Mutual Unbiased Bases as a Countermeasure to Barren Plateaus

Ittay Alfassi*

September 29, 2022

Contents

*Email: ittay.al@cs.technion.ac.il

1 Introduction

Noisy, Intermediate-Scale Quantum (NISQ) computers are quantum computers that are limited in certain ways. They are usually limited in the number of qubits, connectivity between qubits, noise, and maximal circuit depth. A computational paradigm that utilizes NISQ devices is the paradigm of hybrid quantum-classical algorithms. Most hybrid algorithms translate a given problem into an optimization problem. The parameters of the optimization problem control some parametrized quantum state, the problem's cost function is evaluated on the quantum computer, and the optimization steps are performed by the classical computer. These algorithms are called Variational Quantum Algorithms (VQAs) [1]. TODO: Make sure this is accurate!

While being a leading methodology for the use of quantum computers today, VQAs suffer from several problems. One such problem is the problem of Barren Plateaus [2]. Informally, barren plateaus imply that over the optimization parameter space, the gradient of the cost function is negligibly small and the optimization process does not how to converge.

This problem affects many VQAs that employ different types of quantum circuits and classical optimizers (even those that are gradient-free). As the number of qubits and the depth of the quantum circuits increase, this problem worsens exponentially. Effectively, this means that VQAs will not be able to give an advantage over their classical counterparts in problems that are not small in size.

To address this issue, in this project, I will try and utilize Mutually Unbiased Bases (MUBs), their states, and their properties. The basic intuition behind this method is that, for any number of qubits, the set of all MUB states spans the entire Hilbert space (with real coefficients).

Thus, inserting them (in some fashion) into the optimization process can allow for an "exhaustive search" over the Hilbert space of all states.

Unfortunately, this method did not succeed. However, I will detail the different aspects of this approach, the experiments implemented, the conclusions I reached, and possible methods to still utilize the idea of MUB states in VQAs.

This project is mainly based on the work of Arrasmith et al. [3].

2 Preliminaries

2.1 Variational Quantum Algorithms

A Variational Quantum Algorithm is an algorithm that takes some computational problem and solves it by hybrid classical-quantum optimization. VQAs are comprised of several elements: The quantum circuit, the cost function, and the optimizer.

1. **The Quantum Circuit.** Every VQA problem uses a quantum circuit with some parametric values. Its structure can be generally written as the following:

$$U(\vec{\theta}) = \prod_{l=1}^{L} U_l(\theta_l) W_l \tag{1}$$

   Where $U_l(\theta_l) = \exp(-i\theta_l V_l)$, V_l is a Hermitian operator (thus U_l is a unitary operator), and W_l is a generic unitary operator that does not depend on any angle θ_l.

In some VQAs, an **initial state** $|\psi_0\rangle$ is also a part of the algorithm, and it is related to the quantum circuit.

The term *ansatz* (German for "approach"/"attempt") is used in literature to either describe the initial state $|\psi_0\rangle$, the parametric circuit $U(\vec{\theta})$, or the application of the parametric circuit to the initial state $|\psi(\vec{\theta})\rangle = U(\vec{\theta})\,|\psi_0\rangle$. In this project, I will use the term ansatz to refer to the parametric quantum circuit.

Ansatzes[1] can either be problem-specific or problem-agnostic.

A problem-specific ansatz is tailored to the problem solved by the VQE using a-priori knowledge of the structure of the problem. Through this tailoring, there is a better chance that the circuit can reach values that are optimal for the problem at hand. When using a problem-specific ansatz, the initial state is usually also tailored according to the same information.

A problem-agnostic ansatz is not tailored to the problem specifically and can be used with no relevant information. It is usually comprised of repeated layers of quantum gates. Each layer is comprised of quantum gates that are usually easy to implement directly on quantum hardware, in contrast to problem-specific ansatzes. Problem-agnostic ansatzes are usually called "hardware-efficient" for this reason. In this project, I focused on using hardware-efficient ansatzes.

2. **The Cost Function.** The cost function $C(\vec{\theta})$ defines the goal of the VQA. It is defined as a function from the parameters of the ansatz to the real numbers. The goal of the VQA is to find

$$\vec{\theta}^* = \arg\min_{\vec{\theta}} C(\vec{\theta}) \tag{2}$$

The cost function always depends on the ansatz, some Hermitian operators, and some initial states. Generally, it can be written as

$$C(\vec{\theta}) = \sum_k f_k(\mathrm{Tr}[O_k U(\vec{\theta})\rho_k U^\dagger(\vec{\theta})]) \tag{3}$$

Where $\{f_k\}$ is a set of functions, $\{O_k\}$ is a set of operators, and $\{\rho_k\}$ is a set of input states.

The trademark of VQAs is that they use a quantum computer to estimate the cost function $C(\vec{\theta})$ (or its derivatives) while leveraging the power of classical optimizers to train the parameters $\vec{\theta}$.

3. **The Optimizer.** While not necessarily dependent on the problem the VQA is trying to solve, the classical optimizer is an important part of the algorithm. The optimizer is run on a classical computer, and performs an iterative process: it uses data on the current parameters of the cost function to calculate their value in the next iteration. More details can be found in subsection 2.2.

2.1.1 Example: Variational Quantum Eigensolver

One of the earliest examples of a VQA is the VQE algorithm, proposed by Peruzzo et al. [4]. In this algorithm, The goal is to find the lowest eigenvalue of some Hamiltonian H.

[1] The correct plural form in German is Ansaetze.

The cost function is straightforwardly defined as the expectation value of the parametric state over the Hamiltonian:

$$C(\vec{\theta}) = \langle\psi(\vec{\theta})| H |\psi(\vec{\theta})\rangle = \langle\psi_0| U^\dagger(\vec{\theta}) H U(\vec{\theta}) |\psi_0\rangle \tag{4}$$

In case H is a molecular Hamiltonian (converted to a qubit Hamiltonian using an appropriate transformation), a problem-specific ansatz can be used. One such option is the Unitary Coupled-Cluster (UCC) ansatz, together with the Hartree-Fock state as the initial state.

Of course, a hardware-efficient ansatz can be used for any structure of H.

2.1.2 Example: Variational Quantum Compiling

A different example of a VQA is the problem of Variational Quantum Compiling (VQC), initially defined by Khatri et al. [5] as Quantum-Assisted Quantum Compiling (QAQC).

In VQC, the input of the problem is a general n-qubit unitary operation V.[2] The goal of the algorithm is to control the parameters of a parametric quantum circuit $U(\vec{\theta})$ so its behavior will be similar to that of V.

Formally, there are two possible goals: either that $U(\vec{\theta})$ and V will perform the same operation on all states, or that $U(\vec{\theta})$ and V will perform the same operation on a specific input state. In this project, I will use the latter as the problem definition. Thus, we wish that $V |\vec{0}\rangle = U(\vec{\theta}) |\vec{0}\rangle$.

There are two different cost functions presented in [5]. The first is straightforward and is defined by

$$C^{\text{global}} = 1 - \left| \langle\vec{0}| UV^\dagger |\vec{0}\rangle \right|^2 \tag{5}$$

Another cost function, called the local cost function, is defined by

$$C^{\text{local}} = 1 - \frac{1}{n} \sum_{j=1}^{n} p_0^{(j)} \tag{6}$$

where

$$p_0^{(j)} = \text{Tr}[(|0\rangle \langle 0|_j \otimes I) U^\dagger V |\vec{0}\rangle \langle\vec{0}| V^\dagger U] \tag{7}$$

and $|0\rangle \langle 0|_j \otimes I$ is the tensor product of the matrix $|0\rangle \langle 0|$ in the jth term and identity matrices in all other terms. The operational meaning of $p_0^{(j)}$ is the probability to obtain the zero measurement outcome on qubit j for the state $U^\dagger V |\vec{0}\rangle$.

it is proven in [6] that

$$C^{\text{local}} \leq C^{\text{global}} \leq nC^{\text{local}} \tag{8}$$

However, C^{global} suffers from a provably vanishing gradient as n increases, while C^{local} does not. Thus, we will use C^{local} or a linear combination of both costs in our experiments.

[2] In the Khatri paper, the target unitary is denoted as U, while the ansatz is denoted V. I switched the notations to stay consistent with the rest of the examples and papers.

2.2 Optimizers for Variational Quantum Algorithms

Classical optimizers perform the task of finding a (hopefully global) extremum of some scalar function. An optimizer for a function F needs to find the vector $\vec{\theta} \in S$ with $S \subseteq \mathbb{R}^n$ that gives a minimal or maximal value for F. In this project, we will only use optimizers for the minimization of cost functions.

In each step, an optimizer is given a point in the parameter space of the cost function $\vec{\theta}_i$, and is tasked with finding a new guess for parameter values $\vec{\theta}_{i+1}$, which hopefully has a lower cost value.

The optimizer can either use the value of the cost function, its partial derivative (its gradient vector), or its second partial derivatives (its Hessian matrix).

Methods that use the second derivative are often called Newton methods (as Newton's method uses the second derivative). Methods that use *approximations* of the first and second derivative are called Quasi-Newton methods. Methods that use the first derivative are called Gradient-Based methods. Methods that do not use the derivative at all are called Gradient-Free methods.

In this project, I focused on two gradient-free optimizers: the Powell optimizer and the COBYLA optimizer.

2.2.1 Powell

The Powell algorithm [7] is a popular gradient-free optimizer that uses sequential line searches (optimizations over lines in the parameter space).

This method starts with some input set of search vectors $V = \{\vec{v}_i\}$, which are usually chosen to be the coordinate vectors in the parameter space. Each vector defines a search direction (both the direction of the vector and its negation).

Searching along each of those directions in sequence, this method looks for the displacement $\{a_i\}$ along each direction that would minimize the cost when only varying parameters along the current direction. If bounds to the parameter space are specified, the minimization will be under said bounds. If bounds are not provided, then an unbounded line search will be used. Finding the displacements in each direction is done by some univariate gradient-free optimizer. Two common options for such an optimizer are Brent's parabolic interpolation method and the Golden-section search method, but their choice is independent of the general Powell method.

After sweeping through all of the search vectors, two updates are made: First, the current parameters are updated by the rule:

$$\vec{\theta}_{i+1} = \vec{\theta}_i + \sum_i a_i \vec{v}_i \tag{9}$$

Second, the search vector $\vec{v}_j$ that corresponds to the greatest displacement, $a_j = \max(a_i)$ is replaced with

$$\vec{v_j} \to \sum_i a_i \vec{v}_i. \tag{10}$$

The algorithm iterates an arbitrary number of times until no significant improvement to the value of the cost function is made.

2.2.2 COBYLA

Constrained Optimization BY Linear Approximation (COBYLA) is a gradient-free numerical optimization method, invented by Powell [8, 9] (although it is different from the Powell optimization method).

The method tries to optimize a function $F(\vec{\theta})$ according to a set of constraints $\{c_i(\vec{\theta})\}$.

At each iteration, the method has $m+1$ points $\{\vec{\theta}_i\}_{i=0}^{m}$ in the parameter space that are sorted by a combination of the function's value on them and the penalties that the constraints imply on them. The said sorting function is of the form

$$\Phi(\vec{\theta}) = F(\vec{\theta}) + \mu[\max\{-c_i(\vec{\theta}) : i = 1, 2, \ldots, m\}]_+, \vec{\theta} \in \mathbb{R}^n \tag{11}$$

For a value μ that is automatically controlled by the method and the subscript $+$ signifying that 0 is taken in case the value of the expression is negative. Thus, our points are sorted as $\Phi(\vec{\theta}_0) < \Phi(\vec{\theta}_1) < \cdots < \Phi(\vec{\theta}_m)$.

In addition, there is a "trust region" with a radius $\rho > 0$. The method uses the $m+1$ points as the vertices of a simplex (an m-dimensional generalization of a triangle). The values of the cost function in these vertices are used for interpolation to a unique linear polynomial $\hat{F}(\vec{\theta})$, and the values of the constraint functions in these vertices are used for interpolation to unique linear constraints $\{\hat{c}_i\}$ The method then continues by minimizing $\hat{F}(\vec{\theta})$, which is in the trust region — that is, $\frac{\rho}{2} \leq \|\vec{\theta} - \vec{\theta}_0\| \leq \rho$, under the approximated constraints $\{\hat{c}_i\}$.

In case there is no point in the trust region in which all approximated constraints are satisfied, $\vec{\theta}$ is defined by minimizing the greatest of the constraint violations subject to the trust region bound.

The newly calculated point is added to the set of points, and some other point is removed. The choice of the point to be removed is governed mainly by avoiding the degenerate situation where the volume of the simplex collapses to zero, as in that case, only a subspace of the parameter space is reachable.

When the improvements are insufficient, the value of ρ is reduced, until some lower bound, which specifies the termination of the method, is reached.

This explanation of the method is incomplete, and only describes the general principle behind the optimization. In addition, different specifications by Powell (for example, [8] and [9]) define certain requirements and symbols differently, such as using Φ or F for the sorting of vertices, the use of μ, and the use of ρ as the trust radius or as a lower bound for it. Certain technical details and special steps in the method are also omitted and can be found in Section 2 of [8].

3 Barren Plateaus

Informally, a barren plateau is a large section of the parameter space in which the value of the cost function C is "flat", and has no apparent direction in which to move in order to reach a minimum point. The term was first coined by McClean et al. in [2].

When considering a cost function with 2 parameters, this definition coincides with its geometrical intuition. The plane defined by $\{(\theta_1, \theta_2, C(\theta_1, \theta_2))|(\theta_1, \theta_2) \in \mathbb{R}^2\}$ will have large, flat sections, in which the "dips" that represent minima are not indicated by a slope around them (other than a very small neighborhood of the minimum). Such an example can be seen in Figure 1.

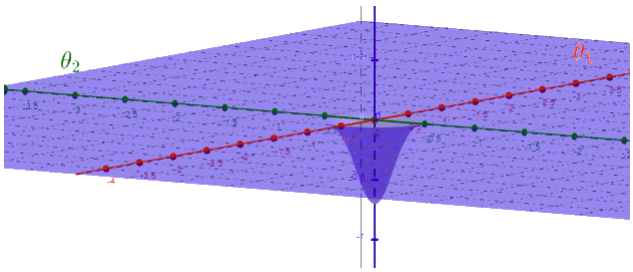

Figure 1: A barren plateau in an inverted bi-variate Gaussian with a small variance.

This is formally defined by the expectation value and variance of the partial derivatives. As defined in [3]:

Definition 1 *Consider the cost function defined in Eq. (3). This cost exhibits a barren plateau if, for all* $\theta_\mu \in \vec{\theta}$*, the expectation value of the cost function partial derivative* $\partial C/\partial\theta_\mu = \partial_\mu C(\vec{\theta})$ *is* $E_{\vec{\theta}}[\partial_\mu C(\vec{\theta})] = 0$ *and its variance vanishes exponentially with the number of qubits* n *as*

$$\mathrm{Var}_{\vec{\theta}}[\partial_\mu C(\vec{\theta})] \leq F(n), \text{ with } F(n) \in O\left(\frac{1}{b^n}\right) \tag{12}$$

for some $b > 1$*. As indicated, the expectation values are taken over the parameters* $\vec{\theta}$*.*

3.1 Ansatzes and Functions Affected By Barren Plateaus

Papers in the field of barren plateaus have shown that many types of cost functions exhibit barren plateaus.

The original work by McClean et al. [2] shows that cost functions of the form in Eq. (3) that use random quantum circuits of polynomial depth and exhibit a property known as a *2-design* suffer from barren plateaus. The definition of a 2-design is beyond the mathematical scope of this project.

However, the work of Arrasmith et al. [3] notes that hardware-efficient layered ansatzes are approximate 2-designs (although they did not show this result themselves). As such, cost functions that use these ansatzes suffer from barren plateaus when the number of layers scales linearly with the number of qubits.

Another paper, by Cerezo et al. [10], shows that if the cost function makes use of global operators, barren plateaus will be encountered in *any* circuit depth. For this reason, it is preferable to use the local cost function in VQC problems and not the global cost function.

3.1.1 Barren Plateaus in VQC

In section 4 of Arrasmith [3], it is shown that even a fairly simple VQC problem suffers from barren plateaus.

The specific problem shown in the paper is concerned with the trivial target unitary $V = I$. Thus, we are trying to optimize the parameters of a parametric quantum circuit $U(\vec{\theta})$ such that $U(\vec{\theta})\,|\vec{0}\rangle = I\,|\vec{0}\rangle = |\vec{0}\rangle$.

The quantum circuit used is a layered hardware-efficient ansatz. Each layer is defined by the following structure:

- A general unitary single-qubit rotation gate $U(\theta_1, \theta_2, \theta_3)$ gate is applied to each qubit.
- A layer of Control-NOT gates is applied between even pairs of qubits (that is, even indices act as controls and odd indices act as targets).
- Another layer of general unitary single-qubit rotations is applied.
- A layer of Control-NOT gates is applied between odd pairs of qubits (odd indices act as controls and even indices act as targets).

A sketch of a single layer of the circuit can be seen in Figure 2.

The cost function used is the local VQC cost function, as shown in Eq. (7). Arrasmith [3] claims that when the number of layers scales linearly with the number of qubits, this problem instance suffers from barren plateaus.

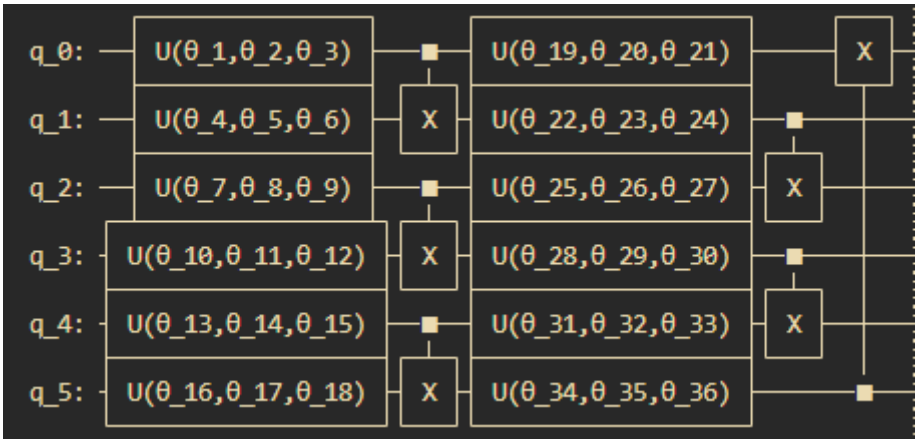

Figure 2: A single layer of the hardware-efficient ansatz used in the project, as designed in Qiskit.

3.2 Barren Plateaus in Gradient-Free Optimization

Seeing as the definition of barren plateaus only discusses the partial derivatives of the cost function, one might assume that using gradient-free optimizers solves this issue. However, the main result of Arrasmith shows that this is incorrect.

The following theorem is shown in [3]:

Theorem 1 *Consider the cost function of Eq. (3). Let $\vec{\theta}_A$ and $\vec{\theta}_B$ be two randomly chosen points in parameter space. Without loss of generality we assume that $\vec{\theta}_B = \vec{\theta}_A + L\hat{l}$ for random L and $\hat{l}$ so that $E_{\vec{\theta}_A,\vec{\theta}_B}[\dots] = E_{\vec{\theta}_A,L,\hat{l}}[\dots]$. If the cost exhibits a barren plateau according to Definition 1, then the expectation value of the difference $\Delta C = C(\vec{\theta}_B) - C(\vec{\theta}_A)$ is $E_{\vec{\theta}_A,L,\hat{l}}[\Delta C] = 0$, and the variance is exponentially vanishing with n as*

$$\mathrm{Var}_{\vec{\theta}_A,L,\hat{l}}[\Delta C] \leq \hat{G}(n) \tag{13}$$

with

$$\hat{G}(n) = m^2 \bar{L}^2 F(n), \text{ and } \hat{G}(n) \in \tilde{O}\left(\frac{1}{b^n}\right). \tag{14}$$

for some $b > 1$. Here m is the dimension of the parameter space, $F(n)$ was defined in (12), and

$$\bar{L} = E_{L,\hat{(l)}}[L] \tag{15}$$

is the average distance between any two points in parameter space.

The effective meaning of this theorem is that the difference in the cost function between different points in the parameter space is exponentially vanishing in n. Thus, even though gradient-free optimizers do not use derivatives, they are still affected — Their main tool for decision-making, which is the difference in the values of the function, gives almost no information.

4 Mutually Unbiased Bases

The vector space that represents a system of n qubits is a 2^n-dimensional Hilbert space. As such, an orthonormal basis for it contains 2^n distinct vectors.

Let $\mathcal{H}$ be a 2^n dimensional Hilbert space, and $\{|\psi_i\rangle\}$, $\{|\phi_i\rangle\}$ be two orthonormal bases for $\mathcal{H}$. We say that $\{|\psi_i\rangle\}$ and $\{|\phi_i\rangle\}$ are *Mutually Unbiased* if the following holds:

$$\forall i, j : |\langle\psi_i|\phi_j\rangle|^2 = \frac{1}{2^n} \tag{16}$$

The intuition for such a definition is that, if a state from one basis is measured in the other, every measurement possibility has equal probability.

References

[1] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644. DOI: 10.1038/s42254-021-00348-9. URL: https://doi.org/10.1038/s42254-021-00348-9.

[2] Jarrod R. McClean et al. "Barren plateaus in quantum neural network training landscapes". en. In: *Nature Communications* 9.1 (Nov. 2018). Number: 1 Publisher: Nature Publishing Group, p. 4812. ISSN: 2041-1723. DOI: 10.1038/s41467-018-07090-4. URL: https://www.nature.com/articles/s41467-018-07090-4 (visited on 07/14/2022).

[3] Andrew Arrasmith et al. "Effect of barren plateaus on gradient-free optimization". en-GB. In: *Quantum* 5 (Oct. 2021). Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, p. 558. DOI: 10.22331/q-2021-10-05-558. URL: https://quantum-journal.org/papers/q-2021-10-05-558/ (visited on 07/13/2022).

[4] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". en. In: *Nature Communications* 5.1 (July 2014). Number: 1 Publisher: Nature Publishing Group, p. 4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213. URL: https://www.nature.com/articles/ncomms5213 (visited on 05/29/2022).

[5] Sumeet Khatri et al. "Quantum-assisted quantum compiling". en. In: *Quantum* 3 (May 2019). arXiv:1807.00800 [quant-ph], p. 140. ISSN: 2521-327X. DOI: 10.22331/q-2019-05-13-140. URL: http://arxiv.org/abs/1807.00800 (visited on 09/25/2022).

[6] Kunal Sharma et al. “Noise resilience of variational quantum compiling”. en. In: *New Journal of Physics* 22.4 (Apr. 2020). Publisher: IOP Publishing, p. 043006. ISSN: 1367-2630. DOI: 10.1088/1367-2630/ab784c. URL: https://doi.org/10.1088/1367-2630/ab784c (visited on 07/14/2022).

[7] M. J. D. Powell. “An efficient method for finding the minimum of a function of several variables without calculating derivatives”. In: *The Computer Journal* 7.2 (Feb. 1964), pp. 155–162. DOI: 10.1093/comjnl/7.2.155. URL: https://doi.org/10.1093/comjnl/7.2.155.

[8] M. J. D. Powell. “A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation”. In: *Advances in Optimization and Numerical Analysis*. Springer Netherlands, 1994, pp. 51–67. DOI: 10.1007/978-94-015-8330-5_4. URL: https://doi.org/10.1007/978-94-015-8330-5_4.

[9] M. J. D. Powell. *A View of Algorithms for Optimization without*. 2007.

[10] M. Cerezo et al. “Cost Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits”. en. In: *Nature Communications* 12.1 (Dec. 2021). arXiv:2001.00550 [quant-ph], p. 1791. ISSN: 2041-1723. DOI: 10.1038/s41467-021-21728-w. URL: http://arxiv.org/abs/2001.00550 (visited on 05/29/2022).