IMPLEMENTING AND CHARACTERIZING A QUANTUM DIVIDE AND CONQUER VARIATIONAL QUANTUM ALGORITHM

by

Alexander Gray

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Applied Physics

Charlotte

2022

Approved by:

_____
Dr. Donald Jacobs

_____
Dr. Tino Hofmann

_____
Dr. Yong Zhang

ABSTRACT

ALEXANDER GRAY. Implementing and Characterizing a Quantum Divide and Conquer Variational Quantum Algorithm. (Under the direction of DR. DONALD JACOBS)

The current landscape of quantum computing devices consists of Noisy Intermediate-Scale Quantum (NISQ) devices, which contain around 100 or less qubits and are not fault-tolerant or error correcting. Due to the physical limitations of these devices, algorithms must be developed with short circuit depths and which are stable to the noise inherent to quantum measurements. A broad class of quantum computing algorithms called Variational Quantum Algorithms (VQAs), which leverage a classical computer for parameter optimization of a problem-related cost function, fit these criteria and serve as the basis for many NISQ-era algorithms. In this work, we develop a VQA based on a newly discovered set of operators called Divide And Conquer Operators (DACOs), which is termed the DACO-VQA. Using these operators, unitary parameter-dependent quantum gates can be constructed which, when consecutively applied on an initial Hadamard state, continually cut the Hilbert space in half, leading to a single pure state bit string in the end. This property is leveraged in the DACO-VQA to restrict the search to consecutive halves of the problem-Hilbert space, reducing the range of measurement sampling. The DACO-VQA also utilizes a cost function based on a partition function of the empirically measured energies of the states generated by the quantum computer. In this work we discuss the development of the DACO-VQA structure and its operators, as well as completeness of the operator pool chosen for the DACO-VQA, entangling capability of the quantum circuits utilized, and benchmark the performance of the algorithm for certain problem-Hamiltonians.

ACKNOWLEDGEMENTS

I would like to thank the members of my committee for their patience and flexibility throughout the process of this thesis. In particular, I am grateful to Drs. Tino Hofmann and Yong Zhang for their understanding and willingness to adapt to the roadblocks which have appeared along the way. Most of all, I want to thank Dr. Donald Jacobs for his dedication to ensuring my success throughout my time at UNC Charlotte. Without the countless hours Dr. Jacobs has spent with me over the years, whether discussing the wonders of physics or sharing anecdotes about research and life, I would not be where I am today.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DACO Divide And Conquer Operator.

DACO-VQA Divide And Conquer Operator-Variational Quantum Algorithm.

NISQ Noisy Intermediate-Scale Quantum.

QEC Quantum Error Correcting.

qubit quantum bit.

VQA Variational Quantum Algorithm.

CHAPTER 1: INTRODUCTION

Quantum computing promises a massive improvement in the computational power of computers, including the ability to solve classically intractable problems [1]. Many algorithmic and experimental developments have been made in pursuit of realizing this computational advantage, however, at present there are numerous experimental difficulties limiting the practical application of many quantum computing algorithms. In the near-term, only the so-called Noisy Intermediate-Scale Quantum (NISQ) [2] devices can be practically utilized. NISQ devices typically contain less than 100 qubits and consist of operations which are not Quantum Error Correcting (QEC) [3, 4]. In addition, it is estimated that it will take decades to develop the hardware necessary to implement fault tolerant QEC algorithms, hence, in order for quantum computing to become useful presently, algorithms with short circuit-depths which are sufficiently stable to noise must be developed. One promising candidate has been Variational Quantum Algorithms (VQAs) – hybrid quantum-classical algorithms which leverage a classical computer for parameter optimization – which are discussed in detail in section 1.2 [5]. The focus of this work was to develop a VQA utilizing newly found Divide And Conquer Operators (DACOs), which have several appealing properties for efficient implementation in a VQA. The algorithm utilizing these operators was termed the Divide And Conquer Operator-Variational Quantum Algorithm (DACO-VQA). The aim of this report is to detail the intellectual development of these operators and the VQA structures which were constructed from them, as well as characterizing the properties which make them appealing. We begin with a fundamental discussion of quantum computing algorithms [6].

1.1 Quantum Computing and Quantum Algorithms

The essential difference between classical and quantum computers is their fundamental unit of information. Classical computers utilize bits, which have two possible states: 0 or 1. Quantum computers utilize quantum-bits (qubits), which are quantum states which can be in an arbitrary linear combination of two states: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$. For example, for a single 1 half spin particle, the two states can be the spin-down state $|\downarrow\rangle \to |0\rangle$ and the spin-up state $|\uparrow\rangle \to |1\rangle$. An arbitrary state of this system is a linear combination of the two orthogonal basis states, and hence furnishes a qubit. Upon measurement, the qubit obtains one of the two classical bit values $|0\rangle$ or $|1\rangle$, however, through utilizing superposition and entanglement, quantum computers are able to perform computations impossible with classical computers. To do so, they utilize gates which are composed of quantum operators. In order to simplify the notation of quantum computing gates, a "computational basis" is adopted, where $|0\rangle \equiv [1,0]^{\mathrm{T}}$ and $|1\rangle \equiv [0,1]^{\mathrm{T}}$, so that $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = [\alpha,\beta]^{\mathrm{T}}$. As an example of a quantum computing gate, we can construct the classically-familiar NOT gate X as:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{1.1}$$

so that

$$X\,|\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \implies X\,|0\rangle = |1\rangle\,. \tag{1.2}$$

Other single-qubit operators can be constructed similarly, and in the computational basis they can all be represented by unitary 2×2 matrices.

We can extend to multiple qubits by taking tensor products of single qubit states. For example, we can denote the 3-qubit state $|ijk\rangle$ where $i,j,k \in \{0,1\}$, as $|ijk\rangle \equiv |i\rangle \otimes |j\rangle \otimes |k\rangle$. For a 2 qubit system, we can label the states in the computational

basis as follows:

$$|00\rangle \equiv \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ |01\rangle \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \ |10\rangle \equiv \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \ |11\rangle \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{1.3}$$

One of the most important multi-qubit gates is the controlled-NOT (CNOT) gate, which targets the second qubit with a NOT gate when the first qubit is 1, and leaves the second qubit unchanged when the first qubit is 0:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.4}$$

The action of this gate on each of the two-qubit states is $\text{CNOT}\,|00\rangle = |00\rangle$, $\text{CNOT}\,|01\rangle = |01\rangle$, $\text{CNOT} = |10\rangle = |11\rangle$, and $\text{CNOT}\,|11\rangle = |10\rangle$. Any multi-qubit gate can be represented as some combination of single-qubit gates and CNOT gates.

Now, to build a quantum computing algorithm, multiple gates are combined in sequence, through which an initial state is passed to produce a desired final state. Throughout the propagation of the state through the quantum computer, measurements of the quantum state may also be taken. This combination of gates and measurements forms a quantum circuit. As a trivial example of a quantum circuit, we can take the initial state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ and pass it through a NOT gate to obtain $X\,|\psi\rangle = \beta\,|0\rangle + \alpha\,|1\rangle$. Then, following this, perform a measurement on the state collapsing it to either $|0\rangle$ or $|1\rangle$ with probability $|\beta|^2$ and $|\alpha|^2$ respectively. While this example is simple, all quantum circuits follow this general structure, and complexity can be added by including parameter dependencies, interactions between multiple

states, and more intricate quantum gates.

Now that the basic ideas of quantum computing have been developed, along with quantum circuits and their mathematical representation, we can move on to describing the quantum circuits of interest to this work: variational quantum algorithms.

1.2 Variational Quantum Algorithms

Variational quantum algorithms are a type of quantum computing algorithm which utilize a quantum computer to generate a parameter-dependent state and to estimate the value of a cost function, and a classical computer for parameter optimization of the cost function. They are particularly promising in the realm of NISQ devices, as they can leverage the power of a classical optimizer to limit the number of qubits required. VQA methods involve a cycle of state-propagation, cost function measurement, and parameter optimization. Once optimized, a new state is generated with the previously found parameter values and the process is repeated until the desired final state is obtained. This process is illustrated in Fig. 1.1. Since each round is influenced by the parameter optimization of previous rounds, these methods are learning-based, making them ideal for *hardware-efficient* [7] quantum algorithms, which are general algorithms applicable to a wide range of problems and are thus not *problem-specific* (more detailed definitions for these terms will be provided in the following sections). A wide variety of VQAs have been developed for various applications, ranging from solving combinatorial optimization problems to simulating quantum mechanical systems. VQAs' advantages in providing NISQ-era quantum computing algorithms, as well as their demonstrated success in solving real problems, makes them ideal candidates for developing near term quantum computing algorithms. Every VQA consists of three main components: the ansatz (a parameter-dependent initial state of the quantum computer), cost function (a criterion related to the problem to be solved), and classical optimizer (utilized to optimize the cost function over the parameter space). The next sections will explain each of these components in detail.

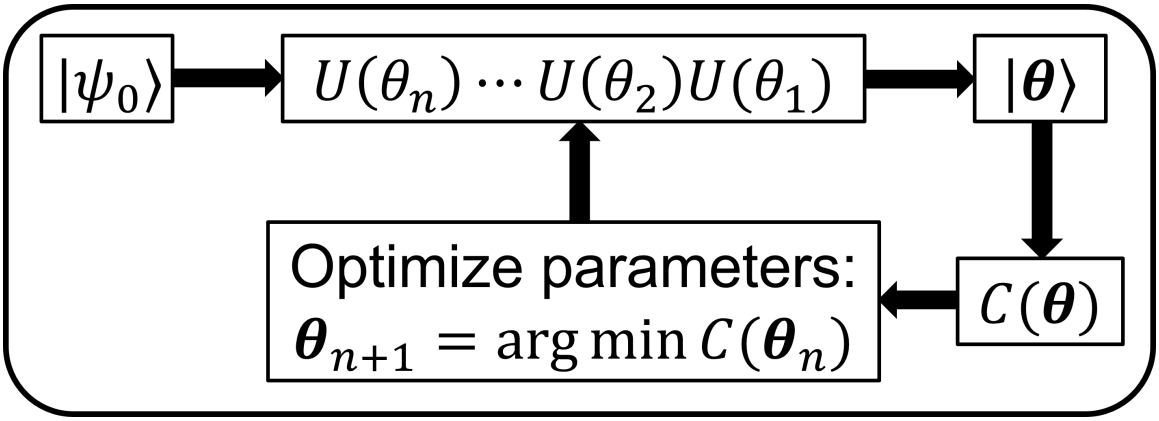

Figure 1.1: Diagrammatic representation of the VQA process. Here, $|\psi_0\rangle$ represents the initial parameter-independent state, which is passed through the unitary gates $U(\theta_i)$ to generate the ansatz $|\boldsymbol{\theta}\rangle$. Then, the cost function $C(\boldsymbol{\theta})$ is calculated and optimized over, and the process is repeated.

1.2.1 Ansatz

The ansatz is the parameter-dependent initial state of the VQA. Typically, the ansatz is generated by acting on some parameter-independent initial state with a series of parameter-dependent unitary gates. Often, the initial parameter-independent state is a Hadamard state, which is a linear combination of all multi-qubit states in the computational basis with each basis state weighted equally:

$$|s\rangle \to |H\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle \,, \tag{1.5}$$

where the sum is over all bit strings z (computational basis states), n is the number of qubits, 2^n is the total number of distinct bit strings, $|s\rangle$ is the standard notation for an initial parameter-independent state, and $|H\rangle$ denotes the Hadamard state. To generate the ansatz, this initial state is passed through parameter-dependent operators:

$$|\boldsymbol{\theta}\rangle \equiv U_N(\theta_N)\cdots U_2(\theta_2)U_1(\theta_1)\,|s\rangle \,. \tag{1.6}$$

where the state $|\boldsymbol{\theta}\rangle$ is the ansatz. The choice of ansatz, and thus the choice of opera-

tors $U(\boldsymbol{\theta})$, depends on the VQA and the problem to be solved. Ansätze which contain operators independent of the problem to be solved are called *hardware-efficient* ansätze. Since these ansätze are problem-independent, they are less resource intensive on the quantum computer, have shallower circuit depth, and are applicable to a wider range of problems. Ansätze which contain operators derived from the problem to be solved (such as those utilizing the problem-Hamiltonian) are called *problem-specific* ansätze. These typically incorporate the cost function in some way, and thus depend on the specific problem to be solved [8, 9]. For example, in solving for the electronic states of small molecules, fermionic creation and annihilation operators can be built into the ansatz, making it problem-specific. Since these ansätze can typically only be used in the context of the problem they were designed for, they are less widely applicable. They are, however, very good at solving the problems they are designed for, making them advantageous in some contexts. There are numerous ways to generate both problem-specific and hardware-efficient ansätze, and both were explored in this work.

One important example of an ansatz relevant to this work is the quantum alternating operator ansatz. This ansatz structure was first developed in the context of solving combinatorial optimization problems in an algorithm called the Quantum Approximate Optimization Algorithm (QAOA) [10]. Both the structure of this ansatz and the specific operators utilized in QAOA are relevant to this work, as they inspired the structure of the DACO-VQA as well as its operator pool. We will now dicsuss QAOA in detail, and describe its connection to this thesis.

QAOA is an algorithm which was developed for solving combinatorial optimization problems by mapping the combinatorial problem into an objective function

$$C(z) = \sum_{\alpha=1}^{m} C_\alpha(z), \tag{1.7}$$

where z is a bit string and C_α is a localized objective function with $C_\alpha(z) = 1$ if the

bit string z satisfies some condition α, and $C_\alpha(z) = 0$ otherwise. The optimization procedure involves getting C as close to its maximum as possible (thus satisfying the most conditions α). With $C(z)$ a diagonal operator in the computational basis, we define the unitary operators

$$U(C, \gamma) = e^{-i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}. \tag{1.8}$$

Since the operators are diagonal in the computational basis, they commute with one another, and the sum in the exponential can be written as a product of exponential factors as shown. Next, we define the operator B as the sum of all single-qubit X operators (with the notation X_j indicating that the j-th qubit is targeted with a single-qubit X gate):

$$B = \sum_{j=1}^{n} X_j, \tag{1.9}$$

from which the unitary operator $U(B, \beta)$ is defined as

$$U(B, \beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta X_j}. \tag{1.10}$$

The initial parameter-independent state used to construct the ansatz is the Hadamard state $|H\rangle$ described above. We define $2p$ parameters by $\gamma_1, ..., \gamma_p \equiv \boldsymbol{\gamma}$ and $\beta_1, ..., \beta_p \equiv \boldsymbol{\beta}$. Then, the ansatz is constructed as

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U(B, \beta_p)U(C, \gamma_p) \cdots U(B, \beta_1)U(C, \gamma_1) |H\rangle . \tag{1.11}$$

With the ansatz and objective function defined, we can construct the cost function as

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| C |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle , \tag{1.12}$$

where the updated parameters are obtained from

$$M_p \equiv \max_{\boldsymbol{\gamma},\boldsymbol{\beta}} F_p(\boldsymbol{\gamma},\boldsymbol{\beta}), \tag{1.13}$$

such that

$$(\boldsymbol{\gamma}_{n+1},\boldsymbol{\beta}_{n+1}) = \arg M_p(\boldsymbol{\gamma}_n,\boldsymbol{\beta}_n). \tag{1.14}$$

As the integer $p \to \infty$, $M_p \to \max_z C(z)$, and so F_p furnishes a good cost function for these types of problems. Note that since the ansatz depends on the objective function $C(z)$, it is a problem-specific ansatz. This algorithm works very well for specific combinatorial optimization problems and does well in reducing circuit depth, but in general has problems with over-sampling the Hilbert space. However, this method of ansatz construction can be generalized with different operators replacing X_j and thus B and $U(B,\boldsymbol{\beta})$, as will be seen with the DACO constructions [11]. In the broader context of quantum alternating operator ansätze, the operator B is known as the mixing Hamiltonian, as it has nothing to do with the specific problem to be solved, and serves only to mix the problem Hamiltonian (in the case of QAOA, $C(z)$) throughout the ansatz. The original inspiration for the operator pool used in the DACO-VQA was finding a better set of operators to serve as the mixing Hamiltonian.

1.2.2 Cost Function

The cost function encodes the problem to be solved into the quantum computer as a function of the tunable parameters $\boldsymbol{\theta}$, and is denoted by $C(\boldsymbol{\theta})$. In general, the cost function is a function of input states $|\psi_i\rangle$, observables (measurements) of those states O_i, and the parameter-dependent gates $U(\boldsymbol{\theta})$:

$$C(\boldsymbol{\theta}) = f(\{|\psi_i\rangle\},\{O_i\},U(\boldsymbol{\theta})). \tag{1.15}$$

In many contexts, problems can be mapped into a Hamiltonian H, and the cost function is taken as the expectation value of the Hamiltonian with respect to the ansatz:

$$C(\boldsymbol{\theta}) = \langle \boldsymbol{\theta} | H | \boldsymbol{\theta} \rangle . \tag{1.16}$$

Typically for cost functions of this form, the solution to the problem is represented by the ground state of the system. Thus, the classical optimizer finds minima of this expectation value, generates a new ansatz with the updated parameter values, then repeats the process until some threshold criterion is met. Some cost functions utilize gradients of this expectation value within the quantum computer, rather than solely relying on the classical optimizer for finding minima.

For cost functions of the form of Eq. 1.16, the process of calculating the cost function for a given set of parameters in a physical quantum computer involves actually measuring the state quantum mechanically. As such, there is inherent uncertainty built into its evaluation. Dealing with this inherent uncertainty is at the forefront of quantum computing research, and is termed the *measurement problem* [12]. Despite this, many recently developed quantum computing algorithms assume infinite precision in the evaluation of these expectation value cost functions. However, since this work is focused on developing practical algorithms for the NISQ-era of quantum computing, which does not assume infinitely precise, fault-tolerant quantum computers, all cost function measurements will be realistically simulated, and theoretical averaging will not be employed.

1.2.3 Optimizer

The final component of a VQA is the classical optimizer which is used to tune the parameters $\boldsymbol{\theta}$. There are two classes of classical optimizers, those which involve gradient descent, and those which do not. The choice of classical optimizer usually depends

on the problem being solved by the quantum computer, as different optimizers have advantages over others in specific contexts. This idea is reinforced by the *no free lunch* theorems [13, 14], which state that no optimizer has an advantage over any other when averaged over all possible problems (at least in the context of combinatorial optimization). Early in this work, several classical optimizers were identified for potential implementation in the DACO-VQA structure [15, 16, 17, 18, 19, 20, 21, 22, 23]. Ultimately, however, we developed our own classical optimizer based on random-walk diffusion processes.

It is worth noting that the selection of a classical optimizer typically depends on the cost function landscape. Landscapes which have many local minima scattered about do not lend well to gradient-descent optimizers, as they have a tendency to get stuck in local minima. As such, gradient free methods are generally preferable in hardware-efficient algorithms, as the cost function landscape varies from problem to problem. However, if a problem-specific algorithm is utilized, then there is a possibility that an ansatz can be generated which acts as a funnel with a single global minimum. In this case, gradient-descent optimizers may prove faster [24]. Evaluating gradients in a quantum computer also amplifies noise from measurements, since taking the gradient of a noisy function increases the noise. In the context of the DACO-VQA, numerous numerical simulations have shown no significant changes to the cost function landscape based on adding problem-specific ansatz gates, so our gradient-free diffusion optimizer has proved sufficiently effective independent of the ansatz structure.

1.3 Operator Pool Completeness

An important feature of many quantum computing algorithms is operator pool completeness. This is a property that ensures any state in the problem-Hilbert space can be accessed by some combination of operators in the pool. This property is important since the operators must be able to reach the ground state, a state which

is not known a priori, and thus is an arbitrary state in the Hilbert space. The notion of completeness was first introduced in the context of ADAPT-VQE, an adaptive VQA which iteratively builds the ansatz one operator at a time [25], [26].

Given some operator pool, we want to characterize whether the operators can reach any state in the Hilbert space. This is done by ensuring that the vectors generated by the operators $\{A_i\}$ and their commutators $\{C_i\}$ form a complete basis, with the vectors taking the form $C_i |\psi\rangle$. We define the overlap matrix

$$M_{ij} = \langle\psi| C_i^\dagger C_j |\psi\rangle , \tag{1.17}$$

with $|\psi\rangle$ an arbitrary real state. If the rank of M_{ij}, satisfies $r(M) \geq 2^n - 1$, where 2^n is the size of the Hilbert space, the operator pool is complete. The rank must be at least $2^n - 1$ rather than 2^n since one of the parameters is fixed by normalization, so that there are only 2^n "free" parameters. The reason for including commutators of the original pool is due to the Baker-Campbell-Hausdorff formula:

$$e^X e^Y = e^Z \implies Z = X + Y + \frac{1}{2}[X,Y] + \frac{1}{12}([X,[X,Y]] - [Y,[X,Y]]) + \cdots . \tag{1.18}$$

Since the ansatz consists of multiple applications of unitary operators of the form $e^{-i\theta A}$, the states implicitly include the commutators in the exponent. It should also be noted that in the overlap matrix in Eq. 1.17, despite the specification that ψ is real-valued, if $r(M) \geq 2^n - 1$ the operator pool is complete for complex-valued vectors as well. This is true by *complexification*, which states that any basis for a real vector space V can also serve as a basis for a complex vector space V^C, where V^C is generated by extending scalar multiplication to include the imaginary unit [27].

Ensuring a complete operator pool led to an expansion of the original DACO pool and significantly impacted the structure of the ansatz in the DACO-VQA. As such,

completeness will be discussed more extensively in a later chapter.

1.4 Ansatz Entangling Capability

Another relevant feature of quantum algorithms is the entangling capability of their operator pool. In order for destructive interference to take effect throughout the quantum computer, leading to convergence to the ground state, the parameter-dependent ansätze must be generated with high levels of entanglement. As such, it is important to understand the ability of a given operator pool to generate entanglement. A method of quantifying the entangling capability of a parameter-dependent ansatz was described in Ref. [28] based on Meyer-Wallach entanglement of quantum states [29]. The method begins by defining a linear map on an n-bit string (with n the number of qubits):

$$t_j(b) \left|b_1 ... b_n\right\rangle = \delta_{bb_j} \left|b_1 ... \hat{b_j} ... b_n\right\rangle, \tag{1.19}$$

where $\hat{b_j}$ denotes the absence of the j-th bit, and $b_i \in \{0, 1\}$. The result of this map is either an $(n-1)$-bit string or 0. As an example, we have

$$t_3(0) \left|0010\right\rangle = \delta_{01} \left|0010\right\rangle = 0, \tag{1.20}$$

and

$$t_3(1) \left|0010\right\rangle = \left|00\hat{1}0\right\rangle = \left|000\right\rangle. \tag{1.21}$$

Next, we define a distance measure between two bit strings $\left|u\right\rangle = \sum_i u_i \left|i\right\rangle$, $\left|v\right\rangle = \sum_i v_i \left|i\right\rangle$ by

$$D(\left|u\right\rangle, \left|v\right\rangle) = \frac{1}{2} \sum_{i,j} \left|u_i v_j - u_j v_i\right|^2, \tag{1.22}$$

where the factor of $1/2$ is to avoid double counting. This distance measure is es-

sentially the magnitude of the wedge product of the vectors $|u\rangle$ and $|v\rangle$. Now, the entanglement of a given vector $|\psi\rangle$ is given by

$$Q(|\psi\rangle) = \frac{4}{n}\sum_{j=1}^{n} D(t_j(0)\,|\psi\rangle\,, t_j(1)\,|\psi\rangle). \tag{1.23}$$

This measure has the property that $0 \leq Q \leq 1$, and that $Q(|\psi\rangle) = 0$ if and only if $|\psi\rangle$ is a product state, with $Q(|\psi\rangle) = 1$ corresponding to a maximally entangled state. This measure of entanglement is equivalent to the average linear entropy of all single qubit reduced states [30].

With the entanglement measure defined, we can now define the entangling capability of an operator pool as the average entanglement over a set of randomly sampled parameter values:

$$\text{Ent} = \frac{1}{|S|}\sum_{\boldsymbol{\theta}_i \in S} Q(|\boldsymbol{\theta}_i\rangle), \tag{1.24}$$

where $|\boldsymbol{\theta}\rangle$ is the parameter-dependent ansatz, $S = \{\boldsymbol{\theta}_i\}$ is the set of sampled parameter values, and the sum is over the set of parameter samples. The entanglement of different ansätze will be presented in the results section. Presently, we provide results for the entanglement of randomly generated complex-valued vectors over 1000 samples as a benchmark, for several different types of random vector. This is done by plotting the average entanglement of the randomized vectors against $1/q$, where q is the number of qubits. This selection for the x-axis was chosen so the trend toward $q \to \infty$ $(1/q \to 0)$ can be more easily seen. The results are shown in Figs. 1.2, 1.3, 1.4.

The entanglement in Figs. 1.3 and 1.4, with Fig. 1.3 corresponding to random vectors with components of the form $Re^{i\theta}$ and Fig. 1.4 corresponding to random vectors with components of the form $a + bi$ with a, b sampled from Gaussian distributions, both saturate to a value of around 1 when approaching 9 qubits. This means

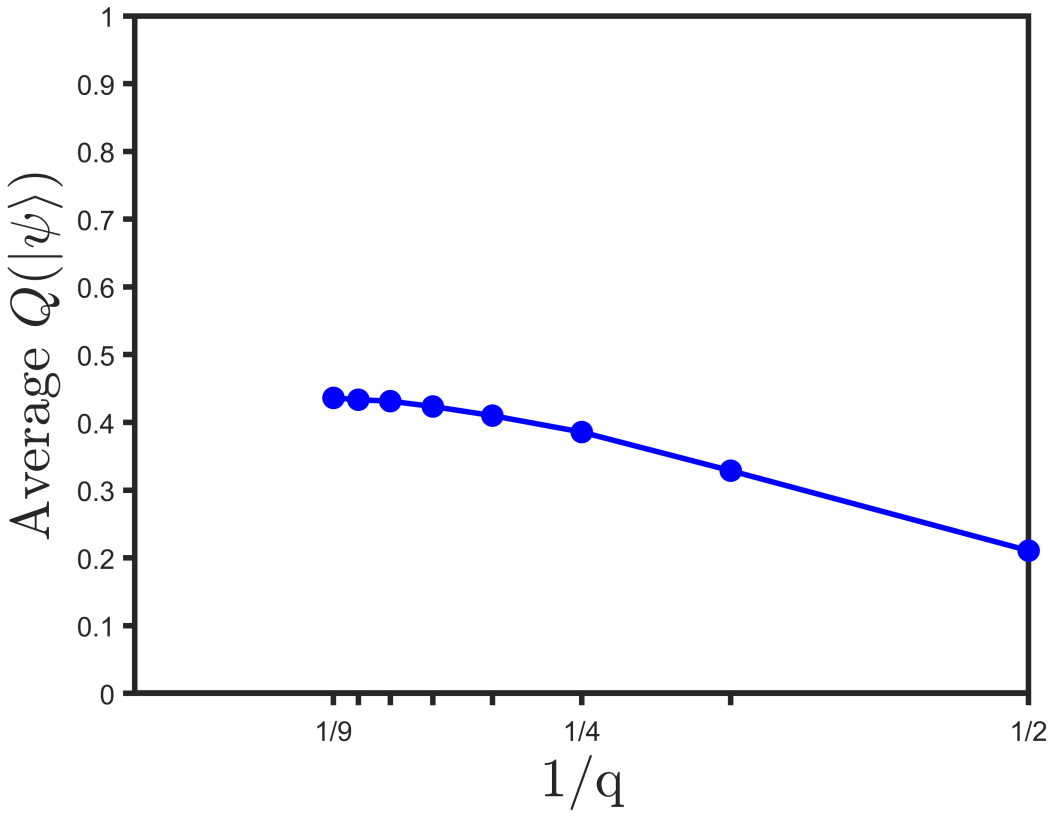

Figure 1.2: Average entanglement for a random complex-valued vector, where each component is of the form $a + bi$, with a, b sampled uniformly between -1 and 1. The value of entanglement begins to saturate to $Q \approx 0.44$ at around 8 qubits.

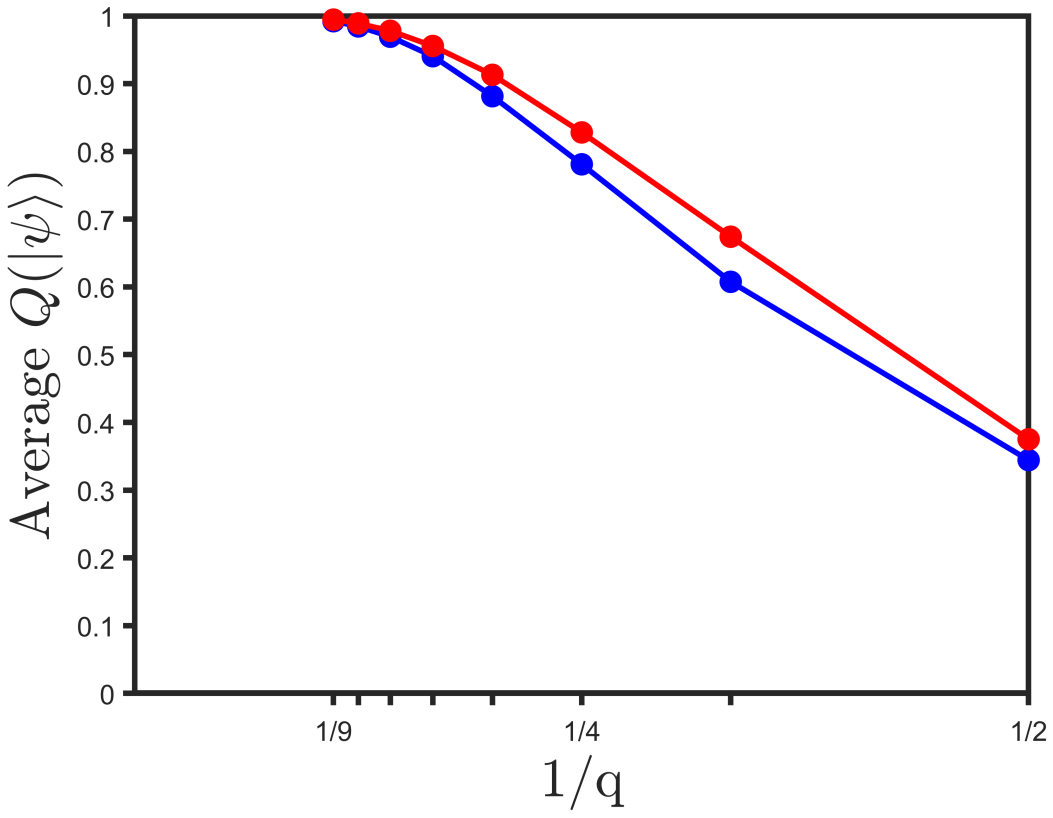

Figure 1.3: Average entanglement for a random complex-valued vector, with components of the form $Re^{i\theta}$. In both cases, θ was uniformly sampled between $-\pi/2$ and $+\pi/2$. The blue line shows R sampled using a Gaussian distribution, with $\mu = 0$ and $\sigma = 0.5$. The red line shows R sampled uniformly between -1 and 1. In both cases, the entanglement saturates to its maximum value of 1 after 9 qubits.

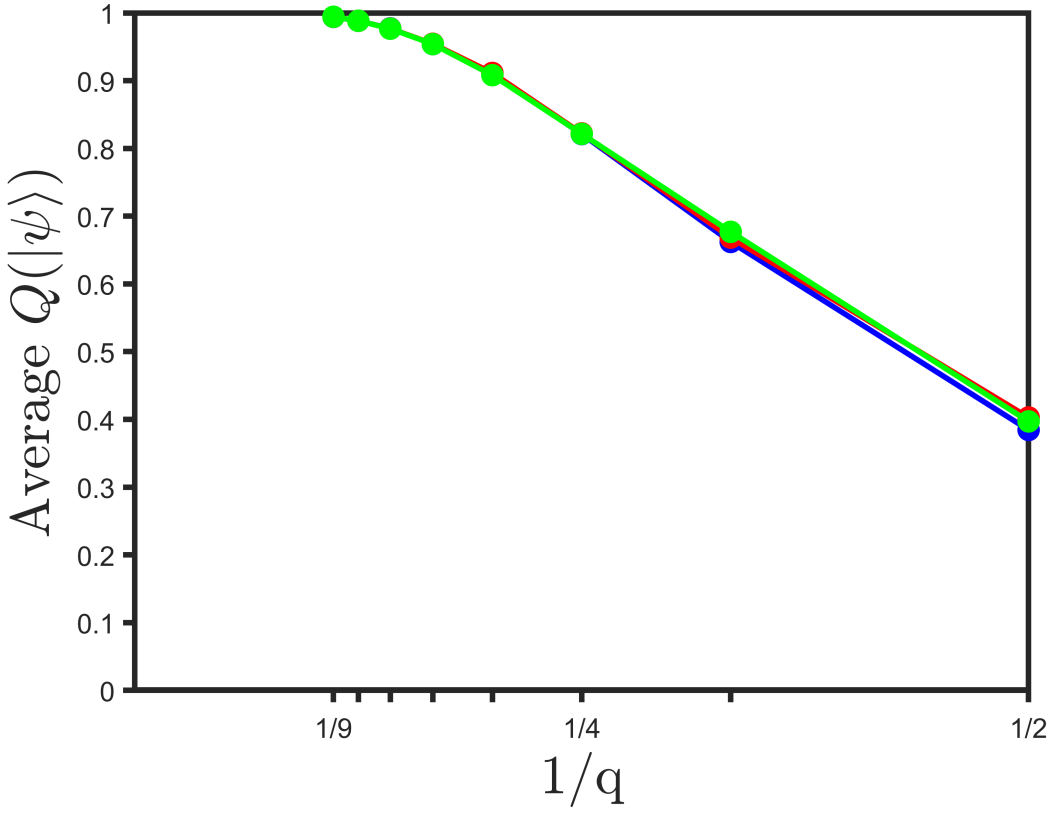

Figure 1.4: Average entanglement for a random complex-valued vector, with components of the form $a + bi$, where a, b were both sampled from a Gaussian distribution. In all cases, $\mu = 0$. The blue line indicates $\sigma = 0.1$, the red line $\sigma = 0.5$, and the green line $\sigma = 0.9$.

that vectors randomly generated in this way quickly become maximally entangled. In contrast, when the randomly generated vector has components of the form $a + bi$ with a, b randomly sampled from a uniform distribution, the entanglement begins to saturate to a value of around 0.44 near 9 qubits, as illustrated in Fig. 1.2. The entanglement measure for uniformly random vectors more closely resembles the entanglement measure observed for various operator pools, as they were not found to saturate rapidly to maximally entangled states.

CHAPTER 2: DIVIDE AND CONQUER ALGORITHM

This chapter serves as an overview of the operators central to this work. It will cover their structure, properties, and implementation in the VQA.

2.1 Operators

The operators which are the centerpiece of this work are the Divide and Conquer Operators, which are operators consisting of tensor products of Pauli matrices. For an n-qubit system, we define n DACOs as follows:

$$\begin{aligned} D_1 &= YXX\cdots XXX, \\ D_2 &= ZYX\cdots XXX, \\ D_3 &= ZZY\cdots XXX, \\ &\vdots \\ D_i &= ZZZ\cdots XXX, \\ &\vdots \\ D_{n-2} &= ZZZ\cdots YXX, \\ D_{n-1} &= ZZZ\cdots ZYX, \\ D_n &= ZZZ\cdots ZZY, \end{aligned} \tag{2.1}$$

where $AB \equiv A \otimes B$. Using these operators, we can build unitary, parameter-dependent operators as

$$U_i(\delta_i) \equiv e^{-i\delta_i D_i}. \tag{2.2}$$

These operators can be used to generate a hardware-efficient ansatz

$$|\boldsymbol{\delta}\rangle \equiv U_n(\delta_n)\cdots U_1(\delta_1)\,|H\rangle\,, \tag{2.3}$$

or can be combined with operators built from the Hamiltonian to generate a problem-specific ansatz in various ways. Since there are only n operators for an n-qubit system, these operators provide a low circuit depth option for generating ansätze, making them ideal for NISQ-era quantum computing algorithms.

These operators have the property that, given an initial Hadamard state $|H\rangle$, any pure state (a single bit string rather than a superposition of bit strings) can be reached through the application of the n operators $U_i(\delta_i)$ with δ_i taking either the value $-\pi/4$ or $+\pi/4$. For example, for a 4-qubit system, we have that

$$e^{-i(-\pi/4)D_4}e^{-i(+\pi/4)D_3}e^{-i(-\pi/4)D_2}e^{-i(-\pi/4)D_1}\,|H\rangle = [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]^{\mathrm{T}}\,, \tag{2.4}$$

which is a pure state in the computational basis. Each of the 2^n combinations of $(\delta_1,\delta_2,\delta_3,\delta_4) = (\pm\pi/4,\pm\pi/4,\pm\pi/4,\pm\pi/4)$ corresponds uniquely to one of the 2^n pure states. The action of $U_1(\delta_1 = \pm\pi/4)$ on the Hadamard state turns either the first $2^n/2$ entries to 0 or the last $2^n/2$ to 0, in a sense splitting the state in half. This is shown explicitly for $n = 4$ below:

$$\begin{aligned} U(\delta_1 = +\pi/4)\,|H\rangle &= \frac{1}{\sqrt{2^3}}[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1]^{\mathrm{T}},\\ U(\delta_1 = -\pi/4)\,|H\rangle &= \frac{1}{\sqrt{2^3}}[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0]^{\mathrm{T}}. \end{aligned} \tag{2.5}$$

The second operator has a similar effect, sending half of the remaining 1 entries to 0. The application of all n operators sequentially sends half of the remaining 1 entries to 0, leaving only one nonzero entry and thus obtaining a pure state. This property of the operators "splitting" the initial Hadamard state into consecutive halves of 1's

and 0's is the reason for the name Divide and Conquer Operators, as the operators divide up the entries consecutively until a final pure state is obtained.

In general, operators such as the DACOs consisting of tensor products of Pauli matrices are called Pauli strings. Pauli strings have two important properties: they all square to the identity matrix (operators with this property are called involutory), and they are all sparse. These two properties in combination are incredibly important in constructing an efficient classical simulation of a quantum algorithm. Sparsity allows for a sparse representation of the matrices to be stored in the computer. Specifically, the Pauli strings have one nonzero entry in each column, meaning for a Pauli string with matrix size $N \times N$, only $2N$ numbers need to be stored by the computer rather than N^2. For each of the N nonzero matrix elements, one number specifies the value of the matrix element, and one number specifies the column/row it falls in. This significantly reduces the space required to store the operators while simulating the VQA on a classical computer, and also allows for much faster matrix multiplication throughout the simulation. Sparsity on its own, however, is not enough to accomplish this increase in classical computational efficiency. The operators must also be involutory. This is due to exponentiation of the operators to generate the unitary quantum gates. In general, it is not true that a sparse matrix, when exponentiated, remains sparse. However, for an involutory operator A, we have that

$$e^{-i\theta A} = \cos(\theta) I - i \sin(\theta) A. \tag{2.6}$$

This can be seen by Taylor expanding the exponential. Since the operator A is involutory, the even power terms are all equal to the identity matrix I, and their coefficients can be grouped together to form the $\cos(\theta)$ term. The odd power terms are all equal to A, and their coefficients can be grouped similarly to form the $-i \sin(\theta)$ term. Thus, since the identity matrix and A are both sparse and $e^{-i\theta A}$ is a linear combination of these two matrices, $e^{-i\theta A}$ also has a sparse representation, and can

thus be efficiently stored in a computer.

2.1.1 Operator Completeness

For the DACOs outlined above to be usable generally, they must form a complete operator pool. This was tested numerically by calculating the rank of the overlap matrix (Eq. 1.17) for a large number of randomly generated real vectors $|\psi\rangle$. Despite the compelling divide and conquer properties of these operators on their own, numerical results indicate that the DACOs do not form a complete operator pool. These results are illustrated in Fig. 2.1 by plotting the rank of the overlap matrix generated by the DACOs against the number of qubits. Since the DACOs were not complete on their own, a primary goal of this work was to identify a supplementary operator pool which would serve to "complete" the pool. Two such sets of operators were found, and will be discussed in the next subsection.

2.1.2 Associated Operators

Since the DACOs failed to form a complete operator pool on their own, two supplementary sets of operators were identified which, when added to the original pool, led to the completeness criterion being satisfied. These secondary sets of operators are termed associated DACOs, and their construction was inspired by the structure of general spin operators in quantum mechanics:

$$S = \sin(\theta)\left(\cos(\phi)X + \sin(\phi)Y\right) + \cos(\theta)Z, \tag{2.7}$$

where X, Y, Z are the typical Pauli spin matrices. Since the original set of DACOs contain a single Y operator each, the two secondary sets of operators were constructed by replacing the Y operators with X and Z operators, with C_i denoting the "X" operators, and E_i denoting the "Z" operators. Specifically, the new operators were defined as:

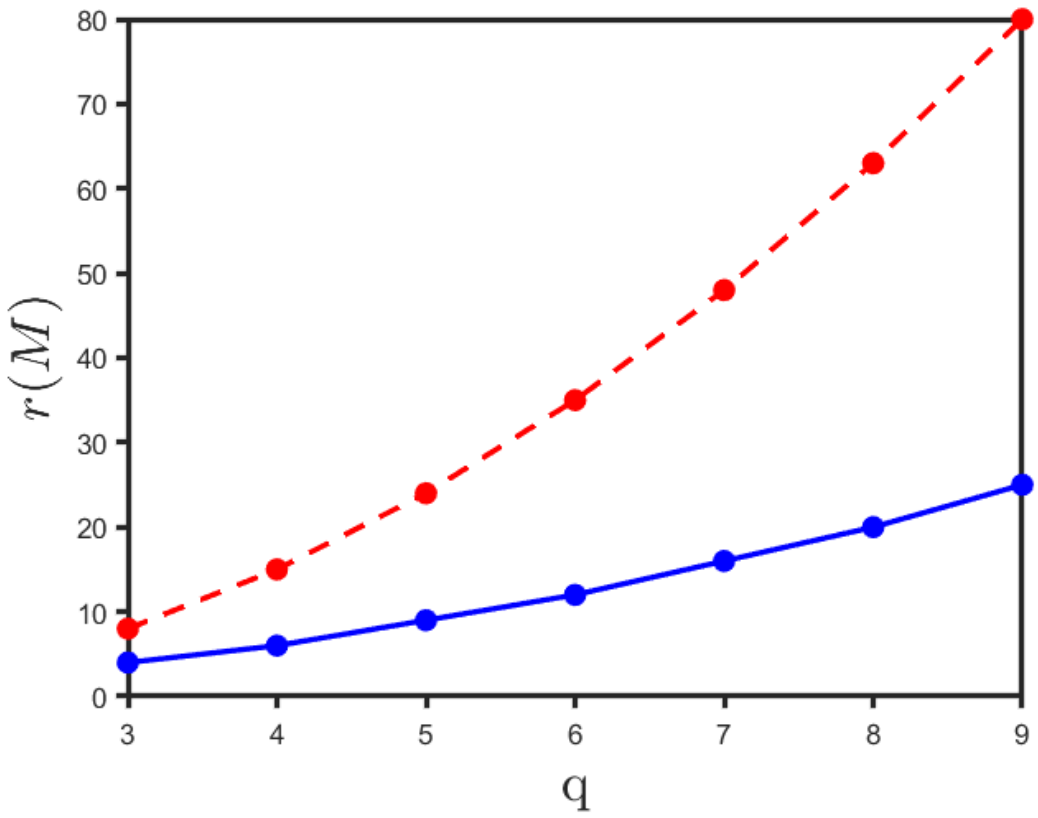

Figure 2.1: Completeness (rank of the overlap matrix M) for the D_i operators vs the number of qubits. The blue line indicates $r(M)$ for D_i, while the red line plots $2^q - 1$, which is the minimum required value of $r(M)$ to obtain completeness.

$$
\begin{aligned}
C_1 &= XXX\cdots XXX,\\
C_2 &= ZXX\cdots XXX,\\
C_3 &= ZZX\cdots XXX,\\
&\vdots\\
C_i &= ZZZ\cdots XXX,\\
&\vdots\\
C_{n-2} &= ZZZ\cdots XXX,\\
C_{n-1} &= ZZZ\cdots ZXX,\\
C_n &= ZZZ\cdots ZZX,
\end{aligned}
\tag{2.8}
$$

and

$$
\begin{aligned}
E_1 &= ZXX\cdots XXX,\\
E_2 &= ZZX\cdots XXX,\\
E_3 &= ZZZ\cdots XXX,\\
&\vdots\\
E_i &= ZZZ\cdots XXX,\\
&\vdots\\
E_{n-2} &= ZZZ\cdots ZXX,\\
E_{n-1} &= ZZZ\cdots ZZX,\\
E_n &= ZZZ\cdots ZZZ.
\end{aligned}
\tag{2.9}
$$

These operators were originally used to construct a mixing Hamiltonian B for use in the quantum alternating operator ansatz as follows:

$$
B_k(\theta_k, \phi_k) = \sin(\theta_k)\left(\cos(\phi_k)C_k + \sin(\phi_k)D_k\right) + \cos(\theta_k)E_k. \tag{2.10}
$$

Early in this work, however, this ansatz was found to have inferior performance when compared to a different ansatz which will be explained in the next subsection.

Another useful aspect of these associated operators was their ability to distribute phase throughout the ansatz in the quantum computer. On their own, the DACOs were able to effectively cut the Hilbert space in half, but were not able to distribute phase throughout the remaining states to generate the destructive interference necessary to converge to the ground state of the system (this property seems to be related to completeness of the operator pool, though no systematic study was conducted to verify this). The associated operators, however, successfully moved phase around the states in the Hilbert space, leading to convergence to the ground state throughout the VQA procedure. As such, these operators became a mainstay in different ansatz structures which were explored, and many variations were developed which could utilize these operators.

2.2 Ansatz

The original ansatz structure explored was a quantum alternating operator ansatz utilizing the operator defined in Eq. 2.10 as the mixing Hamiltonian:

$$|\boldsymbol{\beta},\boldsymbol{\gamma}|\boldsymbol{\theta},\boldsymbol{\phi}\rangle = \prod_{k=1}^{n} e^{-i\beta_k B(\theta_k,\phi_k)} e^{-i\gamma_k H_C} |H\rangle\,, \tag{2.11}$$

with n the number of qubits and H_C the problem Hamiltonian. This ansatz was implemented with a modified version of the diffusion optimizer, which drove the values of θ_k and ϕ_k to $\pi/2$, γ_k to 0, and β_k to $\pm\pi/4$. This meant that by the end of the VQA procedure, the ansatz would be driven to a pure state, since

$$B_k(\pi/2,\pi/2) = \sin(\pi/2)\left(\cos(\pi/2)C_k + \sin(\pi/2)D_k\right) + \cos(\pi/2)E_k = D_k, \tag{2.12}$$

implies that the only operators remaining are the original DACOs with angles $\delta =$

$\pm\pi/4$. Since the problem Hamiltonian is designed specifically so that the ground state is a pure state in the computational basis (since it is an energy eigenstate), the idea was to let the optimizer explore the Hilbert space freely, then drive toward a pure state once specific criteria had been met by implementing dynamic biases. While in general this ansatz led to convergence to the ground state, its sampling of the Hilbert space was larger than subsequently developed ansätze, and so was dropped in favor of the better ansatz structures.

The second iteration of the ansatz structure was inspired by a Trotter-decomposition of the quantum alternating operator ansatz with mixing Hamiltonian $B(\theta, \phi)$. Given a unitary operator e^{A+B} where A, B are operators, the Trotter-decomposition is defined as

$$e^{A+B} = \lim_{n\to\infty} \left(e^{A/n} e^{B/n}\right)^n . \tag{2.13}$$

If a finite number of products are taken, rather than taking the limiting case $n \to \infty$, we have

$$e^{A+B} \approx \left(e^{A/N} e^{B/N}\right)^N , \tag{2.14}$$

where N is called the Trotter number. The new ansatz structure is essentially a Trotter-decomposition of the original alternating ansatz with a Trotter number of 1:

$$|\boldsymbol{\epsilon}, \boldsymbol{\delta}, \boldsymbol{\gamma}, \boldsymbol{\theta}\rangle = \prod_{k=1}^{n} e^{-i\epsilon_k E_k} e^{-i\delta_k D_k} e^{-i\gamma_k C_k} e^{-i\theta_k H_C} |H\rangle , \tag{2.15}$$

where ϵ, δ, γ are the tunable parameters for the DACOs and associated DACOs, and θ is now the tunable parameter controlling the problem Hamiltonian. The ordering of the product of operators in this ansatz turned out to be an important feature, with $e^{-i\epsilon_k E_k} e^{-i\delta_k D_k} e^{-i\gamma_k C_k}$ and $e^{-i\gamma_k C_k} e^{-i\delta_k D_k} e^{-i\epsilon_k E_k}$ the only orderings which led to consistent convergence to the ground state. Of the two, the first ordering (called the

EDC ordering) performed slightly better than the second (CDE). As such, EDC is referred to as the canonical ordering of operators. Interestingly, a quantification of the entangling capability of each of these orderings corresponded to their success in converging to the ground state. These observations will be discussed in greater detail in the results chapter.

The final iteration of the DACO-VQA ansatz is a modifed version of the ansatz in Eq. 2.15. This ansatz makes deliberate use of the divide and conquer property of the DACOs to restrict the Hilbert space search. The ansatz works in stages, sequentially reducing the search space by applying consecutive DACOs in the fashion of Eq. 2.4. During the i-th stage, two searches are run in parallel, one corresponding to the "top" states ($U_i(\delta_i)$ with $\delta_i = -\pi/4$) and one to the "bottom" states ($U_i(\delta_i)$ with $\delta_i = +\pi/4$). At the end of the stage, whichever search yielded the better score is chosen (based on the cost function to be discussed), and the next DACO – $U_{i+1}(\delta_{i+1} = \pm\pi/4)$ – is applied and the process is repeated. The ansatz is constructed as follows:

$$|\boldsymbol{\epsilon}, \boldsymbol{\delta}, \boldsymbol{\gamma}, \theta\rangle = \left[\prod_{k=1}^{n} e^{-i\epsilon_k E_k} e^{-i\delta_k D_k} e^{-i\gamma_k C_k}\right] e^{-i\theta H_C} \prod_{m=1}^{S} e^{-i\delta_{0,m} D_m} |H\rangle\,, \tag{2.16}$$

where S is the current stage and $\delta_{0,m} = \pm\pi/4$. The ansatz is structured so that the DACOs (up to the current stage S) hit the Hadamard state $|H\rangle$ first to restrict the search to consecutive halves of the Hilbert space. Following the restriction, the state is hit with the Hamiltonian unitary, then the set of operators with parameters to be optimized over. Since only a finite number of measurements are taken throughout the VQA procedure, restricting the search to consecutive halves of the Hilbert space can drastically reduce the total number of samples needed to converge to the ground state of the system when compared to other VQAs. In addition, the product of operators contained in the brackets can be swapped with other VQA ansatz structures. As long

as the ansatz structure included in the brackets keeps the states contained to their respective halves of the Hilbert space, they can be implemented into the DACO-VQA structure. This notion of containing the states to their respective sections of the Hilbert space according to the initial DACOs was termed *confinement*. All combinations of $U(\epsilon)$, $U(\delta)$, and $U(\gamma)$ maintain confinement, as well as the operators in the original QAOA ansatz structure. Future work will be dedicated to implementing other ansatz structures in the overall DACO-VQA structure, and characterizing their performance against the ansatz above.

Another important feature of this ansatz is the Hamiltonian parameter θ. Instead of letting θ be a variational parameter to be optimized over, a finite sampling of θ values is taken, cycling between 0 and 2π. In the current iteration of the algorithm, there are 32 samples taken, meaning θ takes on values which are integer multiples of $2\pi/32 = \pi/16$. The different θ values only occur in the measurement process, and are not used to influence subsequent rounds of the DACO-VQA. They are only contained in the ansatz to utilize the Hamiltonian and provide higher resolution during the measurement process, as will be discussed in section 2.3.

2.2.1 Replicas

The final feature of the ansatz which was explored in this work is the concept of *replicas*. Replicas were inspired by parallelization, and serve to repeat the Hamiltonian in the ansatz to effectively run the same system multiple times simultaneously within the quantum computer. To generate an ansatz with R replicas, we take the tensor product of the $R \times R$ identity matrix with the q-qubit problem-Hamiltonian H_C:

$$H = I_{R\times R} \otimes H_C. \tag{2.17}$$

For example, with $R = 2$ we have

$$H = I_{2\times 2} \otimes H_C = \begin{bmatrix} H_C & 0 \\ 0 & H_C \end{bmatrix}, \tag{2.18}$$

repeating the Hamiltonian twice. To implement this physically, we must include r new qubits to generate the replicas, where $2^r = R$. Thus, the total number of qubits required to simulate a q-qubit problem with R replicas is $n = q + r$. The size of the Hilbert space for the original Hamiltonian H_C is $Q = 2^q$, and once replicas are introduced, the new Hilbert space size becomes $N = 2^n = 2^{q+r} = QR$. When this idea was originally posed, the hope was to gain advantage by having the different replicas interact with one another to drive the overall energy of the ansatz down, approaching the ground state, which would ideally outweigh the cost of adding more qubits to the system. However, in practice the replicas do not appear to be interacting to drive the overall energy down, and there does not seem to be any improvement in the performance of the algorithm when replicas are added, which will be discussed in more detail in the results section. As a concept, replicas are still interesting, and propositions to gain the needed interaction between replicas will be proposed in the conclusions section as future work.

2.3 Cost Function and Classical Optimizer

The cost function and classical optimizer employed in the DACO-VQA are used in conjunction to drive the states produced by the quantum computer to the desired ground state of the system. The classical optimizer uses a diffusion process to randomly generate new parameter values for the quantum computer to produce corresponding states. For each new set of parameters, the cost function determines whether the corresponding state is approaching the ground state of the system. If the value of the cost function improves, the classical optimizer adaptively changes the step size of the diffusion to keep pushing toward the ground state of the system. If the value of the cost function worsens, the classical optimizer will instead continue to

explore the parameter space without driving toward any specific parameter values.

For each round of the VQA process, a finite set of M energy measurements, denoted by $\{E_i\}$, are taken of the quantum state generated by the quantum computer per θ value to be used in the evaluation of the cost function. This leads to a total of $32M$ measurements per round of the VQA, since there are 32 samples of θ per round. Utilizing the measured energies $\{E_i\}$, there are two criteria which constitute the cost function. The first is the minimum energy of the current set of energies $E_{\text{min}} = \min\{E_i\}$, and the second is a score based on the average of the partition function of the energies $Z_{\text{avg}} = (1/M)\sum_i e^{-(E_i - \bar{E}_{\text{min}})/kT}$ and a progress variable progE $= (\bar{E}_{\text{max}} - \langle E\rangle)/(\bar{E}_{\text{max}} - \bar{E}_{\text{min}})$. Here kT is an adaptive parameter with units of energy, $\bar{E}_{\text{min}}$ and $\bar{E}_{\text{max}}$ are the lowest and highest energies measured across all previous rounds respectively, and $\langle E\rangle$ is the average of the energies $\{E_i\}$. The score is specifically defined as

$$\text{score} = \frac{Z_{\text{avg}} + \text{progE}}{2}. \tag{2.19}$$

Both Z_{avg} and progE range between 0 and 1, so the score is also restricted to between 0 and 1. Within the score, progE serves to drive the average energy of the states down in the earlier stages of the algorithm while the search is still mostly unconstrained, while Z_{avg} penalizes clustering of high energy states which is useful in later stages of the algorithm. Throughout the VQA process, a newly generated set of angles is accepted if either $E_{\text{min}} < \bar{E}_{\text{min}}$ or if score $>$ score$_{\text{best}}$, where score$_{\text{best}}$ is the highest score recorded across all rounds of the VQA. If neither criteria is satisfied, the new set of angles is rejected, and the classical optimizer selects new angles based on the diffusion process described above. Once the score exceeds 0.99, the process ends and the solution is obtained.

CHAPTER 3: RESULTS

Here, we present the various results obtained for different features of the DACO-VQA procedure. We begin with operator pool completeness since this is a prerequisite to obtaining convergence to ground states in a VQA process. We then discuss entangling capability of different operator orderings in the ansatz of Eq. 2.16 to motivate the final choice of operator ordering. Finally, we discuss the overall performance of the DACO-VQA algorithm, as well as the effect of adding replicas to the system.

3.1 Completeness

As discussed above, completeness of a VQA operator pool is a necessary condition to obtain consistent convergence to the ground state of the problem-Hamiltonian H_C. As such, characterizing the completeness of different operator pools was a major initial focus of this work. The results presented here show the completeness of various combinations of the DACOs and associated DACOs, as well as for a special combination of associated operators.

Specifically, based on the definition of the C_i and E_i operators, one can see that the set of C_i and E_i are identical, with the exception of C_1 and E_n, since for every index i, $E_i = C_{i+1}$. This means that for an operator pool consisting of both C_i and E_i, the pool size is only $n+1$ rather than $2n$. However, in the context of the DACO-VQA, this is not an interesting feature, since if the ansatz is constructed from only C_i and E_i without D_i, convergence to the ground state was not obtained. However, in the context of adaptive VQAs which iteratively build the ansatz one operator at a time, there is a potential for these operators to be utilized, reducing resources required for the quantum computer to function. In the original adaptive-VQA which

Table 3.1: Completeness of various operator pools. ✗ indicates an incomplete pool, ✓ indicates a complete pool.

Operators	$\{C_i\}$	$\{D_i\}$	$\{E_i\}$	$\{C_i\}+\{D_i\}$	$\{D_i\}+\{E_i\}$	$\{C_i\}+E_n$
Completeness	✗	✗	✗	✓	✓	✓

was developed to solve for the ground state of fermionic systems, the operator pools required had a minimal size of $2n-2$, which is larger than $n+1$. However, in this context the operators in the pool were required to contain an odd number of Y Pauli operators in their Pauli strings to physically correspond to fermionic creation and annihilation operators. This means that C_i and E_i could not be directly implemented in a fermionic adaptive-VQA, but it is still possible that they could be implemented in a different adaptive-VQA, which may be explored in future work.

The completeness results were obtained for up to $q=9$ qubits by calculating $r(M)$ for the following operator pools: $\{C_i\}$, $\{D_i\}$, $\{E_i\}$, $\{C_i\}+\{D_i\}$, $\{D_i\}+\{E_i\}$, and $\{C_i\}+E_n$. The notation $\{C_i\}+\{D_i\}$ indicates that the pool is comprised of all C_i and D_i operators, and $\{C_i\}+E_n$ is the set of C_i operators and E_i operators, where only E_n was included for the reasons outlined above. The pool was deemed complete if $r(M) \geq 2^q-1$ for every value of q up to $q=9$. The results are summarized in Table 3.1.

The results indicate that for any pool including D_i, the minimal pool size required for completeness is $2n$, regardless of which other associated operators are added to the pool. Unsurprisingly, any individual set of operators on their own do not form a complete pool. In addition, ansätze which use all 3 sets of operators are automatically complete, because once a pool is complete, adding more operators has no effect on $r(M)$ (it cannot decrease or increase the rank).

3.2 Entangling Capability

The entangling capability of many different orderings of DACOs and associated operators were considered. However, since only the orderings EDC and CDE (defined above) led to consistent convergence to the ground state, only they are presented in this section. The entangling capability of each of these operator orderings in the DACO-VQA ansatz was calculated for up to 10 qubits, with 10000 parameters samples averaged over for each qubit value. The results of these numerical simulations are plotted in Fig. 3.1.

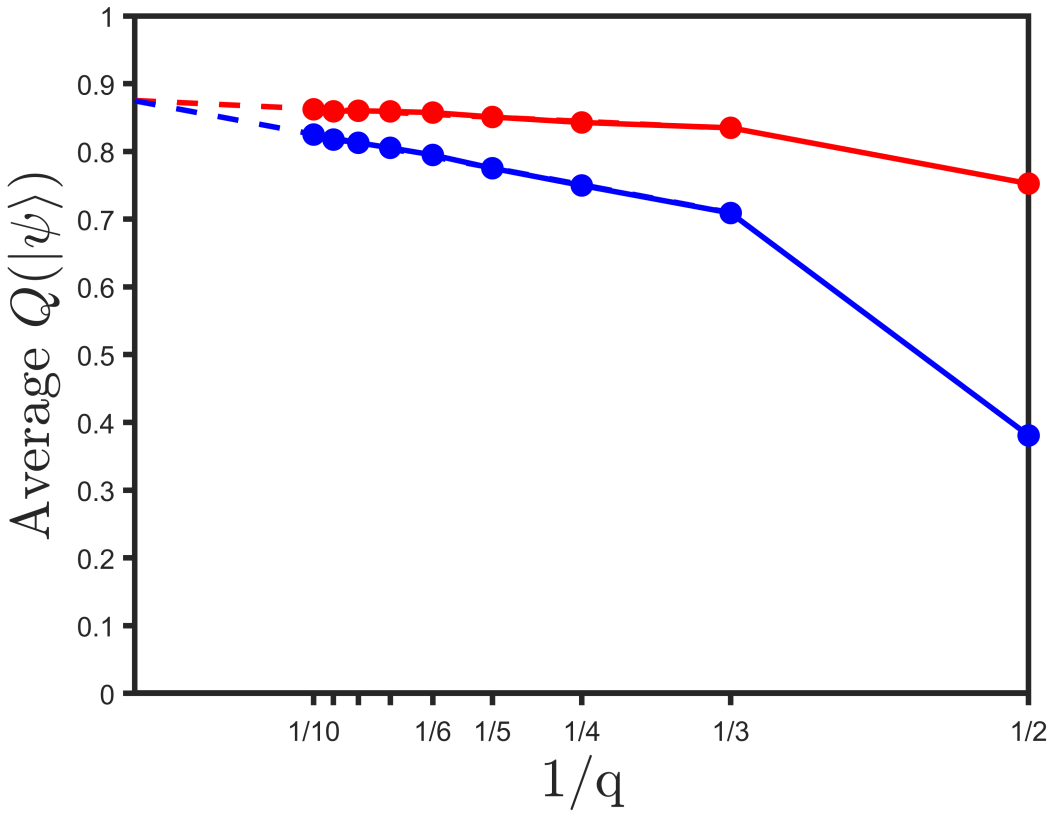

Figure 3.1: A comparison of the entangling capability of the EDC and CDE orderings. CDE corresponds to the blue line, EDC corresponds to the red line. The results are plotted from 2 to 10 qubits, with 10000 parameter samples taken per qubit. The dashed lines are the lines of fit for the data points starting at $q = 3$.

The y-intercepts for the lines of fit occur at 0.8752 for the EDC ordering, and 0.8742

for the CDE ordering, indicating that EDC always maintains the higher entangling capability up to $q \to \infty$. If there is a concrete connection between entangling capability and the oversampling rate of the Hilbert space, this indicates that the EDC ordering maintains the advantage over CDE for all possible numbers of qubits.

3.3 Algorithm Performance

In this section, we characterize the performance of the algorithm for various problem-Hamiltonians. The specific energy spectra used were the hydrogen atom $E_n = -1/n^2$, Gaussian-distributed energies with $\mu = 1$ and $\sigma = 2$, and $E_n = n^2$. $E_n = -1/n^2$ represents an extreme case where the ground state energy is significantly separated from clustered high energy states. $E_n = n^2$ represents another extreme case where the high energy states become increasingly distant from one another with increasing n. The Gaussian case represents a middle ground, with the low and high energy states all relatively close to one another, and where there is not a significant gap between the minimum energy and the first excited state.

In order to characterize the performance of the algorithm, we track the ratio of total measurements to the size of the Hilbert space, with the ratio denoted by κ. If $\kappa > 1$, the Hilbert space was oversampled, meaning more measurements were taken than there are states in the Hilbert space. For quantum advantage to occur in the context of a VQA, the number of total measurements must be less than the size of the Hilbert space, indicating an oversampling ratio $\kappa < 1$. Note that we must have $\kappa > 0$, otherwise the number of measurements could be negative which is impossible.

The following results show the log of the average oversampling ratio sampled 100 times per qubit ($\log_{10} \langle \kappa \rangle$), plotted against the number of qubits for each of the Hamiltonians listed above. For these simulations, the number of measurements taken per round per θ angle is 32, indicating a total of $32 \times 32 = 1024$ measurements per round. Because of this, for less than 10 qubits, the Hilbert space will necessarily be oversampled. For this reason, lines of fit are plotted only for the points where $q > 10$,

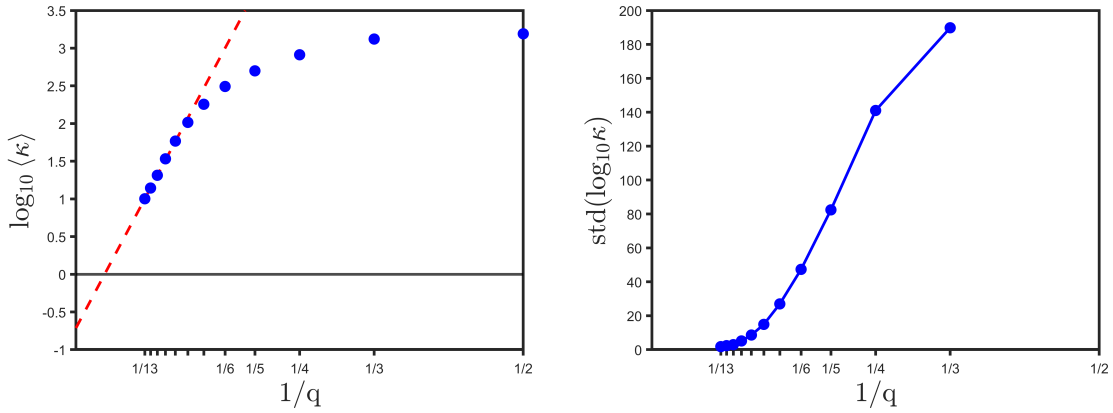

Figure 3.2: (a) Log of the average oversampling ratio for 100 samples for $E_n = n^2$. Blue dots indicate the oversampling data points, the dashed red line indicates the line of fit for the data at $q > 10$, and the black line indicates the crossover point for quantum advantage. (b) Standard deviation of κ over all sampled values.

so that the actual quantum effects are dominating the optimization, rather than the sheer number of measurements. The energy spectrum $E_n = n^2$ is plotted in Fig. 3.2, the Gaussian-distributed energies in Fig. 3.3, and the $E_n = -1/n^2$ spectrum in Fig. 3.4.

The first important observation is that there is not much difference in the oversampling rate for the different Hamiltonians. In particular, the $E_n = n^2$ and the Gaussian-distributed energy spectra share the most similarities. For these two spectra, the average oversampling is almost identical, whereas for the $E_n = -1/n^2$ spectra, the oversampling rate is slightly higher. For example, at 13 qubits, the average oversampling for $E_n = n^2$ and the Gaussian distributed energies are $\langle\kappa\rangle = 10.042$ and $\langle\kappa\rangle = 10.032$ respectively, while the average oversampling for $E_n = -1/n^2$ is $\langle\kappa\rangle = 14.20$. In addition, the standard deviations of κ for $E_n = -1/n^2$ vary more across different qubits than for the other two spectra (in this case the standard deviation is much less smooth), as illustrated in the figures. In all cases, however, both the oversampling rate and standard deviations drop rapidly as the number of qubits increases. This decrease in oversampling is promising, and the red dashed lines in the

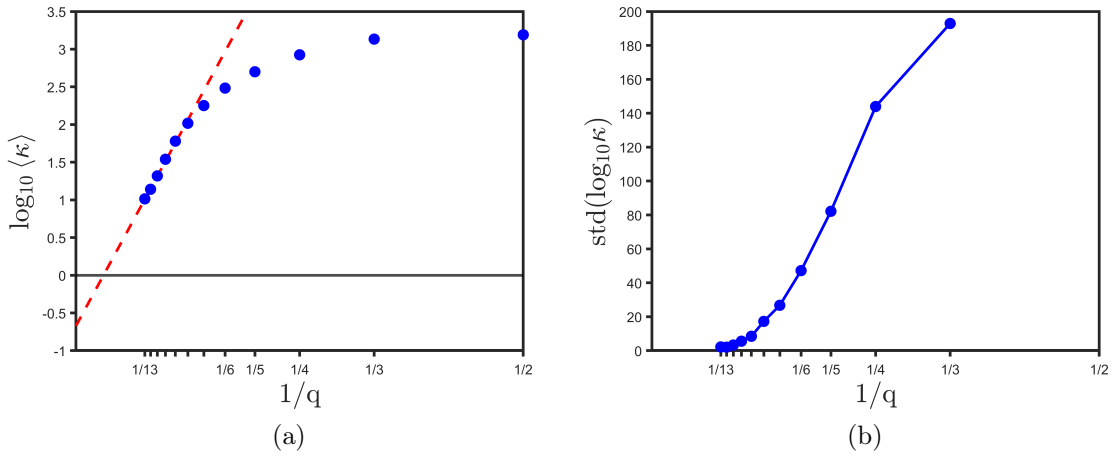

Figure 3.3: (a) Log of the average oversampling ratio for 100 samples for the Gaussian-distributed energy spectrum with $\mu = 1$, $\sigma = 2$. Blue dots indicate the oversampling data points, the dashed red line indicates the line of fit for the data at $q > 10$, and the black line indicates the crossover point for quantum advantage. (b) Standard deviation of κ over all sampled values.

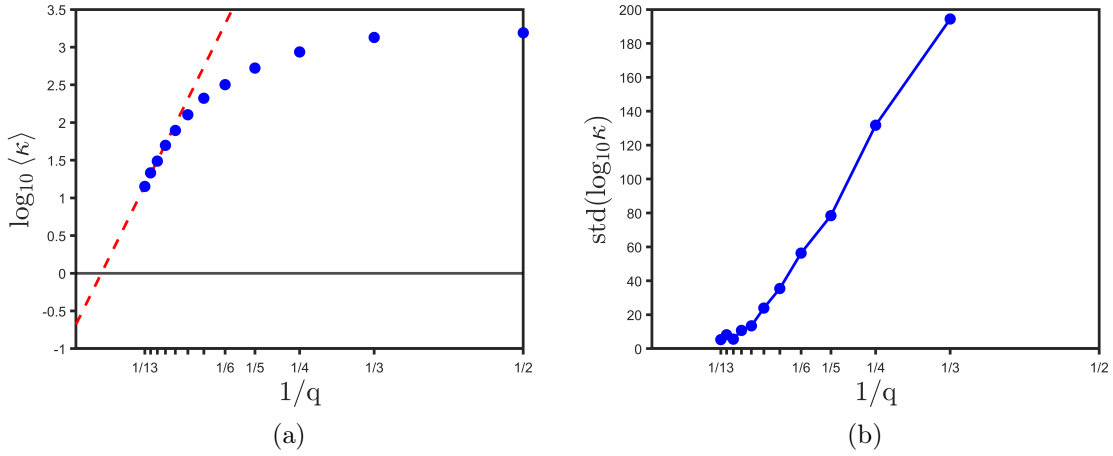

Figure 3.4: (a) Log of the average oversampling ratio for 100 samples for $E_n = -1/n^2$. Blue dots indicate the oversampling data points, the dashed red line indicates the line of fit for the data at $q > 10$, and the black line indicates the crossover point for quantum advantage. (b) Standard deviation of κ over all sampled values.

plots indicate the projection of the oversampling rate to $q \to \infty$, with the crossover between the red and black lines indicating where the Hilbert space would be undersampled. In all three cases, this crossover points occurs around $q = 33$. At face-value, this would indicate that for this algorithm, quantum advantage would be obtained beyond 33 qubits. However, simulating systems with 33 qubits is beyond the bounds of our current computational resources, so this regime cannot currently be probed unless the efficiency of the algorithm is drastically increased. In addition, it is not clear that for larger qubits the same trend will be followed. It is entirely possible that as the number of qubits increases, other mechanisms will begin to take effect that disrupt this trend, thus still requiring oversampling of the Hilbert space to reach the ground state. Despite this, the sampling rate for this algorithm is promisingly low, and so it is worth dedicating future work to attempting to explore this high-qubit regime.

We also explored the oversampling ratio for different Hamiltonians as the number of replicas (section 2.2.1) was varied. As outlined above, the initial motivation for including replicas of the Hamiltonian was to improve the sampling rate by "parallelizing" the Hamiltonian. It was hoped that increasing the number of replicas of the Hamiltonian in the VQA ansatz would lead to better algorithm performance. However, the replicas turned out to have no effect on the performance. This is illustrated in Fig. 3.5 by plotting the oversampling rate vs r, where $R = 2^r$ is the number of replicas, for between 5 and 8 qubits. While there are slight fluctuations in the oversampling rate for different numbers of replicas, they are effectively identical for the same number of qubits, thus including replicas does not influence performance of the algorithm.

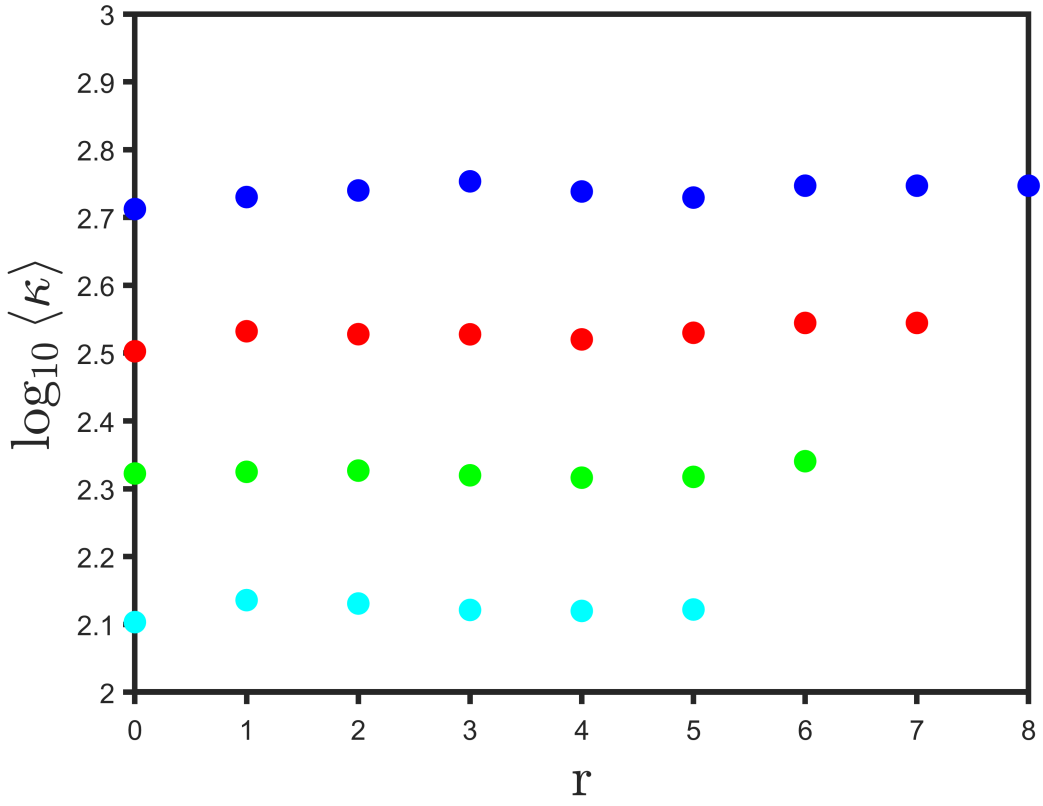

Figure 3.5: Log of the average oversampling ratio (over 100 samples) for the $E_n = -1/n^2$ spectrum with q ranging from 5 to 8, and r up to $n = q + r = 13$. The blue dots are $q = 5$, red are $q = 6$, green are $q = 7$, and cyan are $q = 8$.

Finally, we present the entanglement of the ansätze as they are generated throughout the VQA process. These results show the entanglement of the ansatz generated in each round of the DACO-VQA at each of the 32 θ values for the three Hamiltonians described above. These plots demonstrate how at the beginning of the DACO-VQA process, while the parameter space is being freely explored, highly entangled states are generated. As the process continues, the entanglement is slowly reduced to 0 as the classical optimizer and cost function drive toward the ground state of the system (a product state with no entanglement). It should be noted that for the $E_n = n^2$ and Gaussian-distributed energy spectra, there is significantly more variation of the entanglement throughout the VQA process when compared to $E_n = -1/n^2$. Considering

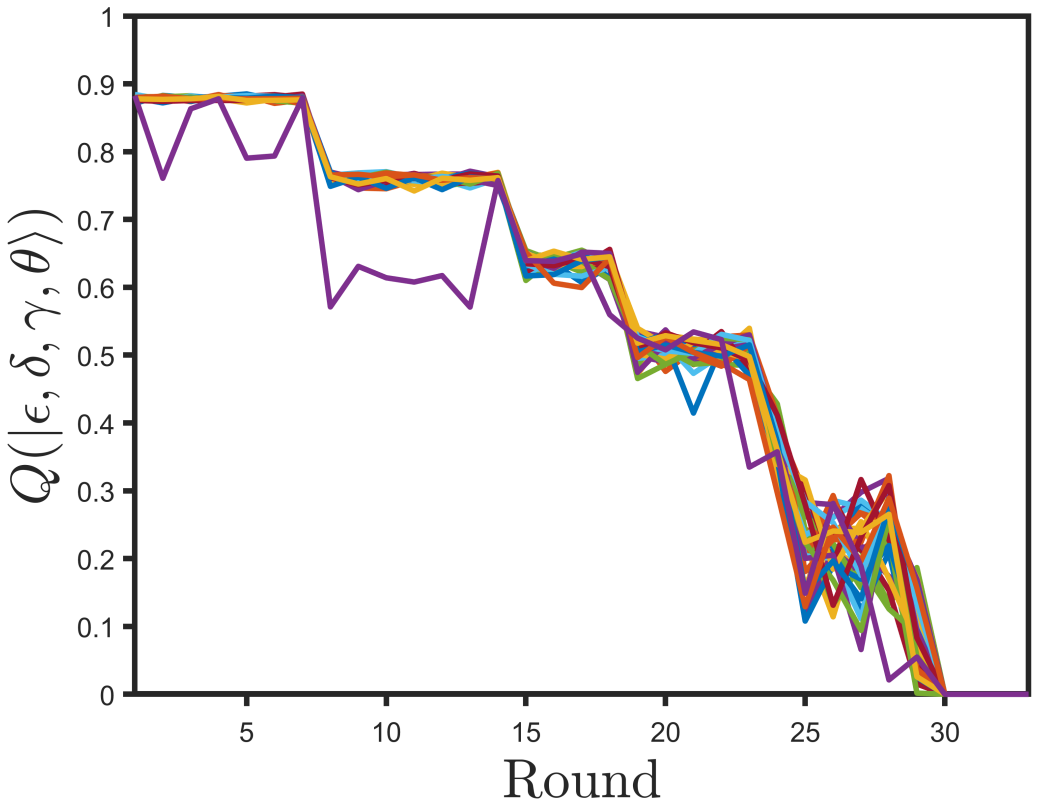

Figure 3.6: Entanglement of the ansätze generated by the quantum computer for each round of the algorithm at each θ value. In this case, the energy spectrum was $E_n = n^2$. Each line in the plot represents one of the 32 θ values.

the oversampling rates for the energy spectra, this further indicates that entanglement plays a role in algorithm performance, as here more variation in entanglement corresponds to better algorithm performance. Despite this correlation, differences in performance are slight, and a more thorough study must be conducted to make any definitive statements on the relation between oversampling and ansatz entanglement.

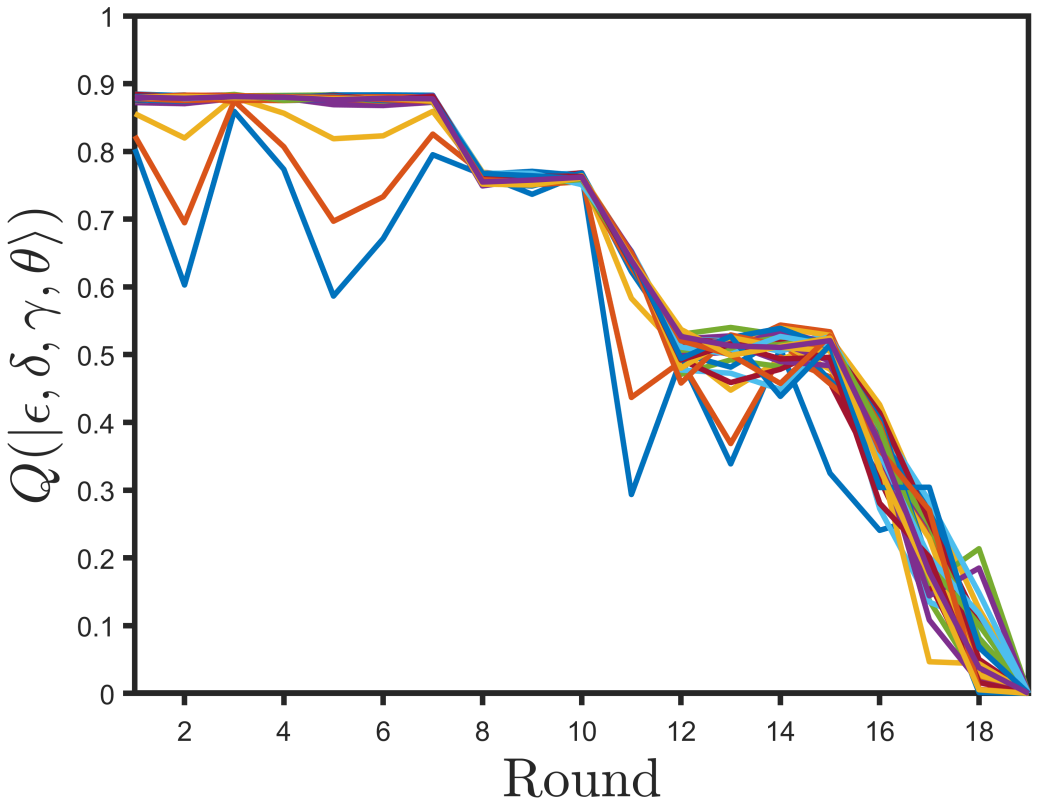

Figure 3.7: Entanglement of the ansätze generated by the quantum computer for each round of the algorithm at each θ value. In this case, the energy spectrum was Gaussian-distributed energies. Each line in the plot represents one of the 32 θ values.

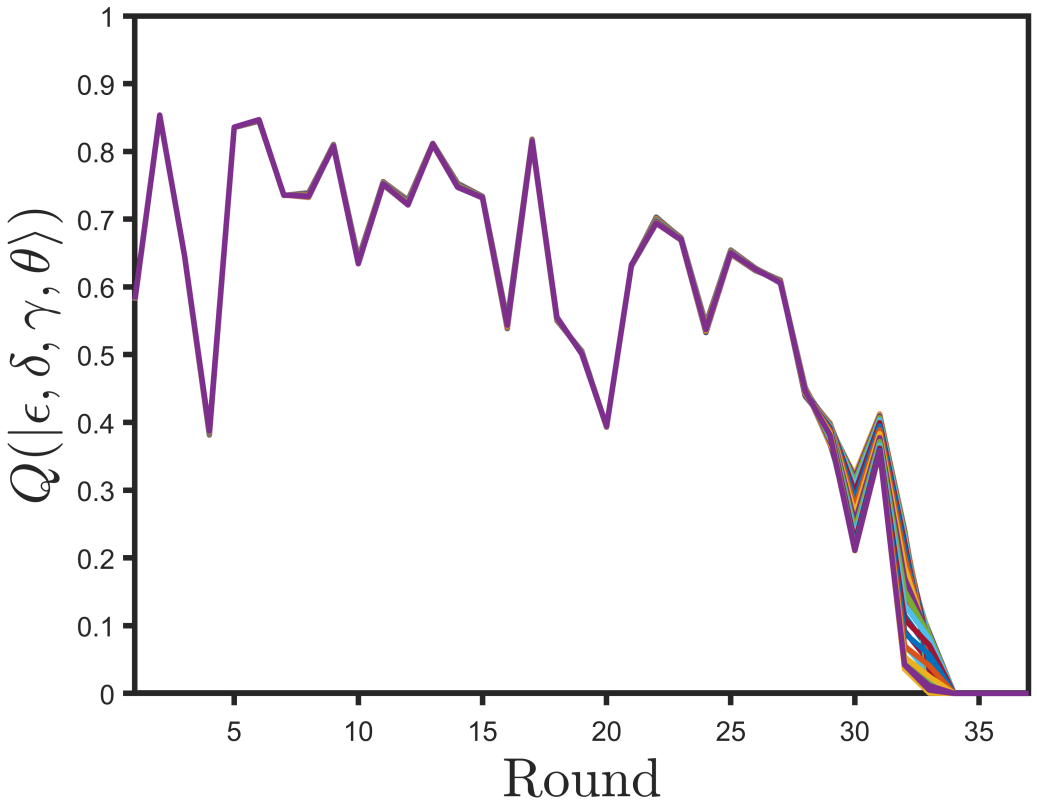

Figure 3.8: Entanglement of the ansätze generated by the quantum computer for each round of the algorithm at each θ value. In this case, the energy spectrum was $E_n = -1/n^2$. Each line in the plot represents one of the 32 θ values.

CHAPTER 4: CONCLUSIONS

The overall objective of this work was to implement the Divide and Conquer Operators in a VQA which leverages their unique halving property, as well as making use of their sparsity and their being involutory. To do so, issues relating to the completeness of the operator pool had to be understood and addressed. To this end, we established the incompleteness of the DACO pool, while developing two supplementary sets of operators to "complete" the pool. The structure of these new associated operators led to multiple new frameworks for the VQA ansatz to be implemented in the DACO-VQA. The performance of each of these ansätze were compared, and a final ansatz was selected which specifically leverages the halving property of the DACOs. In addition, the final DACO-VQA ansatz can be combined with other operator ansätze, provided confinement is satisfied.

Next, the entangling capability of different ansätze was considered, and a correspondence was found with higher entangling capability and lower sampling of the Hilbert space. This connection was also supported by tracking the entanglement of the states generated by the quantum computer over the rounds of the VQA. Despite this, no direct relation between entangling capability and Hilbert space sampling has been identified, so future work will be dedicated to exploring this connection more deeply.

Finally, the performance of the algorithm was characterized for different Hamiltonians. The DACO-VQA appears to be rather robust to different Hamiltonians, but seems to favor spectra which have clustered low energy states, rather than spectra with a large gap separating the ground state and first excited state. These differences, however, are minimal, and the algorithm proved successful across the board. Despite

the success of the algorithm in identifying ground states of various Hamiltonian systems, the algorithm still oversamples the Hilbert space. This is less of a problem than it seems, as no known VQA is able to undersample the Hilbert space while consistently identifying the ground state of the system (and many algorithms employ theoretical averaging, skirting the measurement problem altogether). Our results also indicate that there may be a regime in which the algorithm undersamples the Hilbert space, but reaching this regime is computational difficult, and it is not clear whether the observed trend of decreasing sampling will continue through to larger numbers of qubits. We also observed that with the current implementation of replicas, there is no effect on the performance of the algorithm.

With all of these findings in mind, there is a large body of future work which can be conducted. First, the connection between entangling capability of ansätze and performance of the VQA can be explored to identify whether or not the seeming correlation is coincidence. The role of entanglement is obviously important, but exactly how it influences the success of VQAs is a mystery. Shedding light into this will prove valuable not only to the DACO-VQA, but to the broader field of VQAs as well.

Beyond this, the concept of replicas is still worth pursuing, despite their lack of success in this implementation. The possibility of parallelizing quantum computations with limited increases in the number of qubits required is incredibly promising, provided replicas can be introduced with interactions. In line with this, future work will be dedicated to identifying different ways to generate replicas in the quantum computer.

Another possibility for improving algorithmic performance is Trotterization. Trotterization of a VQA ansatz essentially generates multiple new parameter values to optimize over, allowing more freedom to explore the Hilbert space to locate the ground state. Previous work has shown that Trotterizing the original QAOA ansatz led to an

improvement in the performance of the algorithm, so Trotterizing the DACO-VQA ansatz is worth pursuing.

Lastly, given the ability of the DACO-VQA ansatz to integrate with other ansätze (provided confinement is satisfied), it could be highly beneficial to implement other successful VQA ansätze by integrating them with the DACO-VQA ansatz. Given the low Hilbert space sampling observed with the "canonical" ansatz ordering implemented in this work, it is possible that other ansatz structures will lead to undersampling of the Hilbert space.

REFERENCES

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. Bardin, R. Barends, R. Biswas, S. Boixo, F. Brandao, D. Buell, B. Burkett, Y. Chen, J. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. M. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. Harrigan, M. Hartmann, A. Ho, M. R. Hoffmann, T. Huang, T. Humble, S. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. MandrÃ , J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, p. 505â510, 2019.

[2] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, *et al.*, "Noisy intermediate-scale quantum algorithms," *Reviews of Modern Physics*, vol. 94, no. 1, p. 015004, 2022.

[3] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[4] A. J., A. Adedoyin, J. Ambrosiano, P. Anisimov, A. BÃ€rtschi, W. Casper, G. Chennupati, C. Coffrin, H. Djidjev, D. Gunter, S. Karra, N. Lemons, S. Lin, A. Malyzhenkov, D. Mascarenas, S. Mniszewski, B. Nadiga, D. O'Malley, D. Oyen, S. Pakin, L. Prasad, R. Roberts, P. Romero, N. Santhi, N. Sinitsyn, P. J. Swart, J. G. Wendelberger, B. Yoon, R. Zamora, W. Zhu, S. Eidenbenz, P. J. Coles, M. Vuffray, and A. Y. Lokhov, "Quantum algorithm implementations for beginners," 2018.

[5] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, 2000.

[7] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, 2017.

[8] A. G. Taube and R. J. Bartlett, "New perspectives on unitary coupled-cluster theory," *International journal of quantum chemistry*, vol. 106, no. 15, pp. 3393–3401, 2006.

[9] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. Oâbrien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, pp. 1–7, 2014.

[10] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[11] S. Hadfield, Z. Wang, B. Oâgorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, 2019.

[12] C. Zalka, "Simulating quantum systems on a quantum computer," *Proceedings: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 313–322, 1998.

[13] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[14] D. H. Wolpert, W. G. Macready, *et al.*, "No free lunch theorems for search," tech. rep., Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[15] M. J. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in optimization and numerical analysis*, pp. 51–67, Springer, 1994.

[16] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[17] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964.

[18] J. C. Spall *et al.*, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE transactions on automatic control*, vol. 37, no. 3, pp. 332–341, 1992.

[19] R. Fletcher, "A new approach to variable metric algorithms," *The computer journal*, vol. 13, no. 3, pp. 317–322, 1970.

[20] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[21] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving," *Journal of research of the National Bureau of Standards*, vol. 49, no. 6, p. 409, 1952.

[22] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta numerica*, vol. 4, pp. 1–51, 1995.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] A. Pellow-Jarman, I. Sinayskiy, A. Pillay, and F. Petruccione, "A comparison of various classical optimizers for a variational quantum linear solver," *Quantum Information Processing*, vol. 20, no. 6, pp. 1–14, 2021.

[25] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, "An adaptive variational algorithm for exact molecular simulations on a quantum computer," *Nature communications*, vol. 10, no. 1, pp. 1–9, 2019.

[26] H. L. Tang, V. Shkolnikov, G. S. Barron, H. R. Grimsley, N. J. Mayhall, E. Barnes, and S. E. Economou, "qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor," *PRX Quantum*, vol. 2, no. 2, p. 020310, 2021.

[27] P. R. Halmos, *Finite-dimensional vector spaces*. Courier Dover Publications, 2017.

[28] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019.

[29] D. A. Meyer and N. R. Wallach, "Global entanglement in multiparticle systems," *Journal of Mathematical Physics*, vol. 43, no. 9, pp. 4273–4278, 2002.

[30] G. K. Brennen, "An observable measure of entanglement for pure states of multi-qubit systems," *arXiv preprint quant-ph/0305094*, 2003.