into the matrix of encodings, where each row z_i corresponds to the embedding of a vertex. This matrix is passed through a decoder to obtain Q-values of actions associated with each vertex. The spin flipping procedure involves "flipping" spins one by one according to Q-Values and recording the energy state after each step.

financial portfolio management [16] to name just a few. Promising new methods for solving Ising instances rely on quantum annealing.

The latter is a form of quantum computing, well-tailored for discrete optimization [7,17]. It is closely related to adiabatic quantum computation [12], a paradigm of universal quantum computation which relies on the adiabatic theorem [8] to perform calculations. It is equivalent, up to polynomial overhead, to the better-known gate model of quantum computation [12]. Nevertheless, commercially available quantum annealers are prone to various errors, and their ability to find low energetic states is limited.

Inspired by the recently proposed deep Reinforcement Learning (RL) method for finding spin glass ground states [3], here, we propose a new post-processing error correction schema for quantum annealers called *Simulated Annealing with Reinforcement* (SAwR). This procedure combines deep reinforcement learning with simulated annealing (SA). We employ a graph neural network to encode the Ising instance into an ensemble of low-dimensional vectors used for RL. The agent learns a strategy for improving solutions outputted by the D-Wave annealer. The process of finding the lower energy state involves "flipping" spins one by one according to the learned strategy and recording the energy state after each step. The solution is defined as the lowest energy state found during this procedure. Figure 1 presents an overview of this method. In SAwR, we start with SA, and at low temperature, we replace the Metropolis-Hasting criterion with a single pass of the spin flipping procedure.

2 Results

Ising Problem and Quantum Annealing. The Ising problem is defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Its Hamiltonian is given by [15]:

$$H_{\text{Ising}} = \sum_{\langle i,j \rangle \in \mathcal{E}} J_{ij}\sigma_i\sigma_j + \sum_{i \in \mathcal{V}} h_i\sigma_i, \tag{1}$$

where $\sigma_i \pm 1$ is the i-th spin, $\langle i, j \rangle$ denotes neighbours in $\mathcal{G}$, J_{ij} strength of interaction between i-th and j-th spin, h_i is an external magnetic field. Our goal is to find the ground state, which corresponds to the minimal energy of H_{Ising}, which is NP-hard. Quantum annealing is one of the methods for finding the ground state of (1). This can be achieved by the adiabatic evolution from the initial Hamiltonian H_X of the transverse field Ising model to the final Hamiltonian H_{Ising}. The Hamiltonian for such process is described by $\mathcal{H}(t) = A(t)H_X + B(t)H_{\text{Ising}}$, where $H_X = \sum_i \hat{\sigma_i}^x$ and $\hat{\sigma_i}^x$ is the standard Pauli X matrix acting on the i-th qubit. The function $A(t)$ decreases monotonically to zero, while $B(t)$ increases monotonically from zero, with $t \in [0, t_f]$, where t_f denotes the annealing time [6,17].

Reinforcement Learning Formulation. We consider a standard reinforcement learning setting defined as a Markov Decision Process [18] where an agent interacts with an environment over a number of discrete time steps $t = 0, 1, \ldots, T$. At each time step t, the agent receives a state s_t and selects an action a_t from some set of possible actions $\mathcal{A}$ according to its policy π, where π is a mapping from set of states $\mathcal{S}$ to set of actions $\mathcal{A}$. In return, the agent receives a scalar reward r_t and moves to the next state s_{t+1}. The process continues until the agent reaches a terminal state, s_T, after which the process restarts. We call one pass of such a process an episode. The return at time step t, denoted, $R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$ is defined as a sum of rewards that the agent will receive for the rest of the episode discounted by the discount factor $\gamma \in (0, 1]$. The goal is to maximize the expected return from each state s_t. The basic components, in the context of the Ising spin-glass model, are:

- **State**: s represents the observed spin glass instance, including the spin configuration σ, the couplings $\{J_{ij}\}$ and values of the magnetic field $\{h_i\}$.
- **Action**: $a^{(i)}$ refers to the flip of spin i, i.e., changing is value to the opposite. For example, after the agent performs an action, $a^{(i)}$, $\sigma_i = 1$ becomes -1.
- **Reward**: $r(s_t; a_t^{(i)}; s_{t+1})$ is the energy change after flipping spin i from state s_t to a new state s_{t+1}.

Starting at $t = 0$, an agent flips one spin during each time step, which moves him to the next state (different spin configuration). The terminal state s_T is met when the agent has flipped each spin. The solution is defined as the spin configuration σ corresponding to the lowest energy state found during this procedure. An action-value function $Q^\pi(s, a) = \mathbb{E}(R_t \mid s_t = s,\ a_t = a)$ is the expected return for selecting action a in state s and following policy π. The value $Q^\pi(s, a)$ is often called Q-value of action a in state s. The optimal action-value function, $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ which gives the maximum action value for state

s and action a achievable by any policy. As learning of the optimal action-value function is in practice infeasible, We seek to learn the function approximator $Q(s, a; \Theta) \approx Q^*(s, a)$ where Θ is a set of learnable model parameters. We denote policy used in such approximation as π_Θ.

Model Architecture. Our model architecture is inspired by DIRAC (**D**eep reinforcement learning for sp**I**n-glass g**R**ound-st**A**te **C**alculation), an Encoder-Decoder architecture introduced in [3]. It exploits the fact that the Ising spin-glass instance is wholly described by the underlying graph. In this view, couplings J_{ij} become edge weights, external magnetic field h_i and spin σ_i become node weights. Employing DIRAC is a two-step process. At first, it encodes the whole spin-glass instance such that every node is embedded into a low-dimensional vector, and then the decoder leverages those embeddings to calculate the Q-value of every possible action. Then, the agent chooses the action with the highest Q-value. In the next sections, We will describe those steps in detail.

Encoding. As described above, the Ising spin-glass instance can be described in the language of graph theory. It allows us to employ graph neural networks [4,5], which are neural networks designed to take graphs as inputs. We use modified SGNN (**S**pin **G**lass **N**eural **N**etwork) [3] to obtain node embedding. To capture the coupling strengths and external field strengths (J_{ij} and h_i), which are crucial to determining the spin glass ground states, SGNN performs two updates at each layer: the edge-centric update and the node-centric update, respectively.

Let's $z_{(i,j)}$ denote embedding of edge (i, j) and $z_{(i)}$ embedding of node i. The edge-centric update aggregates embedding vectors from its adjacent nodes (i.e. for edge (i, j) this update aggregate embeddings $z_{(i)}$ and $z_{(j)}$), and then concatenates it with self-embedding $z_{(i,j)}$. The vector obtained in this way is then subject to non-linear transformation (ex. $\mathrm{ReLU}(x) = \max(0, x)$). Mathematically, it can be described by the following equation

$$z^{k+1}_{(i,j)} = \mathrm{ReLU}(\gamma_\theta(z^k_{(i,j)}) \oplus \phi_\theta(z^k_{(i)} + z^k_{(j)})), \tag{2}$$

where $z^k_{(i,j)}$ denotes encoding of edge (i, j) obtained after k layers. Similarly, $z^k_{(i)}$ denotes encoding of node i obtained after k layers, γ_θ and ϕ_θ are some differentiable functions of θ. Symbol $\oplus$ is used to denote concatenation operation.

The node-centric update is defined in a similar fashion. It aggregates embedding of adjacent edges and then concatenates it with self-embedding $z_{(i)}$. Later, we transform this concatenated vector to obtain the final embedding. Using notation from equation (2), the final result is the following:

$$z^{k+1}_{(i)} = \mathrm{ReLU}(\phi_\theta(z^k_{(i)}) \oplus \gamma_\theta(\mathrm{E}^k_i)), \quad \mathrm{E}^k_i = \sum_j z^k_{(i,j)}. \tag{3}$$

Edge features are initialized as edge weights $\{J_{ij}\}$. It is not trivial to find adequate node features, as node weights $\{h_i\}$ and spins σ_i are not enough.

We also included pooling layers not presented in the original design. We reasoned that after concatenation, vectors start becoming quite big, so we employ

pooling layers to not only reduce the model size but also preserve the most essential parts of every vector. As every node is a potential candidate for action, we call the final encoding of node i its *action embedding* and denote it as Z_i. To represent the whole Chimera, we use *state embedding*, denoted as Z_s, which is the sum over all node embedding vectors, which is a straightforward but empirically effective way for graph-level encoding [9].

Decoding. Once all action embeddings Z_i and state embedding Z_s are computed in the encoding stage, the decoder will leverage these representations to compute an approximated state-action value function $Q(s, a; \Theta)$ which predicts the expected future rewards of taking action a in state s, and following the policy π_Θ till the end. Specifically, we concatenate the embeddings of state and action and use it as decoder input. In principle, any decoder architecture may be used. Here, we use a standard feed-forward neural network. Formally, the decoding process can be written as $Q(s, a^{(i)}; \Theta) = \psi_\Theta(Z_s \oplus Z_i)$, where ψ_Θ is a dense feed-forward neural network.

Training. We train our model on randomly generated Chimera instances. We found that the minimal viable size of the training instance is C_3. Smaller instances lack couplings between clusters, crucial in full Chimera, which leads to poor performance. We generate $\{J_{ij}\}$ and $\{h_i\}$ from a normal distribution $\mathcal{N}(0, 1)$ and starting spin configuration σ from a uniform distribution. To introduce low-energy instances, we employed the following pre-processing procedure. For each generated instance, with probability $p = 10\%$, we perform simulated annealing before passing the instance through SGNN.

Our goal is to learn approximation of optimal action-value function $Q(a, s; \Theta)$, so as the reinforcement learning algorithm we used standard n-step deep Q learning [13,14] with memory replay buffer. During the episode, we collect sequence of states action and rewards $\tau = (s_0, a_0, r_0, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ with terminal state as final element. From those we construct n-step transitions $\tau_t^n = (s_t, a_t, r_{t,t+n}, s_{t+n})$ which we collect in memory replay buffer $\mathcal{B}$. Here $r_{t,t+n} = \sum_{k=0}^{k=n} \gamma^k r_{t+k}$ is return after n-steps.

Simulated Annealing with Reinforcement. Simulated annealing with reinforcement (SAwR) combines machine learning and classical optimization algorithm. Simulated annealing (SA) takes its name from a process in metallurgy involving heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system energy. In SA, we start in some state s and in each step, we move to a randomly chosen neighbouring state s'. If a move lowers energy $E(s)$ of the system, we accept it. If it doesn't, we use the so-called the Metropolis-Hasting criterion: $\mathbb{P}(\text{accept } s' \mid s) = \min(1, e^{-\beta \Delta E})$, where $\Delta E = E(s') - E(s)$ and β denotes inverse temperature. In our case, the move is defined as a single-spin flip. Simulated annealing tends to accept all possible moves at high temperatures. However, it likely accepts only those moves that lower the energy at low temperatures.

Our idea is to reinforce random sampling with the trained model. As a result, instead of using the Metropolis-Hasting criterion, we perform a single pass of the DIRAC episode at low temperatures, as described in the caption of Fig. 1.

3 Experiments

We collected data from the D-Wave 2000Q device using default parameters by generating 500 random instances of sizes 128, 512, 1152, and 2048 spins. Parameters $\{J_{ij}\}$ and $\{h_i\}$ were generated from a normal distribution $\mathcal{N}(0, 1)$ and initial spin configuration, σ, from a uniform distribution. We used identical distributions for training instances. The low energy states of generated instances were obtained using quantum annealing. We have used three methods - simulated annealing, SAwR, and D-wave steepest descend post-processing [2], cf., Fig. 2.

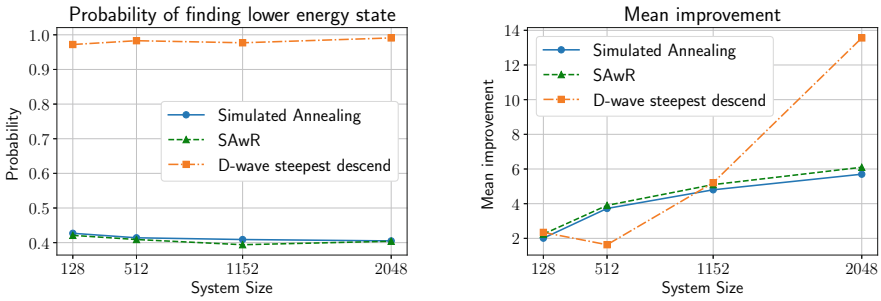

Fig. 2. Results of the experiments. SAwR is the simulated annealing with reinforcement. The probability of finding a lower energy state was computed over 500 random instances for each Chimera size. The value of the improvement was defined as the difference between the initial energy and the lowest energy found.

Two metrics were tested: the probability of finding lower energy states and the mean value of an improvement over the starting energy state. To compute the probability for each Chimera size, we started with proposed solutions obtained from quantum annealer and tried to lower them using different tested methods. Then we counted those instances for which a lower energy state was found. We define the value of the improvement as the difference between the initial energy and the lowest energy found.

SAwR achieved lower probabilities of finding a lower energy state. Although the difference between SAwR and traditional simulated annealing is slight, its consistency across all sizes suggests that it is systemic rather than random noise. It is interesting that, on average, SAwR was able to find a better low energy state than simulated annealing, but still, the difference is not significant.

Acknowledgments. TS acknowledges support from the National Science Centre (NCN), Poland, under SONATA BIS 10 project number 2020/38/E/ST3/ 00269.

This research was funded by National Science Centre, Poland (grant no 2020/39/B/ST6/01511 and 2018/31/N/ST6/02374) - TT, and Foundation for Polish Science (grant no POIR.04.04.00-00-14DE/ 18-00 carried out within the Team-Net program co-financed by the European Union under the European Regional Development Fund) - (LP, ZP, and BG).

References

1. Bryngelson, J.D., Wolynes, P.G.: Spin glasses and the statistical mechanics of protein folding. Proc. Natl. Acad. Sci. U.S.A. **84**(21), 7524–7528 (1987). https://doi.org/10.1073/pnas.84.21.7524
2. D-Wave Systems. Postprocessing. https://docs.dwavesys.com/docs/latest/c_qpu_pp.html. Accessed 8 Apr 2022
3. Fan, C., et al.: Finding spin glass ground states through deep reinforcement learning. arXiv preprint arXiv:2109.14411 (2021)
4. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272. PMLR (2017)
5. Hamilton, W.L.: Graph representation learning. Synthesis lectures on artificial intelligence and machine learning **14**(3), 1–159 (2020). https://doi.org/10.2200/S01045ED1V01Y202009AIM046
6. Jansen, S., Ruskai, M.B., Seiler, R.: Bounds for the adiabatic approximation with applications to quantum computation. J. Math. Phys. **48**(10), 102111 (2007). https://doi.org/10.1063/1.2798382
7. Kadowaki, T., Nishimori, H.: Quantum annealing in the transverse ising model. Phys. Rev. E **58**, 5355–5363 (1998). https://doi.org/10.1103/PhysRevE.58.5355
8. Kato, T.: On the adiabatic theorem of quantum mechanics. J. Phys. Soc. Jpn **5**(6), 435–439 (1950). https://doi.org/10.1143/JPSJ.5.435
9. Khalil, E., Dai, H., Zhang, Y., Dilkina, B., Song, L.: Combinatorial optimization algorithms over graphs. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017). https://proceedings.neurips.cc/paper/2017/file/d9896106ca98d3d05b8cbdf4fd8b13a1-Paper.pdfLearning
10. Kirkpatrick, S., Toulouse, G.: Configuration space analysis of travelling salesman problems. J. Phys. **46**(8), 1277–1292 (1985). https://doi.org/10.1051/jphys:019850046080127700
11. Lucas, A.: Ising formulations of many NP problems. Front. Phys. **2**, 5 (2014). https://doi.org/10.3389/fphy.2014.00005
12. McGeoch, C.C.: Adiabatic quantum computation and quantum annealing: theory and practice. Synth. Lect. Quant. Comput. **5**(2), 1–93 (2014)
13. Mnih, V., et al.: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
14. Peng, J., Williams, R.J.: Incremental multi-step q-learning. In: Machine Learning Proceedings, pp. 226–232. Elsevier (1994). https://doi.org/10.1016/B978-1-55860-335-6.50035-0
15. Rams, M.M., Mohseni, M., Eppens, D., Jałowiecki, K., Gardas, B.: Approximate optimization, sampling, and spin-glass droplet discovery with tensor networks. Phys. Rev. E **104**, 025308 (2021). https://doi.org/10.1103/PhysRevE.104.025308

16. Rosenberg, G., Haghnegahdar, P., Goddard, P., Carr, P., Wu, K., De Prado, M.L.: Solving the optimal trading trajectory problem using a quantum annealer. IEEE J. Select. Top. Signal Process. **10**(6), 1053–1060 (2016). https://doi.org/10.1109/JSTSP.2016.2574703
17. Santoro, G.E., Martoňák, R., Tosatti, E., Car, R.: Theory of quantum annealing of an Ising spin glass. Science **295**(5564), 2427–2430 (2002). https://doi.org/10.1126/science.1068774
18. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, New York (2018). https://doi.org/10.1007/978-1-4615-3618-5