Mutual Unbiased Bases as a Countermeasure to Barren Plateaus in Variational Quantum Algorithms

Ittay Alfassi*

October 4, 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 $\vec{\theta}$.

[1] The plural form of ansatz in German is Ansaetze.

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. 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 $U^\dagger V|\vec{0}\rangle$.

[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.

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.

2.2 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.

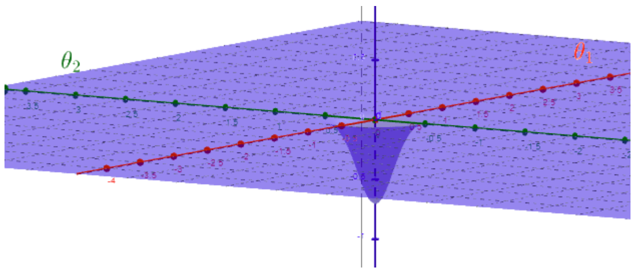

Figure 1: A barren plateau in an inverted bi-variate Gaussian with a small variance.

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.

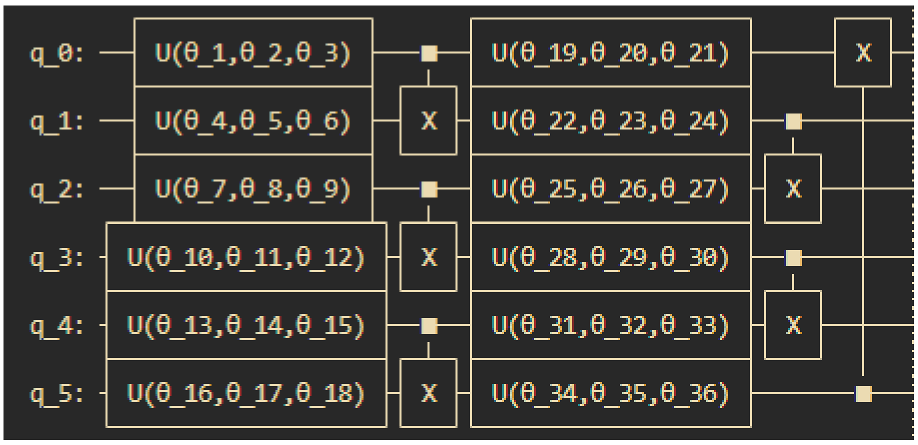

Figure 2: A single layer of the hardware-efficient ansatz used in the project, as designed in Qiskit.

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 $E_{\vec{\theta}_A,\vec{\theta}_B}[\ldots] = E_{\vec{\theta}_A,L,\hat{l}}[\ldots]$. 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 state space of an n-qubit system is a 2^n dimensional Hilbert space. As such, an orthonormal basis for said space contains 2^n distinct vectors. There are infinitely many bases for such a space, but some combinations of bases exhibit special properties. One property, which is the main focus of this project, is *Mutual Unbias.*

The following is a definition for Mutually Unbiased Bases [11]:

Definition 2 *Let* $\mathcal{B}_1 = \{|\phi_i\rangle\}$ *and* $\mathcal{B}_2 = \{|\psi_i\rangle\}$ *be two orthonormal bases in the d dimensional state space. They are said to be* ***mutually unbiased bases (MUB)*** *iff the following holds:*

$$\forall i, j : |\langle\psi_i|\phi_j\rangle| = \frac{1}{\sqrt{d}} \tag{16}$$

A set $\{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$ *of orthonormal bases in* $\mathbb{C}^d$ *is called a* set of mutually unbiased bases *(a set of MUB) if each pair of bases* $\mathcal{B}_i$ *and* $\mathcal{B}_j$ *are mutually unbiased.*

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.

In general, the term “MUB set” refers to a set of bases that are pairwise mutually unbiased. The term “MUB states”, in the context of some MUB set, is the collection of states that comprises the MUBs of that set.

4.1 Generation of Mutually Unbiased Bases

The question of the existence, number, and generation of MUB sets is only answered for certain cases.

The following result has been achieved [11]: For an d-dimensional state space, if d is prime or a prime power (p^k for a prime p and a natural number k), there exists a set of $d + 1$ MUBs. There is constructive proof for this result. Because the dimension of an

n-qubit state space is 2^n (a prime power), there is a construction for $2^n + 1$ MUBs for every n.

It is important to note, of course, that once a MUB set is found, there is an infinite amount of possibilities for MUB sets of the same size. Applying any arbitrary unitary U to all states of the set of bases will preserve the constraints on the inner products of every pair of states, which gives a new MUB set (assuming U did not act as a permutation on the set).

Connections between the problem of finding MUB sets and other problems have been made. For example, Lawrence et al. [12] showed that the set of $4^n - 1$ Pauli operators (in the n-qubit case) can be partitioned into $2^n + 1$ distinct subsets, each consisting of $2^n - 1$ commuting operators. The same work shows two interesting conclusions:

- The set of eigenbases that mutually diagonalize each subset is a set of MUBs. That is, if each subset $\mathcal{F}_i$ is mutually diagonalized by an orthonormal basis $\mathcal{B}_i$, then the set $\{\mathcal{B}_i\}$ is a MUB set.
- Each such partitioning defined a unique choice of 2^n+1 MUBs in the n-qubit Hilbert space.

I used the examples given in the work of Lawrence et al. to construct a set of MUBs for 2 and 3 qubits. However, it is important to note that the generation of MUBs in this fashion requires exponential time in the number of qubits: it requires the simultaneous diagonalization of matrices that are exponential in size.

4.2 MUB States as an Exhaustive Search

The set of MUB states for one qubit is well known. The set contains 3 MUBs, and each basis is the eigenbasis of a Pauli matrix. Specifically, the bases and states are

$$\mathcal{B}_z = \{|0\rangle, |1\rangle\} = \{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\}$$

$$\mathcal{B}_x = \{|+\rangle, |-\rangle\} = \{\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}, \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}\}$$

$$\mathcal{B}_y = \{|+_i\rangle, |-_i\rangle\} = \{\frac{1}{\sqrt{2}}\begin{pmatrix}1\\i\end{pmatrix}, \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-i\end{pmatrix}\}$$

When displayed on the Bloch sphere, these states are presented as the poles of the sphere in the z, x, and y axes, appropriately. As such, every 1-qubit state is reachable through a linear combination of these six states with real coefficients.

This is our initial intuition: these states reach every pole of the sphere that defines all 1-qubit states. We (Tal Mor and I) believed that this can be extended to higher dimensions. If our hypothesis of the importance of MUBs is correct, they can be useful when trying to check for a certain property, **or when searching for a specific point in space.**

An observation made in [12] supports our hypothesis. In [12], a MUB is generated by being the orthonormal basis that mutually diagonalizes a subset of Pauli operators. Let $\{O_i\}$ be such a subset of Pauli operators, and let $\{|\psi_i\rangle\}$ be the mutually diagonalizing eigenbasis calculated for that subset. Let $U_i(\theta) = exp(-i\theta O_i)$ be a unitary high-dimensional rotation defined by O_i. When U_i acts on a MUB state $|\psi_j\rangle$, that state is not affected (up to a global phase) **and acts as a pole**.

In the next section, I will discuss the different methods to use the concept of MUB states as an exhaustive search in mitigating barren plateaus.

5 MUB Utilization Techniques

This section specifies the novel ideas that Tal and I had for the project. As such, they are comprised of theoretic extensions and assumptions that we made.

To explain our idea, let us consider a VQA in which the target is to reach a specific state. One such example is VQE (section 2.1.1), in which the lowest eigenvalue is calculated by reaching the lowest eigenstate. An example of utilizing MUBs in a different type of VQA is given in subsection 5.4.

Let us make the following assumptions:

1. The cost function $C(\vec{\theta})$ (shown in Eq. (4)) experiences a barren plateau in the parameter space.
2. The optimization starts with a specific initial parameter vector, $\vec{\theta}_0$, whose entries were chosen uniformly randomly from the parameter space.
3. The ansatz $U(\theta)$ is *expressive* — Every state $|\psi\rangle$ in the state has a parameter vector $\vec{\theta}_\psi$ such that $U(\vec{\theta}_\psi)\,|\vec{0}\rangle = |\psi\rangle$.

The target state $|\psi^*\rangle$ that the VQA is trying to reach, according to assumption 3, has an appropriate parameter choice $\vec{\theta}^*$. However, since $C(\vec{\theta})$ suffers from barren plateaus, any optimizer cannot travel long distances in the parameter space. Thus, if $U(\vec{\theta}_0)\,|\vec{0}\rangle$ is far from $|\psi^*\rangle$ (in terms of the state space), the VQA will not reach the required parameters.

However, because the MUB states act as an exhaustive search over the Hilbert space (or so we hope), there exists some MUB state $|\alpha\rangle$ such that $U(\vec{\theta}_0)\,|\alpha\rangle$ and $|\psi^*\rangle$ are close (in terms of the state space). Equivalently, there exists some parameter vector $\vec{\theta}_0^\alpha$ such that $U(\vec{\theta}_0)\,|\alpha\rangle = U(\vec{\theta}_0^\alpha)\,|\vec{0}\rangle$, and $\vec{\theta}_0^\alpha$ and $\vec{\theta}^*$ are close **in terms of the parameter space**. Thus, an optimizer will be able to find an appropriate parameter set and reach $|\psi^*\rangle$ despite the barren plateaus of the cost function.

Non-Uniformity of Parameter Space
An ansatz, even if fully expressive, does not necessarily bring its input to all output states in the Hilbert space with equal probability. For example, it could be that the vast majority of parameter vectors bring the result to a very small distance from the input vector. For this reason, the use of a MUB state to dramatically shift the output result is valuable. This phenomenon is connected to the mathematical term called *Concentration of Measure* and to the Haar Measure for unitaries and states, as mentioned in [2]. However, the research of these terms and phenomena were left outside the scope of this report.

Non-Expressive Ansatzes
While assumption 3 seems useful (both for applications and for our analysis), it is unfortunately unrealistic. Seeing as a general $2^n \times 2^n$ unitary matrix has an exponential number of degrees of freedom, reaching the entire set of unitary transformations requires an exponential number of parameters and an ansatz of exponential size.

Thus, the output of a hardware-efficient ansatz of linear depth will only span a small subset of the Hilbert space. If our target is a state $|\psi^*\rangle$, it could lie outside the subset $\{U(\vec{\theta})\,|\vec{0}\rangle \,|\vec{\theta} \in \mathbb{R}^m\}$. Using a MUB state as the input instead of $|\vec{0}\rangle$ could dramatically shift the subset and change it to include $|\psi^*\rangle$.

The question that remains is how *exactly* to use the set of MUB states.

5.1 Using All MUB states

The most trivial idea is to attempt optimization from all possible MUB states. That is, set a specific random initial parameter vector $\vec{\theta}_0$, and run a separate experiment with every MUB state as the initial state.

However, there is a major drawback to this method. For n qubits, there are $2^n + 1$ different MUBs, with 2^n states each. Thus, this requires performing $O(4^n)$ experiments for every instance of the problem, which is non-scalable for reasonable problem sizes.

5.2 Sampling k MUB states

A way to mitigate the drawback I just mentioned is to randomly sample a small number of MUB states from the entire set and only run experiments from them.

This will allow an arbitrary amount of experiments, which we can control. In addition, as will be discussed in section 6.1, this serves as an interesting benchmark to compare against naïve methods.

However, as mentioned in subsection 4.1, the cost of calculation of the MUBs and their states is still exponential in time. Although the work of MUB calculation only needs to happen once and can be considered pre-processing, this makes the current suggestion unfeasible, for example, for $n > 60$ qubits.

In addition, although the work of [11] shows a theoretical construction for the MUB sets for n qubits, it is difficult to implement in practice. Thus, Tal and I faced an interesting question: what can be done, for n qubits, with only the MUB states for 2 and 3 qubits?

5.3 "Half-MUB" states

We came up with a relatively simple idea: given an n-qubit register, pick 2 or 3 qubits. Generate a MUB state between the chosen qubits, and keep the rest in the 0 state.

The amount of such states for n qubits is polynomial. There are $\binom{n}{3} = O(n^3)$ triplets for the qubit choices, and each one has a constant amount of MUB state options. Thus, their generation (and even exhaustive experimentation with them) takes reasonable time.

This method got the nickname "Half-MUBs", as its results resemble MUB states but do not give the same exhaustive properties. However, no one guaranteed that MUB states act as an exhaustive search themselves, so what the hell.

5.4 MUB Utilization in VQC

As explained in subsection 2.1.2, the goal of VQC is to make the ansatz act like a specific target unitary (either in general or on a specific state). In this sense, it deviates from the previous discussions in this section, as it does not aim to reach a specific state, but a specific transformation. However, the claims presented in this section still apply in the case of VQC.

The only difference is that instead of considering the state Hilbert space, we look at the group of unitary transformations on n qubits $U(2^n)$. Non-Uniformity still applies: a uniform choice of parameter values for the ansatz does not necessarily send the resulting unitary to a uniformly random unitary in $U(2^n)$. Because of that, the prepending of the MUB generation circuit to the ansatz could move it dramatically closer to the target unitary (in terms of the parameter space). Non-expressivity applies in a similar fashion.

6 Experiments

In this section, I will detail the experiments that I implemented in this project.

Section 6.1 explains the control group to which I compare each idea.

Section 6.2 details the hyper-parameters of every experiment.

Section 6.3 details the experiments used for picking hyper-parameters.

Section 6.4 shows a recreation of the barren plateau simulation presented in of [3].

Section 4.1 details the synthesis of MUB-generating circuits.

Section 6.6 shows experiments on 3 qubits and k random MUB states.

Section 6.7 shows experiments on higher numbers of qubits and k half-MUB states.

All experiments were implemented in Python, and can be found on GitHub under the link `https://github.com/ittay-alfassi/barren-plateaus`. MUB circuit synthesis was done using the Qsymm [13] and BQSkit [14] packages. The construction and execution of quantum circuits was done using the Qiskit [15] package. The optimizers were taken from the SciPy package [16].

6.1 Control Group: Random Initialization Vectors

A critical step that we must take before testing the use of MUBs is a naïve equivalent. A reasonable equivalent to using k different MUB (or half-MUB) states and a single initial vector is to run k experiments with no MUB states involved — just use a different random initial vector for each experiment.

Note that if our ansatz is expressive (assumption 3 in section 5), another interesting comparison can be made.

As shown in section 5, if an ansatz is expressive, every MUB state $|\alpha\rangle$ can be described as $U(\vec{\theta}^{\alpha})\,|\vec{0}\rangle$ for some parameter vector $\vec{\theta}^{\alpha}$. Thus, the comparison between k MUB states and k random initial vectors is equivalent to a comparison between a carefully chosen set of starting parameters $\{\vec{\theta}^{\alpha} |\, |\alpha\rangle \text{ is a MUB state}\}$ and a random set of starting parameters $\{\vec{\theta}_0^i | \forall i, \vec{\theta}_0^i \in [-\pi, \pi)^m\}$.

6.2 Experiment Hyper-Parameters

Every VQA experiment has, besides of the initial state and the given problem, other parameters that affect its execution. These parameters are often called hyper-parameters (to separate them from the parameter vector of the ansatz and optimizer). These hyper-parameters can have a deep impact on the result of the experiment. Thus, it is important to list them and explain the choice of every hyper-parameter.

Optimizer The same optimization problem can be solved by different optimizers. The optimizers that I used in my experiments are the gradient-free optimizers COBYLA and Powell.

Tolerance Condition Every optimizer has certain halting conditions. For example, an upper bound on the number of optimization iterations can be defined. To allow for long optimization times, I did not place a max-iteration limit.

Another halting condition is the tolerance condition. It is a condition that defines a minimal tolerance for differences in the optimization. For example, an optimizer can require a minimum difference between two different iterations' cost values. Another example is a lower bound for parameters that usually decrease in the optimization procedure, such as the trust region in COBYLA.

In general, a tolerance condition is defined as a floating point number. Different optimizers give different meaning to the tolerance value. I chose 4 different values to experiment with: 2×10^{-1}, 10^{-2}, 10^{-5}, and 10^{-12}. They are used in different settings.

Success Bound Another possible halting condition for an optimizer is a manual limit. For example, in the numerical simulations of Arrasmith [3], they stopped the optimization process once the cost function's value reached $C = 0.4$. I call these values "success bounds", as they present a lower bound for the function's value, such that reaching it (or below it) counts as a success of the optimization procedure. Specifically, it signifies that we succeeded in overcoming the barren plateaus.

In my experiments, I define two different success bounds: 0.4 and 0.1. Both are used in different settings.

Number of Repetitions Every experiment must be repeated several times to ensure the statistical significance of the results. In our case, the value k (the number of MUB states, Half-MUB states, or random vectors) fits this role well. Performing five experiments with $k = 5$ in each one, then generating statistics for the collection of results is equivalent to performing a single experiment with $k = 25$. I chose to use $k = 25$ in all of my experiments.

6.3 Perliminary Experiments: Setting Hyper-Parameters

Choosing the Tolerance Condition In SciPy, the tolerance condition of all optimizers is defined by a single floating-point parameter named "tol". However, its practical meaning is different for every optimizer.

For the COBYLA optimizer, the tol value is a lower bound on the size of the trust region. For the Powell optimizer, the tol value becomes (in the words of the SciPy documentation) the relative error in the solution and its function value acceptable for convergence. The documentation gives no further details on the meaning of "relative error". For the Nelder-Mead optimizer, the tol value has the same definition as the Powell optimizer.

In order to see what tol values are appropriate for every optimizer, I tried each optimizer with three different tol values. The task was the same: Solve the trivial VQC example with 3 qubits, 12 layers of the hardware-efficient ansatz, and a success bound of 0.1.

6.4 Experiment 1: Reproducing the Arrasmith Barren Plateau

In section 4 (Numerical Simulations) of Arrasmith [3], a numerical demonstration of the barren plateaus in the VQC problem. In that section, the trivial VQC problem mentioned in section 3.1.1 is tested for $n = 5, \ldots, 11$ qubits. The hyper-parameters were defined as the following:

- **Optimizer.** Four different optimizers were tested: Nelder-Mead, Powell, COBYLA, and Gradient Descent (for reference).
- **Success Bound.** The success bound was defined as a cost value of $C = 0.4$.
- **Number of repetitions.** Each experiment was performed $k = 20$ times.

6.5 Synthesis of MUBs in Code

6.6 Experiment 2: 3 qubits and n layers

6.7 Experiment 3: n qubits and n layers

7 Discussion

8 Future Work

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).

[11] Somshubhro Bandyopadhyay et al. "A New Proof for the Existence of Mutually Unbiased Bases". en. In: *Algorithmica* 34.4 (Nov. 2002), pp. 512–528. ISSN: 0178-4617, 1432-0541. DOI: 10.1007/s00453-002-0980-7. URL: http://link.springer.com/10.1007/s00453-002-0980-7 (visited on 09/29/2022).

[12] Jay Lawrence, Časlav Brukner, and Anton Zeilinger. "Mutually unbiased binary observable sets on N qubits". In: *Physical Review A* 65.3 (Feb. 2002). Publisher: American Physical Society, p. 032320. DOI: 10.1103/PhysRevA.65.032320. URL: https://link.aps.org/doi/10.1103/PhysRevA.65.032320 (visited on 06/23/2022).

[13] Dániel Varjas, Tómas Ö Rosdahl, and Anton R Akhmerov. "Qsymm: algorithmic symmetry finding and symmetric Hamiltonian generation". In: *New Journal of Physics* 20.9 (Sept. 2018), p. 093026. DOI: 10.1088/1367-2630/aadf67. URL: https://doi.org/10.1088%2F1367-2630%2Faadf67.

[14] Ed Younis et al. *Berkeley Quantum Synthesis Toolkit (BQSKit) v1*. Apr. 2021. DOI: 10.11578/dc.20210603.2. URL: https://www.osti.gov/biblio/1785933.

[15] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[16] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.