Learning Based Hardware-Centric Quantum Circuit Generation

Schalkers, Merel A.; Möller, Matthias

DOI
10.1007/978-3-031-06668-9_22

Publication date
2022

Document Version
Final published version

Published in
Innovations for Community Services - 22nd International Conference, I4CS 2022, Proceedings

Learning Based Hardware-Centric Quantum Circuit Generation

Merel A. Schalkers(✉) and Matthias Möller

Delft Institute of Applied Mathematics, Delft University of Technology, Delft, The Netherlands
m.a.schalkers@tudelft.nl

Abstract. In this paper we present an approach to find quantum circuits suitable to mimic probabilistic and search operations on a physical NISQ device. We present both a gradient based and a non-gradient based machine learning approach to optimize the created quantum circuits. In our optimization procedure we make use of a cost function that differentiates between the vector representing the probabilities of measurement of each basis state after applying our learned circuit and the desired probability vector. As such our quantum circuit generation (QCG) approach leads to thinner quantum circuits which behave better when executed on physical quantum computers. Our approach moreover ensures that the created quantum circuit obeys the restrictions of the chosen hardware. By catering to specific quantum hardware we can avoid unforeseen and potentially unnecessary circuit depth, and we return circuits that need no further transpilation. We present the results of running the created circuits on quantum computers by IBM, Rigetti and Quantum Inspire.

Keywords: Quantum computing · NISQ · Quantum machine learning · Quantum compiling · Hybrid quantum computing

1 Introduction

Quantum computers are the quantum equivalent of classical computers, which have quantum specific properties, such as qubit entanglement and superposition of states. When properly exploited these properties can provide a speed-up over classical computers for some computational problems [15].

We are currently at the stage of Noisy Intermediate Scale Quantum (NISQ) devices. NISQ devices have approximately 50 to 100 noisy qubits at their disposal [16]. The noisiness of the qubits imply that we can only perform a set amount of operations on each qubit before the influence of noise on the system becomes too large, and the outcome of the computation meaningless. Since NISQ devices only have a limited amount of noisy qubits available, we are furthermore restricted in the quantum algorithms we can execute on today's and near-future quantum computers. In this paper we focus on finding quantum circuits, which are designed to take into account the physical constraints of a specific NISQ device and as such can be reliably used in the near term.

F. Phillipson et al. (Eds.): I4CS 2022, CCIS 1585, pp. 308–322, 2022.
https://doi.org/10.1007/978-3-031-06668-9_22

Aside from noise, other physical limitations of typical NISQ hardware entail that not all qubits are physically connected and only a few gates can be natively applied to the qubits. The first property implies that when a quantum circuit, which uses a two-qubit gate between non-connected qubits, gets compiled, it needs to be decomposed using multiple additional swap operations. Due to the second property many quantum gates used in a theoretical description of a quantum algorithm need to be decomposed into a sequence of native gates when run on the physical qubits. Both the first and second property described add extra depth to the quantum circuit. Since NISQ devices are rather noisy, extra depth of the circuit means less reliable results upon measurement. The aforementioned limitations of current quantum computers imply that the algorithms that can be reliably run on current quantum devices are limited.

1.1 Paper Outline

In this paper we introduce an approach that can be used to create a quantum circuit to mimic a probabilistic operation or a search operation. In Sect. 2 we give an outline of the problem and in Subsects. 2.1 and 2.2 we further elaborate on these types of operations. Furthermore the circuits found are such that they are specifically catered to a chosen quantum hardware. The circuits take into account the limitations of the hardware by restricting the potential ansatzes[1] to only use gates native to the chosen hardware. Our approach can be directly used on a physical quantum computer, which allows us to take device specific noise into account and find noise resilient circuits. Since we let our ansatz design depend on the properties of the hardware, ours is a so-called hardware-centric approach. The properties of the hardware considered in this paper and the noise-resilience of the approach are further discussed in Subsect. 2.3.

In order to be able to optimize the circuits we need to define a cost function to quantify the goodness of fit of a considered quantum circuit. Our cost function, which we introduce in Sect. 3, is such that it only distinguishes between the probability of finding each basis state upon measurement after running the found quantum circuit. We are, to our knowledge, the first to choose such a cost function for finding quantum circuits geared at mimicking probabilistic operations and search operations for learning quantum circuits. A similar cost function was used in [7], where they use this cost function to optimize a reference circuit.

Section 4 gives an overview of the method currently used to determine the ansatzes of which we optimize the continuous parameters.

We make use of both a gradient-based and a non-gradient based machine learning approach as an optimization procedure to optimize the parameters of the quantum circuit ansatzes. The non-gradient based machine learning technique we use is Particle Swarm Optimization (PSO), which we implement using PySwarms [6], and is further described in Subsect. 5.1. The gradient-based machine learning technique is implemented using PyTorch [4], in combination with Pennylane [5], and is further described in Subsect. 5.2.

[1] Here the term ansatz refers to the initial layout of the quantum circuit in terms of which gate is applied to which qubit in each layer.

We evaluated the performance of our approach by finding hardware-centric quantum circuits that mimic a thin predetermined circuit and the amplitude amplification step of Grover's search operation, respectively, using the native gate sets of physical quantum computers from Rigetti [17,18], IBM [1] and Quantum Inspire [3]. We present the resulting quantum circuits in Sect. 6.

1.2 State of the Art

There have been several other techniques which make use of a hardware-centric approach to find quantum circuits, with the aim to make optimal use of the properties of NISQ devices [8–14]. In this subsection we give a short overview of those approaches and explain how they differ from ours.

In [8] the Hilbert-Schmidt test is used to determine the cost of a chosen ansatz, which distinguishes between the desired and the created unitary and not between the resulting measurement probabilities of the basis states. Furthermore the authors evaluate their cost function by running the desired unitary and the learned quantum circuit in parallel on maximally entangled qubits. As a result their method requires $2n$ qubits and a functioning implementation of the desired unitary must be known and at the users disposal, which we do not require.

In [9] the authors also make use of a cost function that distinguishes between the quantum states, not probability vectors upon measurement. Therefore their choice of cost function is also not suitable for our use cases.

Paper [10] again distinguishes between the unitary created by the ansatz and the desired unitary, not the associated probability vectors. This requires to find the exponentially large matrix expressions for the desired and created unitaries and calculate the distance between the two.

Papers [11–13] only learn quantum circuits that create one specific probability distribution given a specific input vector. The proposed approaches do not learn circuits that can be used on a more general input.

In Paper [14] the authors also make use of a cost function distinguishing between the fidelity of the resulting and desired quantum states, not the probability of finding each basis state upon measurement. Next to that they do not built up the learned circuits using quantum gates native to a chosen quantum computer, meaning that their resulting circuits would still need to be decomposed into native gates before they can be run on a physical quantum device.

2 Problem Description

Given a function $f(x) : \mathbb{C}^{2^n} \rightarrow \mathbb{R}_+^{2^n}$, where we assume $\|x\|_2 = 1$ and $\|f(x)\|_1 = 1$, we want to find a quantum circuit $U(\Theta)$, such that $|\langle i| U(\Theta) |x\rangle|^2 \approx f(x)_i$ for all $i \in \{0, 1, \ldots, 2^n - 1\}$. Here, $|i\rangle$ represents a computational basis state for the space $\mathbb{C}^{2^n}$. Our method works equally well with any orthonormal choice of basis, we have chosen the computational basis for readability. In this paper we describe our method to learn such a quantum circuit $U(\Theta)$, which can be used to mimic probabilistic operations and search operations.

2.1 Probabilistic Operations

Definition 1. *Let $g : \mathbb{C}^N \to \mathcal{S}$ be a probabilistic operation, where $\mathcal{S} := \{0, 1, \ldots, N-1\}$ and we restrict the input vectors to $x \in \mathbb{C}^N$ such that $\|x\|_2 = 1$. The probabilistic operation g takes input $x \in \mathbb{C}^N$ and returns i with probability p_i.*

Notice that a probabilistic operation is not a function, as the same input can give different outputs. We can use g to create a function f, for which $f(x)_i = p_i$. We can then use our approach to find the quantum circuit $U(\Theta)$, that acts on $n = \lceil \log_2(N) \rceil$ qubits, for which $|\langle i| U(\Theta) |x\rangle|^2 \approx f(x)_i$ holds. Let $|i\rangle$ represent $i \in \mathcal{S}$, then applying such a circuit $U(\Theta)$, followed by a measurement in the computational basis, acts precisely like g on an input state in the space $\mathbb{C}^{2^n}$.

An example of a probabilistic operation is the application of a quantum circuit V on a quantum state $|\phi\rangle$, followed by a measurement in a predetermined basis. This implies that our approach can be used to find a circuit $U(\Theta)$, that simulates the behaviour of any chosen quantum circuit V followed by a measurement. Since we limit our approach to learn the correct circuit up to measurement in a predetermined basis, we can find thinner circuits than theoretically possible[2] when finding a decomposition for a quantum circuit into native gates.

Seeing a quantum circuit as a probabilistic operation can be particularly useful for hybrid quantum computing, where a subroutine is run on a quantum computer and its resulting measurement used in the remainder of the routine.

2.2 Search Operations

Definition 2. *A search operation $h(x) : \mathbb{C}^N \to \mathcal{S}$, with $\mathcal{S} := \{0, 1, \ldots, N-1\}$, is such that it returns a result $i \in \mathcal{M}_x$, where $\mathcal{M}_x \subseteq \mathcal{S}$ is the set of solution states for the given input x. When two solution states k and l are in the same set of possible outcomes $\mathcal{M}_x$, we say they are equivalent under x. For the search operation the probabilities of returning each specific state is irrelevant as long as the state returned is in the set $\mathcal{M}_x$.*

Notice that a search operation is also not a function as it can return different states for the same input, as long as they are equivalent under the input. We can then use h to create a function f, such that

$$f(x)_i = \begin{cases} \frac{1}{|\mathcal{M}_x|} & \text{for } i \text{ in } \mathcal{M}_x \\ 0 & \text{otherwise.} \end{cases}$$

Subsequently we employ our approach to find a quantum circuit $U(\Theta)$, such that $|\langle i| U(\Theta) |x\rangle|^2 \approx f(x)_i$. Note that we could choose any other function $\tilde{f}$

[2] An example of this is the Hadamard gate H. There are multiple textbook decompositions into rotation operations, all requiring 3 rotations, an example is $H = e^{\frac{i\pi}{2}} R_Z\left(\frac{\pi}{2}\right) R_X\left(\frac{\pi}{2}\right) R_Z\left(\frac{\pi}{2}\right)$. When the application of H is followed by a measurement in the computational basis, however, it suffices to apply $\tilde{H} = R_Z\left(\frac{\pi}{2}\right) R_X\left(\frac{\pi}{2}\right)$ to the qubits instead.

to describe our search operation h, as long as $\tilde{f}(x)_i$ exclusively returns nonzero values for $i \in \mathcal{M}_x$

An example of a search operation is Grover's search algorithm [23]. Here the goal is to identify a non-zero index i of a given binary vector y. In the problem the size N and Hamming weight[3] t of the vector y are assumed to be known. Furthermore access to an oracle function $\mathcal{O}_y$ is assumed. One can translate this to a search operation by translating the binary vectors y to an input state $s \in \mathbb{C}^N$ and subsequently returning a basis state representing an index i for which $x_i = 1$. In Subsect. 6.2 we go into further detail of Grover's search operation and explain how to uniquely translate a binary vector y to a corresponding input state s.

2.3 Properties of Quantum Computers Considered

We restrict possible ansatzes to have a predetermined depth and to be built up from gates native to the chosen quantum computer between physically connected qubits. These restrictions are implemented to ensure that the found circuits $U(\Theta)$ will be directly executable on near term quantum devices. We implemented a Python program to randomly create such ansatzes with a predetermined number of qubits n and circuit depth D, for a chosen quantum hardware.

An example of such a native gate set used in our study is $\{R_X\left(\pm\frac{\pi}{2}\right), R_Z(\theta),$ CZ, XY $(\theta)\}$, which is the set of native gates used by Rigetti [17,18]. We also consider the IBM [1] and Quantum Inspire [3] native gate sets which are $\{R_X\left(\frac{\pi}{2}\right),$ $R_X(\pi)$, $R_Z(\theta)$, CNOT$\}$ and $\{R_X(\theta)$, $R_Y(\theta)$, $R_Z(\theta)$, CZ$\}$, respectively.

Some native gates on a quantum computer are parameterized, meaning that the created ansatz $U(\Theta)$ has continuous variables Θ to optimize. The vector $\Theta \in [0, 2\pi)^{|\Theta|}$ consists of the parameters θ of the parameterized quantum gates in the ansatz. In this paper we optimize over the quantum gate parameters. In future work we plan to extend our method to include optimization approaches for the discrete problem of determining which gates to apply to the qubits.

Note that when we use our quantum circuit generation approach in combination with a physical quantum computer or noisy simulator, we automatically take noise into account. This implies that we might find quantum circuits which have a slight offset on paper but perform better in practice when run on real hardware. In this way our approach is particularly well suited for the NISQ era.

3 Cost Function

In order to distinguish the goodness of fit of a certain quantum circuit $U(\Theta)$, we use the following cost function:

$$C(U(\Theta), \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{|\phi\rangle \in \mathcal{S}} \| f(|\phi\rangle) - f_{U(\Theta)}(|\phi\rangle) \|_2. \tag{1}$$

[3] The Hamming weight of a binary vector equals the amount of nonzero indices.

Here, $\mathcal{S} = \{|\phi_0\rangle, \ldots, |\phi_{l-1}\rangle\}$ is the batch of input states considered for the training. The function f expresses the desired behaviour as described in Sect. 2 and $f_{U(\Theta)}$ is defined as in [7]:

$$f_{U(\Theta)}(|\phi\rangle) = \left(|\langle i|U(\Theta)|\phi\rangle|^2\right)_{i=0,\ldots,2^n-1}, \tag{2}$$

where $U(\Theta)$ is the quantum circuit to be created. Notice that our cost function is defined over the probabilities of finding each basis state after applying the circuit $U(\Theta)$ to the quantum states $|\phi\rangle \in \mathcal{S}$. We stress that our cost function does not distinguish between two quantum states which are distinct, but will lead to the same measurement outcomes in the chosen basis.

Since we define our cost function over the resulting probability vectors of applying our learned circuit to the quantum states in the batch $\mathcal{S}$ and subsequently performing a measurement, we do not need expressions for the resulting quantum states $U(\Theta)|\phi\rangle$ or the created circuit $U(\Theta)$. Only the vectors $f_{U(\Theta)}(|\phi\rangle)$ are required, and these can be estimated by performing measurements in the chosen basis after performing the circuit $U(\Theta)$ to the input states $|\phi\rangle$.

As we learn over a batch of possible input quantum states and their associated measurement results, it is also not necessary to have a known expression or quantum circuit for the "perfect" quantum operation V. In fact, a quantum operation which perfectly performs our chosen probabilistic or search operation need not even exist for our method to work. This feature makes our approach widely applicable as we are not limited to learning quantum circuits we can already (theoretically) perform. This is one of the outstanding features of our approach we wish to highlight.

A common choice of function to evaluate the distance between two probability vectors is the Kullback-Leibler divergence [15]:

$$D_{\mathrm{KL}}(f|f_{U(\Theta)}) = \sum_{i=1}^{2^n-1} f(|\phi\rangle)_i \log \frac{f(|\phi\rangle)_i}{f_{U(\Theta)}(|\phi\rangle)_i}. \tag{3}$$

Note that our cost function (1) is relatively cheap compared to the Kullback-Leibler divergence. Taking the logarithm is an expensive operation, and since in our use cases many indices $f(|\phi\rangle)_i$ will have value 0 we will need to replace those with a small value ϵ to avoid NaNs. These properties make our cost function more suited for the chosen application than the Kullback-Leibler divergence.

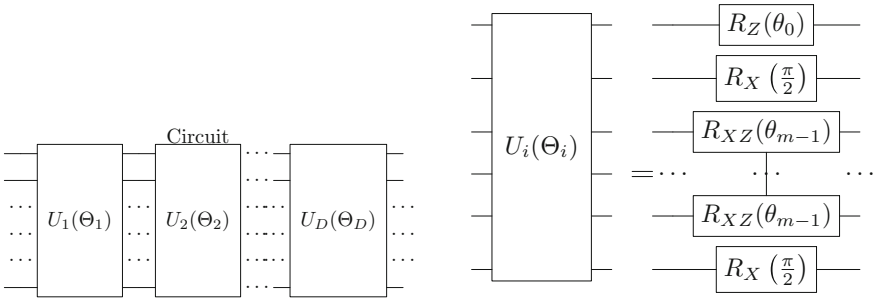
Our cost function takes the 2-norm between the found probability distribution and the sought after probability distribution. Therefore the outcome of our cost function is not directly informative, as its value is influenced by the number of qubits n and the chosen batch $\mathcal{S}$ to learn over. This implies that we cannot use its value as a measure for the goodness of fit of the quantum circuit $U(\Theta)$. However, our cost function is suited to distinguish between good and bad ansatzes and parameter values Θ, and is therefore a good choice for practical use.

4 Determining Ansatzes

As described in Sect. 2, we only consider ansatzes composed of hardware native gates and respecting the sparse connectivity of qubits.

In the current approach we create a random ansatz $U(\Theta)$ of predetermined depth D and number of qubits n for a chosen quantum hardware, with the ansatz restricted to the native gate set and hardware topology. We subsequently optimize the parameters Θ to determine the value of the cost function $C(U(\Theta), \mathcal{S})$ as given in (1) for a batch of input states $\mathcal{S}$ of predetermined size $|\mathcal{S}|$. Figure 1 illustrates such a random ansatz $U(\Theta)$ and a zoom in of a single layer. We create a large amount of random ansatzes and optimize their continuous parameters Θ as described in Sect. 5, from which we determine the best one afterwards.

In this paper we focus on the optimization of the continuous parameters Θ for a fixed U. As subject of ongoing research we are currently working on adding layers of discrete optimization to our method. In the current method, using the computational basis, we have made use of certain rules such as that any R_Z or CZ gate appearing right before measurement falls away.



(a) Quantum circuit representing the found valid ansatz of which the unitary operations are built up using only natively implemented quantum gates

(b) An example of a valid layer U_i for an IBM quantum computer, where all the gates used are natively implemented on the specified hardware

Fig. 1. Representation of a random quantum ansatz $U(\Theta)$

5 Optimization of Continuous Parameters

Once we have created a hardware-centric ansatz for our quantum circuit $U(\Theta)$, we need to optimize the continuous parameters Θ. In this work we consider two different optimization approaches to find the optimal value of Θ, one non-gradient based approach called Particle Swarm Optimization (PSO) and one gradient-based optimization approach.

5.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolutionary meta-heuristic which mimics swarm intelligence to find a point of minimal cost in a large scale space. In this paper the particles Θ represent an instance of the parameter vector, therefore our particles live in a $|\Theta|$ dimensional hypercube $\Theta \in [0, 2\pi)^{|\Theta|}$. In each iteration i of the scheme the particles Θ move through the parameter space and their velocity v_j is subsequently updated:

$$v_j^{i+1} = \omega v_j^i + c_1 \left(p_j - \Theta_j\right) + c_2 \left(\Theta_{\text{opt}} - \Theta_j\right). \tag{4}$$

Here Θ_{opt}, p_j and Θ_j are parameters which are updated in each iteration, whereas ω, c_1 and c_2 are hyperparameters determined beforehand. The parameter Θ_{opt} represents the best position in space found by any particle so far and the parameter p_j represents the best position found by the particle j. The hyperparameter ω represents the inertia coefficient, i.e. the particles unwillingness to change direction. The hyperparameter c_1 is the memory coefficient which determines how heavily past success influences the direction the particle travels in. Finally, c_2 is the social coefficient which represents how heavily the best location in space found by any particle influences the velocity in the next iteration.

By iteratively running this scheme for many particles, all the Θ_j will slowly cluster together at a point in space $\tilde{\Theta}$ which represents a (global) minimum. As PSO is a meta-heuristic it cannot give any guarantees that a minimum will be found. Our results presented in Sect. 6 show that PSO performs very well for the different test cases considered in this paper. For our implementation of the non-gradient based approach we used PySwarms version 1.3.0 [6].

5.2 Gradient-Based Optimization

As an alternative to the PSO approach we utilized a gradient-based optimization method to find the optimal parameters Θ of the quantum circuit U.

In this case we optimize the circuit parameters Θ by calculating the gradient of the cost function (1), with respect to θ, where θ is the i-th index of Θ. We then update Θ in the direction of the negative gradients:

$$\Theta = \Theta - l \nabla c\left(\Theta\right). \tag{5}$$

In the above equation l is a hyperparameter which represents the learning rate. As there are $|\Theta|$ input parameters and only one output parameter, backpropagation is used to calculate the gradients in a computationally efficient way.

To calculate the gradients, set $\hat{Z}_i := |i\rangle\langle i|$ to get the following equality:

$$\langle \hat{Z}_i \rangle_{U(\Theta)} = f_{U(\Theta)}(|\phi\rangle)_i. \tag{6}$$

Here $\langle \hat{Z}_i \rangle_{U(\Theta)}$ is the expectation value of the observable $\hat{Z}_i$ upon measurement of $U(\Theta)|\phi\rangle$. It follows that the cost function can be expressed as the expectation value of the observable $\hat{Z}_i$, of which the gradient can be calculated as shown in

the Paper by Schuld et al. [19]. Gradients of an expectation value of an observable with respect to θ can be evaluated on a quantum computer directly. Let us define:

$$U(\Theta) = V(\Theta_0^{i-1})G(\theta_i)W(\Theta_{i+1}^{|\Theta|}) = VG(\theta_i)W. \tag{7}$$

In the above equation $G(\theta_i)$ represents the layer which only consists of the quantum gate depending on the parameter θ_i. Furthermore we set $|\phi'\rangle = W|\phi\rangle$ and $\hat{Q}_i = V^\dagger \hat{Z}_i V$, note that $\hat{Q}_i$ is still an observable.[4]

For all native gates except $XY(\theta)$ considered in this paper the partial derivative with respect to θ_i can be calculated using the following equality [19]:

$$\begin{aligned}\partial_{\theta_i} f(\Theta) = &\frac{r}{2}\left\langle \phi' | G^\dagger\left(\theta_i + \frac{\pi}{4r}\right)\hat{Q}_i G\left(\theta_i + \frac{\pi}{4r}\right)|\phi'\right\rangle \\ &- \frac{r}{2}\left\langle \phi' | G^\dagger\left(\theta_i - \frac{\pi}{4r}\right)\hat{Q}_i G\left(\theta_i - \frac{\pi}{4r}\right)|\phi'\right\rangle.\end{aligned} \tag{8}$$

In the above equation r is a constant that depends on the operation $G(\theta)$ of which we determine the derivative. Since $XY(\theta)$ has more than two distinct eigenvalues $\pm r$, we need a different technique to calculate the gradients. This can either be accomplished by using the second method presented by Schuld et al. [19], or using the method presented by Banchi and Crooks in [20].

The partial derivative of the cost function with respect to θ_i, $\partial_{\theta_i} f(\Theta)$, can be evaluated on a classical computer in combination with backpropagation [21].

For our implementation of the gradient based optimization approach we made use of PyTorch version 1.10.0 [4] and Pennylane version 0.19.1 [5].

6 Results

In this section we demonstrate the performance of our approach in generating executable quantum circuits that mimic probabilistic operations and search operations. We first use our quantum circuit generator to mimic a probabilistic operation we have named thin circuit. We subsequently use our approach to generate a quantum circuit to mimic the amplitude amplification step of Grover's search algorithm, which is a search operation as described in Subsect. 2.2.

6.1 Thin Circuit

The first test for our machine learning approach is to learn the function f, where $f : \mathbb{C}^{2^n} \to \mathbb{R}_+^{2^n}$ and $f(|x\rangle) = |\langle i|A|x\rangle|^2$, here

$$A := \begin{cases} (X \otimes H)^{\otimes \frac{n}{2}}, & \text{for } n \text{ even} \\ (X \otimes H)^{\otimes \frac{n-1}{2}} \otimes X, & \text{for } n \text{ odd.} \end{cases} \tag{9}$$

[4] An observable is a Hermitian matrix, multiplying a Hermitian matrix from both sides with some unitary V results in a Hermitian matrix $Q = V^\dagger \hat{Z} V$. In this case, since $\hat{Z}_i = |i\rangle\langle i|$ we get $\hat{Q}_i = V^\dagger \hat{Z}_i V = |\xi\rangle\langle\xi|$, where $|\xi\rangle$ is some quantum state.

In the above expression, H is the Hadamard and X the NOT gate.

Using our approach we were able to find a decomposition $U(\Theta)$ using only two layers $U_i(\Theta_i)$ where $i \in \{1, 2\}$, whereas the theoretical decomposition requires three layers. This has to do with our choice of the cost function. Our generated quantum circuit creates the same size amplitudes for each possible outcome state, but with a relative phase shift between the states.

After using our approach to find a circuit for $n = 2$, we extrapolated this circuit to $n \in \{5, 8, 10, 20\}$ and optimized those instances. With PSO we were able to find optimal quantum circuits for all instances. Our gradient-based approach was able to find optimal quantum circuits for $n \in \{2, 5, 8\}$, for larger n we ran into the issue of barren plateaus. In the literature barren plateaus have been shown to occur for growing circuit depths [22], as well as when the number of qubits grows [8]. Using PSO, our approach was able to correctly optimize the created quantum circuit, with a cost function defined over a batch as small as $|\mathcal{S}| = 2$ for $n \in \{2, 5, 8, 10, 20\}$. This shows that in certain cases we are able to find the right results, while optimizing over a subspace much smaller then theoretically required to span the space. Our gradient-based approach was able to optimize the quantum circuit for batch sizes as small as $|\mathcal{S}| = 2$ for $n \in \{2, 5\}$, a batch of size $|\mathcal{S}| = 4$ was required to correctly optimize the circuit for $n = 8$.

6.2 Grover's Search

Next we use our approach to learn a circuit $U(\Theta)$ to mimic the amplitude amplification step of Grover's search algorithm [23]. Grover's search algorithm can be used to find the non-zero indices of a binary vector x of length N and Hamming weight t, assuming that access to the oracle $\mathcal{O}_x$ is provided. The oracle $\mathcal{O}_x$ is defined as follows: Let $|i\rangle$ be a computational basis state, then $\mathcal{O}_x |i\rangle = (-1)^{x_i} |i\rangle$. Furthermore it is assumed that N and t are known to the user.

In this paper we restrict ourselves to finding circuits for the amplitude amplification step of Grover's search algorithm for vectors x with $t = \frac{N}{4}$. We restrict ourselves to this case, as it is such that the amplitude amplification step of Grover's search algorithms only requires one query to the oracle.

To summarise, we try to find a quantum circuit $U(\Theta)$ that mimics:

$$f(\mathcal{O}_x |+\rangle)_i = \begin{cases} \frac{1}{t}, & \text{if } x_i = 1, \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

with $i \in \{0, 1, \ldots, N-1\}$ for all possible inputs $x \in \text{bin}(N, t)$, where $\text{bin}(N, t)$ is the set of binary vectors of length N with Hamming weight t. Since we are working in a space of size N, we require $n = \log_2(N)$ qubits. In (10) the quantum state $|+\rangle$ is defined as $|+\rangle := H^{\otimes n} |0\rangle$.

Let us define the states $|\phi_x\rangle := \mathcal{O}_x |+\rangle$, which represent the quantum state reached in Grover's search algorithm after applying the oracle function. In our approach we learn over the batch $\mathcal{S}_{N,t} = \{|\phi_x\rangle\}$, for $x \in \text{bin}(N, t)$. By learning over $\mathcal{S}_{N,t}$ we find a $U(\theta)$, which can be used as the amplitude amplification step in Grover's search algorithms for problems with size N and Hamming weight t.

We first use our approach to find circuits to mimic the amplitude amplification step for the case $N = 4$ and $t = 1$. Using the IBM native gate set we were able to find a circuit of depth $D = 4$ that solves the problem with certainty, whereas the textbook circuit compiled by the IBM toolchain gives rise to a circuit of depth $D = 14$. For the Rigetti native gate our approach found a circuit of depth $D = 2$ that solves the problem with certainty. On the other hand, the textbook circuit compiled using Rigetti's toolchain [2] results in a quantum circuit of depth 10. For the Starmon 5 quantum processor available through the Quantum Inspire cloud service we were able to find a quantum circuit with depth $D = 3$ that solves the problem with certainty. As we do not have access to the compiled version of the textbook circuit, we compare our result with a theoretical decomposition which suggests it would have a depth $D = 7$.

This shows that our method can lead to a significantly thinner quantum circuits, while still finding the correct quantum state upon measurement with certainty. Figure 2 depicts the textbook circuit compiled for Rigetti and its counterpart produced by our quantum circuit generation approach. In what follows, we will use the abbreviation QCG whenever we refer to our approach.

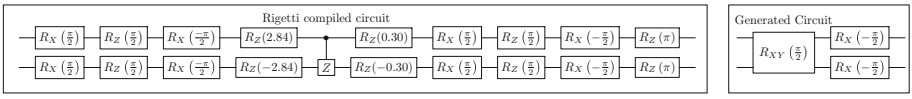


Fig. 2. Example of executable quantum circuits for the 2-qubit amplitude amplification step of Grover's search algorithm. Left: compiled textbook circuit. Right: QCG circuit

For the amplitude amplification step of Grover's search algorithm with $N = 8$ we were able to create significantly thinner circuits for both the IBM and Rigetti native gate sets. For instance, the QCG circuit has depth 8, whereas the textbook circuit compiled with the Rigetti compiler [2] consists of 63 layers. It needs to be stressed that these QCG created circuits do not return the correct result with certainty, the probabilities of returning the correct results are around 55 percent. This appears to be a result of the known barren plateau issue in quantum machine learning training landscapes [8,22]. We plan to combat this issue in future work, by adding layers of discrete optimization.

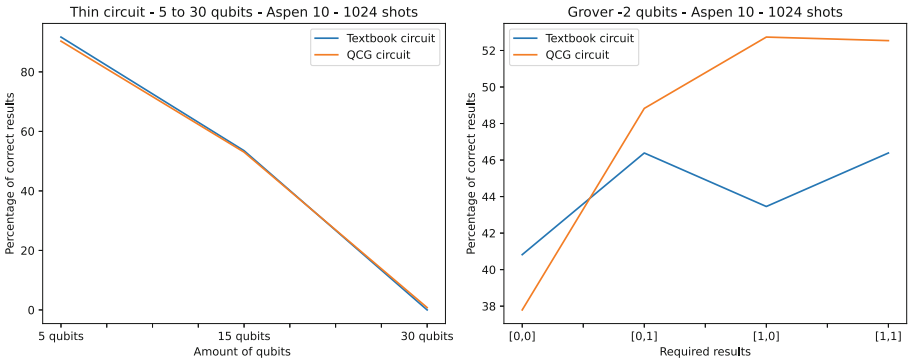
7 Experimental Results on Quantum Computers

In this section we present a representative selection of experimental results that were obtained by executing the QCG circuits on physical quantum computers. We compare the measurement outcomes with that of textbook circuits, which were turned into executable circuits with the help of the vendor toolchains.

7.1 Rigetti - Aspen 10

The Rigetti Aspen 10 quantum computer has a total of 32 qubits. We used this computer to test and compare the results of the generated and textbook version of the thin circuit operation and amplitude amplification step of Grover's search.

For the thin circuit example, Fig. 3a shows that the QCG circuit outperforms the compiled textbook circuit as the number of qubits grows, though the overall result becomes dominated by noise in both cases. Figure 3b shows that the QCG version of the amplitude amplification step of Grover's search returns the correct result with a slightly higher percentage then the compiled textbook circuit.



(a) Thin circuits for 5-30 qubits (b) 2-qubit amplitude estimation

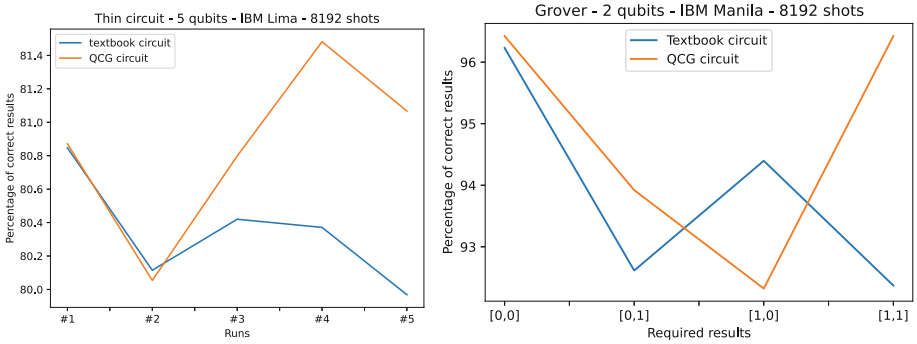
Fig. 3. Performance comparison between compiled textbook circuits and circuits created with our QCG approach for the Rigetti Aspen 10 quantum computer

7.2 IBM - Lima and Manila

The IBM Lima and Manila machines are open access quantum computers which both have 5 qubits available. We used the IBM Lima computer to compare the results of our QCG circuits with that of the compiled textbook thin circuit for 5 qubits. We used the IBM Manila computer to compare the results of our QCG amplitude amplification step of Grover's search for 2 qubits to the compiled textbook circuit. The results of these runs are shown in Fig. 4. It can be seen that in both cases the QCG circuits performs slightly better.

7.3 Quantum Inspire - Starmon 5

The Quantum Inspire Starmon 5 is an open access quantum computer with 5 qubits available. In contrast to IBM and Rigetti, Quantum Inspire does not provide a full software stack that decomposes a given quantum circuit into one that can be run on the hardware. Here the benefit of our approach is particularly apparent as any found QCG circuit can always be directly applied to the



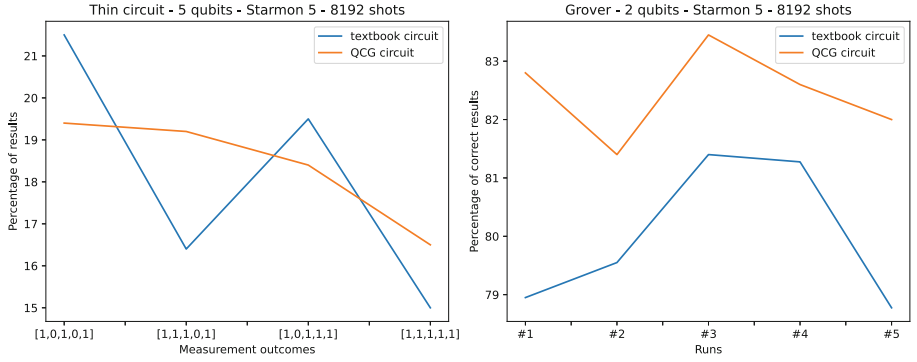
(a) Thin circuits for 5 qubits (b) 2-qubit amplitude estimation

Fig. 4. Performance comparison between compiled textbook circuits and circuits created with our QCG approach for the IBM Lima and Manila quantum computers

hardware. Using our quantum circuit generation method we learned a quantum circuit to mimic the thin circuit operation for 5 qubits and the amplitude amplification step of Grover's search for 2 qubits. We note that for the Quantum Inspire results we exclusively made use of the non-gradient based PSO approach.

Upon running and comparing the thin circuit results, using both our QCG circuit and a textbook circuit that we turned into an executable circuit by hand using theoretical decompositions and manual qubit placement, we found that the probability of finding one of the correct basis states upon measurement remained equal. However, our generated circuit results in a much more even spread of the possible basis states to be found upon measurement, as is desired. This shows that our method can lead to more reliable quantum circuits; see Fig. 5a.

In Fig. 5b we compare the outcomes of running the textbook circuit of the amplitude amplification step of Grover's search on the Starmon 5 with that of the QCG circuit. Here the QCG circuit performs overall significantly better.



(a) Thin circuits for 5 qubits (b) 2-qubit amplitude estimation

Fig. 5. Performance comparison between compiled textbook circuits and circuits created with our QCG approach for the Quantum Inspire Starmon 5 quantum computer

8 Discussion

In this paper we have presented a method to make use of NISQ devices, by implementing machine learning methods to learn circuits for mimicking probabilistic and search operations. To this end, we introduced a cost function that only distinguishes between finding the correct basis states upon measurement.

We have shown that our method can be successfully implemented with both gradient-based and non-gradient based machine learning to create thin quantum circuits, which behave as desired. The created quantum circuits are thinner then their compiled textbook counterparts and can lead to more stable results when executed on physical quantum computers. In doing so we have shown the potential of our cost function for generating quantum circuits.

At the current stage, our method is restricted to work on quantum circuits for relatively small numbers of qubits and circuit depth. This is due to the barren plateau issue known in quantum machine learning approaches. In further research we will extend our approach by implementing a discrete optimization method to circumvent the barren plateau issue.

Acknowledgment. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. We also wish to express our gratitude to IBM and Quantum Inspire for making their quantum computers available online for public use. Furthermore the authors thank David de Laat for his useful ideas, fruitful discussions and valuable feedback on the manuscript.

References

1. IBM Quantum (2021). https://quantum-computing.ibm.com/
2. Smith, R.S., Curtis, M.J., Zeng, W.J.: A practical quantum instruction set architecture (2016)
3. QuTech: Quantum Inspire Home. Retrieved from Quantum Inspire (2018). https://www.quantum-inspire.com/
4. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library (2019)
5. Bergholm, V., et al.: PennyLane: automatic differentiation of hybrid quantum-classical computations (2020)
6. Miranda, L.J.V.: PySwarms: a research toolkit for Particle Swarm Optimization in Python. J. Open Sour. Softw. **3**, 433 (2018)
7. Adarsh, S., Möller, M.: Resource optimal executable quantum circuit generation using approximate computing (2021)
8. Khatri, S., LaRose, R., Poremba, A., Cincio, L., Sornborger, A.T., Coles, P.J.: Quantum-assisted quantum compiling. Quantum **3**, 140 (2019)
9. Cincio, L., Rudinger, K., Sarovar, M., Coles, P.J.: Machine learning of noise-resilient quantum circuits (2020)
10. Davis, M.G., Smith, E., Tudor, A., Sen, K., Siddiqi, I., Iancu, C.: Heuristics for quantum compiling with a continuous gate set (2019)
11. Benedetti, M., et al.: A generative modeling approach for benchmarking and training shallow quantum circuits (2019)

12. Zhu, D., et al.: Training of quantum circuits on a hybrid quantum computer (2019)
13. Liu, J.-G., Wang, L.: Differentiable learning of quantum circuit born machine (2018)
14. Martinez, E.A., Monz, T., Nigg, D., Schindler, P., Blatt, R., et al.: Compiling quantum algorithms for architectures with multi-qubit gates. New J. Phys. **18**, 063029 (2016)
15. Nielsen, M., Chuang, I.: Quantum computation and quantum information (2000)
16. Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum J. **2**, 79 (2018)
17. Abrams, D.M., Didier, N., Johnson, B.R., da Silva, M.P., Ryan, C.A.: Implementation of the XY interaction family with calibration of single pulse (2019)
18. Murali, P., Linke, N.M., Martonosi, M., Abhari, A.J., Nguyen, N.H., Alderete, C.H.: Full-stack, real-system quantum computer studies: architectural comparisons and design insights (2019)
19. Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., Killoran, N.: Evaluating analytic gradients on quantum hardware (2018)
20. Banchi, L., Crooks, G.E.: Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule (2021)
21. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning (2016)
22. McClean, J.R., Boixo, S., Smelyanskiy, V.N., Babbush, R., Neven, H.: Barren plateaus in quantum neural network training landscapes (2018)
23. Grover, L.K.: A fast quantum mechanical algorithm for database search (1996)