Reinforcement-Learning-Based Variational Quantum Circuits Optimization for Combinatorial Problems

Sami Khairy
Illinois Institute of Technology
skhairy@hawk.iit.edu

Ruslan Shaydulin
Clemson University
rshaydu@g.clemson.edu

Lukasz Cincio
Los Alamos National Laboratory
lcincio@lanl.gov

Yuri Alexeev
Argonne National Laboratory
yuri@alcf.anl.gov

Prasanna Balaprakash
Argonne National Laboratory
pbalapra@anl.gov

Abstract

Quantum computing exploits basic quantum phenomena such as state superposition and entanglement to perform computations. The Quantum Approximate Optimization Algorithm (QAOA) is arguably one of the leading quantum algorithms that can outperform classical state-of-the-art methods in the near term. QAOA is a hybrid quantum-classical algorithm that combines a parameterized quantum state evolution with a classical optimization routine to approximately solve combinatorial problems. The quality of the solution obtained by QAOA within a fixed budget of calls to the quantum computer depends on the performance of the classical optimization routine used to optimize the variational parameters. In this work, we propose an approach based on reinforcement learning (RL) to train a policy network that can be used to quickly find high-quality variational parameters for unseen combinatorial problem instances. The RL agent is trained on small problem instances which can be simulated on a classical computer, yet the learned RL policy is generalizable and can be used to efficiently solve larger instances. Extensive simulations using the IBM Qiskit Aer quantum circuit simulator demonstrate that our trained RL policy can reduce the optimality gap by a factor up to 8.61 compared with other off-the-shelf optimizers tested.

1 Introduction

Currently available Noisy Intermediate-Scale Quantum (NISQ) computers have limited error-correction mechanisms and operate on a small number of quantum bits (qubits). Leveraging NISQ devices to demonstrate quantum advantage in the near term requires quantum algorithms that can run using low-depth quantum circuits. The Quantum Approximate Optimization Algorithm (QAOA) [1], which has been recently proposed for approximately solving combinatorial problems, is considered one of the candidate quantum algorithms that can outperform classical state-of-the-art methods in the NISQ era [2]. QAOA combines a parameterized quantum state evolution on a NISQ device with a classical optimization routine to find optimal parameters. Achieving practical quantum advantage using QAOA is therefore contingent on the performance of the classical optimization routine.

QAOA encodes the solution to a classical unconstrained binary assignment combinatorial problem in the spectrum of a cost Hamiltonian H_C by mapping classical binary variables $s_i \in \{-1, 1\}$ onto the eigenvalues of the quantum Pauli-Z operator $\hat{\sigma}^z$. The optimal solution to the original combinatorial problem can therefore be found by preparing the highest energy eigenstate of H_C. To this end, QAOA constructs a variational quantum state $|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle$ by evolving a uniform superposition quantum state

Second Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019), Vancouver, Canada.

$|\psi\rangle = |+\rangle^{\otimes n}$ using a series of alternating operators $e^{-i\beta_k H_M}$ and $e^{-i\gamma_k H_C}$, $\forall k \in [p]$,

$$|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_C} \cdots e^{-i\beta_1 H_M} e^{-i\gamma_1 H_C} |+\rangle^{\otimes n}, \tag{1}$$

where $\boldsymbol{\beta}, \boldsymbol{\gamma} \in [-\pi, \pi]$ are $2p$ variational parameters, n is the number of qubits or binary variables, and H_M is the transverse field mixer Hamiltonian $H_M = \sum_i \hat{\sigma}_i^x$. In order to prepare the highest energy eigenstate of H_C, a classical optimizer is used to maximize the expected energy of H_C,

$$f(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \langle \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | H_C | \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) \rangle . \tag{2}$$

For $p \to \infty$, $\exists \boldsymbol{\beta}_*, \boldsymbol{\gamma}_* = \arg\max_{\boldsymbol{\beta}, \boldsymbol{\gamma}} f(\boldsymbol{\beta}, \boldsymbol{\gamma})$ such that the resulting quantum state $|\psi(\boldsymbol{\beta}_*, \boldsymbol{\gamma}_*)\rangle$ encodes the optimal solution to the classical combinatorial problem [1]. QAOA has been applied to a variety of problems, including network community detection [3, 4], portfolio optimization [5], and graph maximum cut (Max-Cut) [6, 7]. In this work, we choose graph Max-Cut as a target problem for QAOA because of its equivalence to quadratic unconstrained binary optimization.

Consider a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. The goal of Max-Cut is to partition the set of vertices V into two disjoint subsets such that the total weight of edges separating the two subsets is maximized:

$$\max_{\mathbf{s}} \sum_{i,j \in V} w_{ij} s_i s_j + c, \qquad s_k \in \{-1, 1\}, \forall k, \tag{3}$$

where s_k is a binary variable that denotes partition assignment of vertex k, $\forall k \in [n]$, $w_{ij} = 1$ if $(i, j) \in E$, and 0 otherwise, and c is a constant. In order to encode (3) in a cost Hamiltonian, binary variables s_k are mapped onto the eigenvalues of the Pauli-Z operator $\hat{\sigma}^z$,

$$H_C = \sum_{i,j \in V} w_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z. \tag{4}$$

The works of [8, 9, 6] show that QAOA for Max-Cut can achieve approximation ratios exceeding those achieved by the classical Goemans-Williamson algorithm [10]. However, QAOA parameter optimization is known to be a hard problem because (2) is nonconvex with low-quality nondegenerate local optima for high p [11, 7]. Existing works explore many approaches to QAOA parameter optimization, including a variety of off-the-shelf gradient-based [12, 7, 6] and derivative-free methods [13, 14, 11]. Noting that the optimization objective (2) is specific to a given combinatorial instance through its cost Hamiltonian (4), researchers have approached the task of finding optimal QAOA parameters as an instance-specific task. To the best of our knowledge, approaching QAOA parameter optimization as a learning task is underexplored, with few recent works [15].

Thus motivated, in this work we propose a method based on reinforcement learning (RL) to train a policy network that can learn to exploit geometrical regularities in the QAOA optimization objective, in order to efficiently optimize new QAOA circuits of unseen test instances. The RL agent is trained on a small Max-Cut combinatorial instance that can be simulated on a classical computer, yet the learned RL policy is generalizable and can be used to efficiently solve larger instances from different classes and distributions. By conducting extensive simulations using the IBM Qiskit Aer quantum circuit simulator, we show that our trained RL policy can reduce the optimality gap by a factor of up to 8.61 compared with commonly used off-the-shelf optimizers.

2 Proposed approach

Learning an ptimizer to train machine learning models has recently attracted considerable research interest. The motivation is to design optimization algorithms that can exploit structure within a class of problems, which is otherwise unexploited by hand-engineered off-the-shelf optimizers. In existing works, the leraned optimizer is implemented by a long short-term memory network [16] or a policy network of an RL agent [17]. Our proposed approach to QAOA optimizer learning departs from that of [17] in the design of the reward function and the policy search mechanism.

In the RL framework, an autonomous agent learns how to map its state $s \in \mathcal{S}$, to an action $a \in \mathcal{A}$, by repeated interaction with an environment. The environment provides the agent with a reward signal $r \in \mathbb{R}$, in response to its action. A solution to the RL task is a stationary Markov policy that maps the agent's states to actions, $\pi(a|s)$, such that the expected total discounted reward is maximized [18]. Learning a QAOA optimizer can therefore be regarded as learning a policy that produces iterative QAOA parameter updates, based on the following state-action-reward formulation,

1. $\forall s_t \in \mathcal{S}$, $s_t = \{\Delta f_{tl}, \Delta \boldsymbol{\beta}_{tl}, \Delta \boldsymbol{\gamma}_{tl}\}_{l=t-1,\ldots,t-L}$; in other words, the state space is the set of finite differences in the QAOA objective and the variational parameters between the current iteration and L history iterations, $\mathcal{S} \subset \mathbb{R}^{(2p+1)L}$.
2. $\forall a_t \in \mathcal{A}$, $a_t = \{\Delta \boldsymbol{\beta}_{tl}, \Delta \boldsymbol{\gamma}_{tl}\}_{l=t-1}$; in other words, the action space is the set of step vectors used to update the variational parameters, $\mathcal{A} \subset \mathbb{R}^{2p}$.
3. $\mathcal{R}(s_t, a_t, s_{t+1}) = f(\boldsymbol{\beta}_t + \Delta \boldsymbol{\beta}_{t,tl}, \boldsymbol{\gamma}_t + \Delta \boldsymbol{\gamma}_{tl}) - f(\boldsymbol{\beta}_t, \boldsymbol{\gamma}_t), l = t-1$; in other words, the reward is the change in the QAOA objective between two consecutive iterations.

The motivation for our state space formulation comes from the fact that parameter updates at $\mathbf{x}_t = (\boldsymbol{\beta}_t, \boldsymbol{\gamma}_t)$ should be in the direction of the gradient at $\mathbf{x}_t$ and the step size should be proportional to the Hessian at $\mathbf{x}_t$, both of which can be numerically approximated by using the method of finite differences. The RL agent's role is then to find an optimal reward-maximizing mapping to produce the step vector $a_t = \{\Delta \boldsymbol{\beta}_{tl}, \Delta \boldsymbol{\gamma}_{tl}\}_{l=t-1}$, given some collection of historical differences in the objective and parameters space, $\{\Delta f_{tl}, \Delta \boldsymbol{\beta}_{tl}, \Delta \boldsymbol{\gamma}_{tl}\}_{l=t-1,\ldots,t-L}$. Note that the cumulative rewards are maximized when $\mathcal{R}(s_t, a_t, s_{t+1}) \geq 0$, which means the QAOA objective has been increased between any two consecutive iterates. The reward function adheres to the Markovian assumption and encourages the agent to produce parameter updates that yield higher increase in the QAOA objective (2), if possible, while maintaining conditional independence on historical states and actions.

3 Experiments

We train our proposed approach on a small $n_R = 8$ qubit graph instance drawn from an Erdos Renyi random graph with edge generation probability $e_p = 0.5$. Training was performed over 750 epochs, where each epoch corresponds to $8,192$ QAOA circuit simulations executed by using IBM Qiskit Aer simulator [19]. Within each epoch are $1,28$ episodes. Each training episode corresponds to a trajectory of length $T = 64$ that is sampled from a depth-p QAOA objective (2) for the training instance. At the end of each episode, the trajectory is cut off and is randomly restarted.

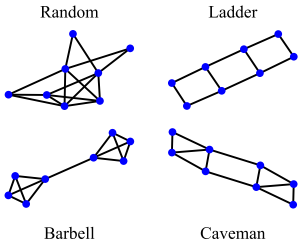


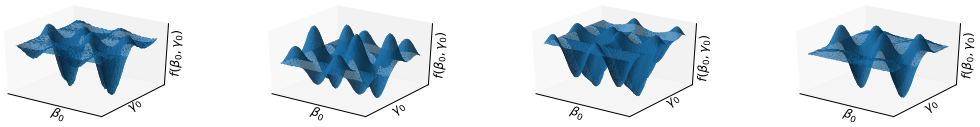
Figure 1: Sample graph instances, TBLR: $G_R(n_R = 8, e_p = 0.5)$, $G_L(n_L = 4)$, $G_B(n_B = 4)$, $G_C(n_C = 2, n_k = 4)$.

For policy search, we adopt the actor-critic Proximal Policy Optimization (PPO) algorithm [20], which uses a clipped surrogate advantage objective as a training objective. To further mitigate policy updates that can cause policy collapse, we adopt a simple early stopping method, which terminates gradient optimization on the PPO objective when the mean KL-divergence between the new and old policy hits a predefined threshold. Fully connected multilayer perceptron networks with two 64-neuron hidden layers for both the actor and critic networks are used. A Gaussian policy with a constant noise variance of e^{-6} is adopted throughout training. At testing, the trained policy network corresponding to the mean of the learned Gaussian policy is used, without noise.

To test the performance of the learned RL policy, we chose a large set G_{Test} of Max-Cut test instances, coming from different sizes, classes, and distributions. Specifically, four classes of graphs are considered: (1) Erdos Renyi random graphs $G_R(n_R, e_p)$, $n_R \in \{8, 12, 16, 20\}$, $e_p \in \{0.5, 0.6, 0.7, 0.8\}$, seed $= \{1, 2, 3, 4\}$; (2) ladder graphs $G_L(n_L)$, where $n_L \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ is the length of the ladder; (3) barbell graphs $G_B(n_B)$, formed by connecting two complete graphs K_{n_B} by an edge, where $n_B \in \{3, 4, 5, 6, 7, 8, 9, 10, 11\}$; and (4) caveman graphs $G_C(n_C, n_k)$, where n_C is the number of cliques and n_k is the size of each clique, $\{(n_C, 4) : n_C \in \{3, 4, 5\}\}$, $\{(n_C, 3) : n_C \in \{3, 5, 7\}\}$,$\{(2, n_K) : n_K \in \{3, 4, 5, 6, 7, 8, 9, 10\}\}$. Thus, $|G_{\text{Test}}| = 97$. Figure 1 shows sample graph instances in G_{Test}. G_{Test} is chosen to demonstrate that combining our proposed RL-based approach with QAOA can be a powerful tool for amortizing the QAOA optimization cost across graph instances, as well as demonstrating the generalizability of the learned policy.

4 Results

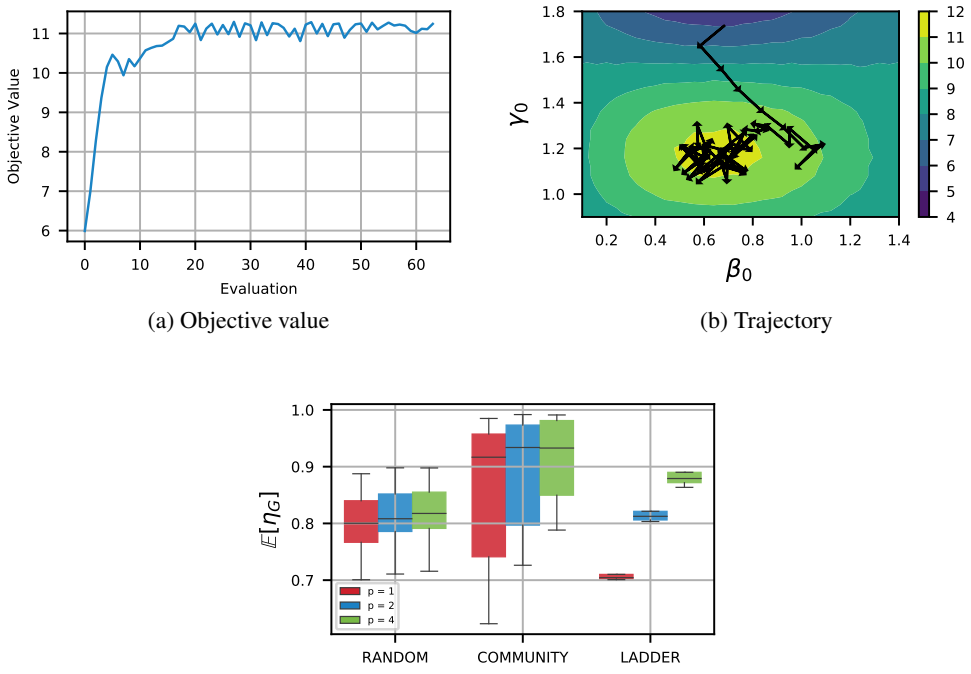
In Figure 2, the expected energy in (2) for a depth $(p = 1)$ QAOA circuit is shown for some graph instances. We observe that the expected energy is nonconvex in the parameter space. Figures 3(a) and



(a) $G_R(16, 0.5)$ (b) $G_L(8)$ (c) $G_B(8)$ (d) $G_C(2, 8)$

Figure 2: QAOA energy landscapes for $p = 1$.

(b) visualize the trajectory produced by the learned RL policy on one of the test instances. We can see that the learned policy produces trajectories that quickly head to the maximum (in about 20 iterations in this example), yet a wiggly behavior is observed afterwards. Figure 3(c) shows a boxplot of the expected approximation ratio performance, $\mathbb{E}[\eta_G] = \mathbb{E}[f/C_{\text{opt}}]$, of QAOA with respect to the classical optimal C_{opt} found by using brute-force methods across different graph instances in G_{Test}, which are grouped in three subgroups: (1) random graphs, which contains all graphs of the form $G_R(n_R, e_p)$; (2) community graphs, which contains graphs of the form $G_C(n_C, n_k)$ and $G_B(n_B)$; and (3) ladder graphs, which contains graphs of the form $G_L(n_L)$. We can see that increasing the depth of the QAOA circuit improves the attained approximation ratio. Next, we benchmark the performance of



(a) Objective value

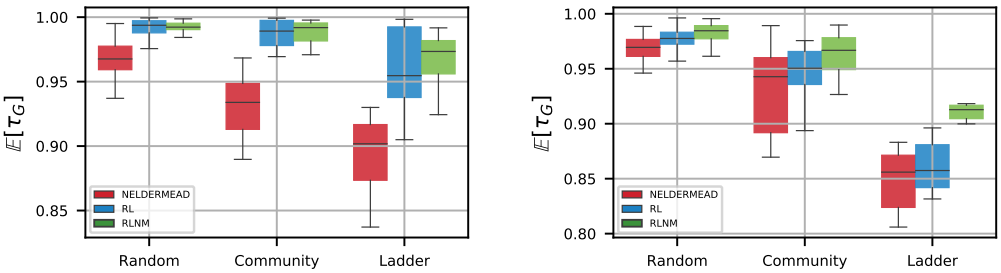
(b) Trajectory

(c) QAOA Approximation Ratio

Figure 3: Visualization of the trajectory produced by the learned RL policy on one of the test instances for the QAOA circuit of $p = 1$ (a)–(b), and the approximation ratio performance of QAOA with respect to classical optimal on graph instances in G_{Test} (c).

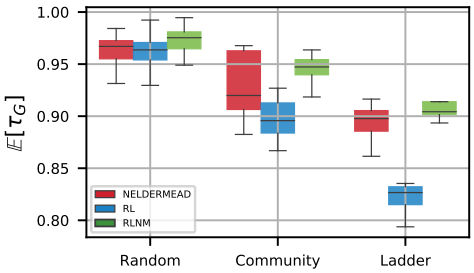
our trained RL-based QAOA optimization policy (referred to as RL) by comparing its performance with that of a commonly used derivative-free off-the-shelf optimizer, namely, Nelder-Mead [21], on graph instances in G_{Test}. Starting from 10 randomly chosen variational parameters in the domain of (2), each optimizer is given 10 attempts with a budget of $B = 192$ quantum circuit evaluations. In addition, we use the learned RL policy to generate trajectories of length $B/2$, and we resume the trajectory from the best parameters found using Nelder-Mead for the rest of $B/2$ evaluations (referred to as RLNM). This approach is motivated by the observed behavior of the learned RL policy in 3(b).

In Figures 4 (a)–(c), we present a boxplot of the expected optimality ratio, $\mathbb{E}[\tau_G] = \mathbb{E}[f/f_{\text{opt}}]$, where the expectation is with respect to the highest objective value attained by a given optimizer in each of its 10 attempts. The optimal solution to a graph instance in G_{Test} is the largest known f value found by any optimizer in any of its 10 attempts for a given depth p. We can see that the median optimality ratios achieved by RL and RLNM outperform that of Nelder-Mead for $p = \{1, 2\}$ and $p = \{1, 2, 4\}$, respectively. The median optimality gap reduction factor of RLNM with respect to Nelder-Mead ranges from 1.16 to 8.61 depending on the graph subgroup and QAOA circuit depth p.



(a) $p = 1$

(b) $p = 2$



(c) $p = 4$

Figure 4: Expected optimality ratio performance of Nelder-Mead, learned RL optimization policy, and the combined RLNM for a given QAOA circuit depth $p \in \{1, 2, 4\}$ on graph instances in G_{Test}.

5 Conclusion

In this paper, we addressed the problem of finding optimal QAOA parameters as a learning task. We propose an RL-based approach that can learn a policy network that exploits regularities in the geometry of QAOA instances to efficiently optimize new QAOA circuits. We demonstrate that there is a learnable policy that generalize well across different instances of different sizes, even when the agent is trained on small instances. The learned policy can reduce the optimality gap by a factor up to 8.61 compared with other off-the-shelf optimizers tested.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. This research was funded in part by and used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided on Bebop, a high-performance computing cluster operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

References

[1] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv:1411.4028*, 2014.

[2] Michael Streif and Martin Leib. Comparison of QAOA with quantum and simulated annealing. *arXiv preprint arXiv:1901.01903*, 2019.

[3] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Ilya Safro, Susan Mniszewski, and Yuri Alexeev. Network community detection on small quantum computers. *Advanced Quantum Technologies*, page 1900029, 2019.

[4] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Ilya Safro, Susan Mniszewski, and Yuri Alexeev. Community detection across emerging quantum architectures. *Proceedings of the 3rd International Workshop on Post Moore's Era Supercomputing*, 2018.

[5] Panagiotis Kl Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving variational quantum optimization using CVaR. *arXiv preprint arXiv:1907.04769*, 2019.

[6] Gavin E Crooks. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv:1811.08419*, 2018.

[7] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *arXiv:1812.01041*, 2018.

[8] Ojas D. Parekh, Ciaran Ryan-Anderson, and Sevag Gharibian. Quantum optimization and approximation gadialgorithms. Technical report, 2019.

[9] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Physical Review A*, 97:022304, 2018.

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

[11] Ruslan Shaydulin, Ilya Safro, and Jeffrey Larson. Multistart methods for quantum approximate opxtimization. *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019.

[12] Jonathan Romero, Ryan Babbush, Jarrod McClean, Cornelius Hempel, Peter Love, and Alán Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Science and Technology*, 2018.

[13] Dave Wecker, Matthew B Hastings, and Matthias Troyer. Training a quantum optimizer. *Physical Review A*, 94(2):022309, 2016.

[14] Zhi-Cheng Yang, Armin Rahmani, Alireza Shabani, Hartmut Neven, and Claudio Chamon. Optimizing variational quantum algorithms using Pontryagin's minimum principle. *Physical Review X*, 7(2):021027, 2017.

[15] Guillaume Verdon, Michael Broughton, Jarrod R McClean, Kevin J Sung, Ryan Babbush, Zhang Jiang, Hartmut Neven, and Masoud Mohseni. Learning to learn with quantum neural networks via classical neural networks. *arXiv preprint arXiv:1907.05415*, 2019.

[16] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural Information Processing Systems*, pages 3981–3989, 2016.

[17] Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

[18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[19] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, and et al. Qiskit: An open-source framework for quantum computing, 2019.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[21] John A Nelder and Roger Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.