# An Application of Reinforcement Learning for Minor Embedding in Quantum Annealing

Riccardo Nembrini[1,*], Maurizio Ferrari Dacrema[1] and Paolo Cremonesi[1]

[1]Politecnico di Milano, Milano, Italy

**Abstract**

Research in the Quantum Computing (QC) field has been soaring thanks to the latest developments and wider availability of real hardware. The strong interest in this technology has naturally spurred a contamination with the Machine Learning (ML) field. Both quantum methods to perform ML and ML methods to support quantum computation has been developed. A largely diffused QC paradigm is that of Quantum Annealers, machines that can rapidly search for solutions to optimization problems. Their sparse qubit structure, however, requires to search for a mapping between the problem's and the hardware's graphs before computation. This is a NP-hard combinatorial optimization task in itself, called Minor Embedding. In this work, we aim at developing and assessing the capabilities of Reinforcement Learning to perform this task.

**Keywords**

Quantum Computing, Quantum Annealing, Reinforcement Learning, Proximal Policy Optimization

## 1. Introduction

Development on Quantum Computing has soared in recent years, thanks to core technology advancements and wider hardware availability. Different paradigms for such technology exist. In this work we focus on Quantum Annealing (QA), which exploits quantum mechanics to search for solutions to optimization problems. Despite being prone to errors, a quantum annealer can rapidly find low energy solutions by means of sampling. However, there are some restrictions on the problems an annealer can solve. In order to solve a problem with QA, one needs to map the problem's structure to the QA hardware's sparse topology (an example is shown in Figure 1). This is a NP-hard task called Minor Embedding (ME), mainly solved through heuristics [1–3]. A single variable from the problem may correspond to more than one qubit, forming a so-called *chain*, and thus an entire problem may require many more qubits than its variables. Performing ME is also computationally expensive and generally much slower than the quantum processes used by QA to solve a problem. This difference becomes of few orders of magnitudes as both the problem's and the quantum architecture's sizes grow [4]. Furthermore, a poor minor embedding can negatively affect the quality of the solutions QA finds.

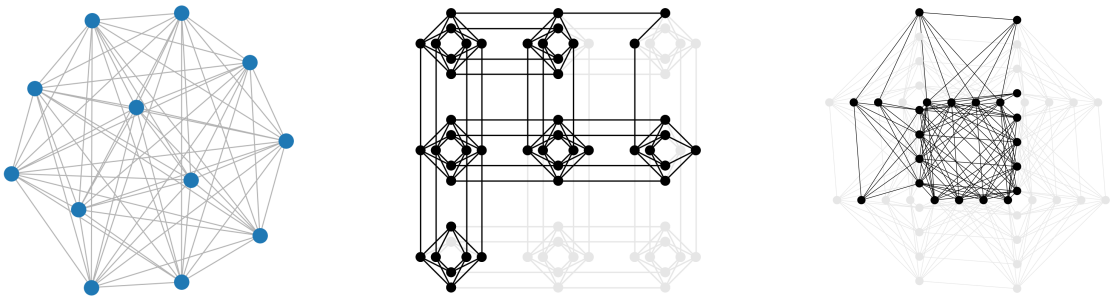
In this work, we propose a new approach to solve the ME problem, based on Reinforcement Learning (RL), in order to provide researchers with a new alternative path w.r.t. heuristics. RL is based on the interaction, at discrete time steps in our case, between an *agent* and an *environment*. The agent *observes* the environment's *state* and performs an *action* according to a certain *policy*. The environment reacts to such action by (possibly) changing its state and giving the agent a *reward*, which should suggest the quality of the action in that particular state. The objective is thus to train the agent to obtain the highest cumulative reward. In our case, we train an agent to produce the ME of a problem on a specific hardware graph from scratch, without prior knowledge. The choice of RL was dictated by the combinatorial nature of minor embedding. Indeed, training a supervised method, for example, would be impractical, since the solution space is too vast. Moreover, RL has already been successfully applied to both Combinatorial Optimization [5–7] and other Quantum Computing related tasks [8–10]. Given

*AIQ x QIA 2024: 2nd International Workshop on AI for Quantum and Quantum for AI, November 25th - 28th 2024, Bolzano, Italy*

*Corresponding author.

riccardo.nembrini@polimi.it (R. Nembrini); maurizio.ferrari@polimi.it (M. Ferrari Dacrema); paolo.cremonesi@polimi.it (P. Cremonesi)

0000-0002-1915-6107 (R. Nembrini); 0000-0001-7103-2788 (M. Ferrari Dacrema); 0000-0002-1253-8081 (P. Cremonesi)



(a) Starting graph (12 nodes) (b) Chimera (47 nodes) (c) Zephyr (22 nodes)

**Figure 1:** Example of minor embeddings of a fully-connected graph of 12 nodes on different QA hardware topologies. Highlighted in 1b and 1c are only the nodes and edges that are part of the mappings.

such applications, we explore the possibility of using RL for ME, by testing the capabilities of an agent trained with the state-of-the-art Proximal Policy Optimization algorithm [11].

## 2. Reinforcement Learning for Minor Embedding

In this section, we describe the components of our agent, called RLME, such as the environment and its state, the possible actions and the reward function. First of all, the objective of the agent is to perform ME mapping a *problem graph* G to a *hardware graph* H. The interaction loop between agent and environment involves mapping one node from G to a node in H at each step. The G node to map is chosen in a round-robin fashion, while the H node is chosen by the agent's policy from the set of selectable nodes, comprised of qubits not yet assigned to a problem variable that are adjacent to qubits already mapped to the same variable (its chain, if present). Therefore, the environment's state includes information about both G and H. An observation of the state , obtained by the agent, is a 1-dimensional array composed of contiguous sections representing different aspects of the state. In a section, each cell corresponds to a single node in G or H, with a predefined mapping consistent among all sections referring to the same graph. After selecting the round-robin G node, the observation's sections are the following:

- a one-hot encoding indicating which is the current round-robin G node,
- one component for each existing qubit, with value 1 if the qubit is part of the current round-robin node's chain, 0 otherwise,
- one component for each existing qubit, with value 1 if the qubit is selectable (not yet mapped and adjacent to the chain of the round-robin node, if present), 0 otherwise,
- for each G node, the number of connections with other G nodes that are missing in the mapping.

In summary, given an intermediate state of the ME process, the agent would be aware of the next G node for which to map a new qubit and its current chain in the mapping, the selectable qubits and the number of missing connections between chains for the mapping to be valid.

The action performed by the agent is the choice of one of the selectable qubits. After the action, which is determined by a policy on the observation of the state, the agent receives a reward from the environment. Depending on the objectives of an agent, one could design different kinds of rewards. In this work, the focus is on obtaining the shortest possible chains, therefore the rewards corresponding to each action are fixed and negative. Thus, maximizing the cumulative reward would teach the agent how to build minor embeddings with fewer nodes. Agent training is performed using the Proximal Policy Optimization RL algorithm [11], which learns the policy with Deep Neural Networks. In order to rule out non-selectable qubits from the possible actions we also use Invalid Action Masking [12].

## 3. Experimental Protocol

Experiments with the RLME agent are performed in two different scenarios. In the first one, our goal is to understand how to build the environment and how the agent would scale in its training process with the sizes of G and H. Therefore, we train multiple agents, one for each couple of specific G and H graphs. All the considered G graphs are fully connected and vary in the number of nodes. H graphs, instead, vary in topology (Chimera and Zephyr [13], shown in Figure 1) and number of nodes. Each agent learns how to perform ME of a certain G graph with |G| nodes on a certain H graph, with a budget of 1 million training actions.

In the second scenario, instead, our goal is to understand if RLME is able to generalize on unseen data and if learning on smaller graphs first can help when scaling. Every agent is trained to perform ME of a synthetic dataset of varying G graphs, with different sizes and connectivity, on a specific H graph. The dataset is built by generating all the possible non-isomorphic graphs with sizes between 3 and 7 nodes, splitting into training and testing sets with respectively 80% and 20% of the graphs, trying to maintain a uniform distribution on the number of edges. Then, in order to have around 1000 graphs for each G size, we duplicate (if there are not enough graphs) or sample (if there are more than required) the corresponding graphs, again keeping a uniform distribution for the edges. During training, performed with a budget of 3 million actions, the agent sees the graphs ordered according to the size of G, so that it learns from simpler graphs first. When the dataset has been completely fed to the agent, it is shuffled (maintaining the size ordering) and re-submitted to the agent.

RL agents are trained with Stable Baslines3 [14], with default hyperparameters. In both scenarios, each agent is trained 10 times with different random seeds and the testing results are averaged between all trained models. In the first scenario, we use each trained agent to generate 100 mappings for their respective fixed G-H couple. In the second, we use each agent to perform ME of all graphs in the testing set on their respective H graphs. We evaluate both scenarios based on how many of the generated mappings are valid and on the number of qubits required. For both scenarios, we compare RLME results with the general heuristic developed in [1]. To replicate our agents' behavior, each time we use the heuristic for the ME process we generate 100 mappings.

## 4. Results and Discussion

Table 1 shows results for the first scenario described in Section 3, when training the agent on fully connected problem graphs and performing ME on the Zephyr topology (Figure 1c), compared with the heuristic. We performed experiments on G graphs with 3 to 7 nodes and H graphs from 160 to 2176 nodes (qubits). Notice that in all results we refer to H's size as $H_{size}$ and the number of qubits can be computed as $16 * H_{size} * (2 * H_{size} + 1)$ for Zephyr and $8 * H_{size}^2$ for Chimera. Only a slice of the results is reported, for clarity's sake, since other results show similar behaviors. Let's remark that, for problems of this size, the heuristic can find optimal solutions, with the lowest possible number of qubits.

As it can be seen, with a lower $H_{size}$, the agent is able to precisely learn how to map nodes from the fully-connected problem, with a number of qubits comparable to that of the heuristic. With a larger $H_{size}$, the number of actions that can be chosen by the agent is higher, therefore training becomes harder, with the agent struggling to find solutions as compact as with a lower $H_{size}$. This behavior is also found when comparing RLME used on the Zephyr and Chimera topology. Indeed, while in Zephyr graphs each qubit is connnected to at most other 20 qubits, in Chimera graphs the maximum degree is 6. Because of the sparser topology, it is harder for the agent to navigate the H graph and find the needed connections. Figure 2 shows the comparison between RLME trained and used on Chimera and Zephyr topologies, when the number of qubits in H increases. When scaling, the number of required qubits in the mapping is drastically lower on Zephyr, while the majority of the agents on Chimera cannot find a valid mapping. This kind of challenge is not present in the heuristic, since it chooses new nodes to add to the mapping based on shortest-path distances, which are not influenced by the size of H, if not for algorithmic complexity.

| $H_{size}$ | Method | \|G\| = 3 | | \|G\| = 4 | | \|G\| = 5 | | \|G\| = 6 | | \|G\| = 7 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SR% | #Q | SR% | #Q | SR% | #Q | SR% | #Q | SR% | #Q |
| 2 | RL | 100 | $3 \pm 0$ | 100 | $4 \pm 0$ | 100 | $6 \pm 1$ | 100 | $9 \pm 1$ | 100 | $11 \pm 3$ |
| | Heuristic | 100 | $3 \pm 0$ | 100 | $4 \pm 1$ | 100 | $6 \pm 0$ | 100 | $8 \pm 0$ | 100 | $10 \pm 0$ |
| 5 | RL | 100 | $5 \pm 2$ | 100 | $10 \pm 5$ | 100 | $16 \pm 5$ | 100 | $26 \pm 8$ | 100 | $39 \pm 18$ |
| | Heuristic | 100 | $3 \pm 0$ | 100 | $4 \pm 0$ | 100 | $6 \pm 0$ | 100 | $8 \pm 0$ | 100 | $10 \pm 0$ |
| 8 | RL | 100 | $6 \pm 3$ | 100 | $21 \pm 24$ | 100 | $34 \pm 39$ | 100 | $172 \pm 159$ | 100 | $490 \pm 188$ |
| | Heuristic | 100 | $3 \pm 0$ | 100 | $4 \pm 0$ | 100 | $6 \pm 0$ | 100 | $8 \pm 0$ | 100 | $10 \pm 0$ |

**Table 1**
Results obtained from the RL agent when trained on fully-connected G graphs for different Zephyr H sizes, compared with the heuristic. SR% is the success rate when using the corresponding method to perform ME. #Q is the number of qubits used in the solution.

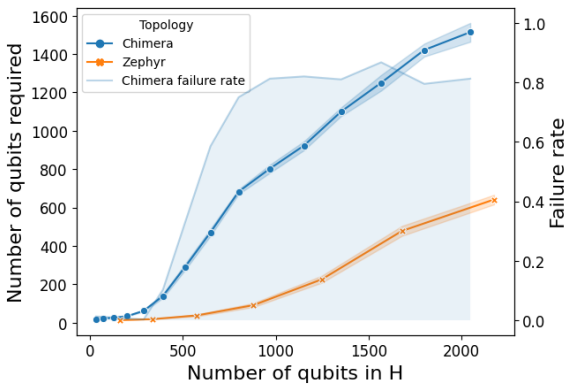



**Figure 2:** Number of qubits required by the RLME agent in the first scenario for a fully-connected G graph with 7 nodes on Chimera and Zephyr graphs. On the X-axis there is the number of qubits of the H graph. On the left Y-axis there is the number of qubits required by the mapping found with RLME. On the right Y-axis, represented by the filled blue area, there is the failure rate of the agent on Chimera graphs.

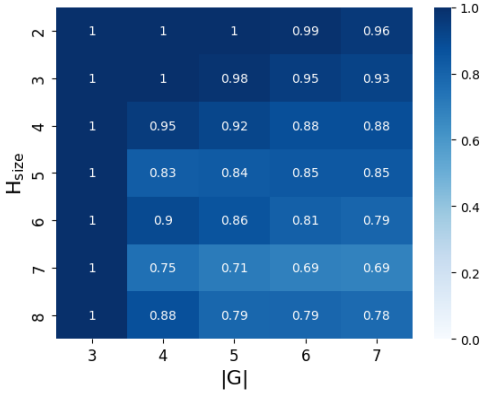



**Figure 3:** Comparison between the number of qubits required by the heuristic w.r.t. to RLME when tested on the testing graphs in the second scenario. The minimum number of qubits used by the heuristic for each testing graph is divided by the minimum used by the agent. Then, these ratios are averaged among the same |G|, to obtain the values shown.

When training the agents on the second scenario with the dataset, instead, the size of H does not affect the results in the same way. Figure 3 shows the comparison between the number of qubits required by the heuristic when performing ME on the training set w.r.t. RLME. As it can be seen, even with $H_{size} = 8$, the agent is able to obtain mappings with a number of qubits comparable to the heuristic (around 2 qubits more for |G| = 7). This suggests that the agent is learning better when seeing smaller graphs first, exploiting the experience made in navigating the H graph when mapping simpler G graphs.

## 5. Conclusions and Future Directions

In this work we develop a Reinforcement Learning agent capable of performing Minor Embedding, a key task when using a Quantum Annealer. We describe the components needed to train the agent and report the results obtained in two different scenarios. From these results we conclude that the RL agent is able to generate valid mappings in both scenarios, obtaining the best results when the training phase is performed with different graphs, starting from simpler ones. Future directions comprise designing and testing new reward functions and additional information to be fed to the agent, such as distances between nodes or qubit chains. An extension making use of Graph Neural Networks to extract better information directly from the graphs is already in the works.

## Acknowledgments

We acknowledge the financial support from ICSC - "National Research Centre in High Performance Computing, Big Data and Quantum Computing", funded by European Union – NextGenerationEU. We acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support.

## References

[1] J. Cai, W. G. Macready, A. Roy, A practical heuristic for finding graph minors, CoRR abs/1406.2741 (2014). URL: http://arxiv.org/abs/1406.2741. arXiv:1406.2741.

[2] T. Boothby, A. D. King, A. Roy, Fast clique minor generation in chimera qubit connectivity graphs, Quantum Inf. Process. 15 (2016) 495–508. URL: https://doi.org/10.1007/s11128-015-1150-6. doi:10.1007/S11128-015-1150-6.

[3] Y. Sugie, Y. Yoshida, N. Mertig, T. Takemoto, H. Teramoto, A. Nakamura, I. Takigawa, S. Minato, M. Yamaoka, T. Komatsuzaki, Minor-embedding heuristics for large-scale annealing processors with sparse hardware graphs of up to 102, 400 nodes, Soft Comput. 25 (2021) 1731–1749. URL: https://doi.org/10.1007/s00500-020-05502-6. doi:10.1007/S00500-020-05502-6.

[4] M. Ferrari Dacrema, F. Moroni, R. Nembrini, N. Ferro, G. Faggioli, P. Cremonesi, Towards feature selection for ranking and classification exploiting quantum annealers, in: E. Amigó, P. Castells, J. Gonzalo, B. Carterette, J. S. Culpepper, G. Kazai (Eds.), SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022, ACM, 2022, pp. 2814–2824. URL: https://doi.org/10.1145/3477495.3531755. doi:10.1145/3477495.3531755.

[5] I. Bello, H. Pham, Q. V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings, OpenReview.net, 2017. URL: https://openreview.net/forum?id=Bk9mxlSFx.

[6] N. Mazyavkina, S. Sviridov, S. Ivanov, E. Burnaev, Reinforcement learning for combinatorial optimization: A survey, Comput. Oper. Res. 134 (2021) 105400. URL: https://doi.org/10.1016/j.cor.2021.105400. doi:10.1016/J.COR.2021.105400.

[7] F. Berto, C. Hua, J. Park, M. Kim, H. Kim, J. Son, H. Kim, J. Kim, J. Park, RL4CO: an extensive reinforcement learning for combinatorial optimization benchmark, CoRR abs/2306.17100 (2023). URL: https://doi.org/10.48550/arXiv.2306.17100. doi:10.48550/ARXIV.2306.17100. arXiv:2306.17100.

[8] L. Moro, M. G. A. Paris, M. Restelli, E. Prati, Quantum compiling by deep reinforcement learning, Communications Physics 4 (2021). URL: http://dx.doi.org/10.1038/s42005-021-00684-3. doi:10.1038/s42005-021-00684-3.

[9] Z. T. Wang, Q. Chen, Y. Du, Z. H. Yang, X. Cai, K. Huang, J. Zhang, K. Xu, J. Du, Y. Li, Y. Jiao, X. Wu, W. Liu, X. Lu, H. Xu, Y. Jin, R. Wang, H. Yu, S. P. Zhao, Quantum compiling with reinforcement learning on a superconducting processor, CoRR abs/2406.12195 (2024). URL: https://doi.org/10.48550/arXiv.2406.12195. doi:10.48550/ARXIV.2406.12195. arXiv:2406.12195.

[10] S. Foderà, G. Turati, R. Nembrini, M. F. Dacrema, P. Cremonesi, Reinforcement learning for variational quantum circuit design, in: Proceedings of the International Workshop on AI for Quantum and Quantum for AI (AIQxQIA 2024) co-located with the 23rd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, CoRR abs/1707.06347 (2017). URL: http://arxiv.org/abs/1707.06347. arXiv:1707.06347.

[12] S. Huang, S. Ontañón, A closer look at invalid action masking in policy gradient algorithms, in: R. Barták, F. Keshtkar, M. Franklin (Eds.), Proceedings of the Thirty-Fifth International Florida

Artificial Intelligence Research Society Conference, FLAIRS 2022, Hutchinson Island, Jensen Beach, Florida, USA, May 15-18, 2022, 2022. URL: https://doi.org/10.32473/flairs.v35i.130584. doi:10.32473/flairs.v35i.130584.

[13] K. Boothby, A. King, J. Raymond, Zephyr topology of d-wave quantum processors, Tech. rep., D-Wave Systems Inc. (2021). URL: https://www.dwavesys.com/media/2uznec4s/14-1056a-a_zephyr_topology_of_d-wave_quantum_processors.pdf.

[14] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Stable-baselines3: Reliable reinforcement learning implementations, Journal of Machine Learning Research 22 (2021) 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.