The University of Melbourne

Simulating Noisy Quantum Algorithms and Low Depth Quantum State Preparation using Matrix Product States

Author
Azar Christian Nakhl

Under the Supervision of
Dr. Charles Hill

Presented in fulfilment of the requirements of the degree of Master of Science.
November 12, 2021

Abstract

Since the proposal of Quantum Computation in the 1980s, many Quantum Algorithms have been proposed to solve problems in a wide variety of fields. However, due to the limitations of existing quantum devices, analysing the performance of these algorithms in a controlled manner must be performed classically. The leading technique to simulate quantum computers classically is based on the Matrix Product State (MPS) representation of quantum systems. We used this simulation method to benchmark the noise tolerance of a number of quantum algorithms including Grover's Algorithm, finding that the algorithm's ability to discern the marked state is exponentially suppressed under noise. We verified the existence of Noise-Induced Barren Plateaus (NIBPs) in the Quantum Approximate Optimisation Algorithm (QAOA) and found that the recursive QAOA (RQAOA) variation is resilient to NIBPs, a novel result. Also integral to the performance of quantum algorithms is the ability to efficiently prepare their initial states. We developed novel techniques to prepare low-depth circuits for slightly entangled quantum states using MPS. We found that we can reproduce Gaussian and W States with circuits of $O(\log(n))$ depth, improving on current best known results which are of $O(n)$.

State of Contribution and Originality

I declare that

- Chapter 1 provides an introduction to the field and discusses work that is well established in the literature. The review is original in its formation but maintains the conventions established in the referenced texts.
- Sections 2.1, 3.1, 3.3.1, and 4.1 provide outlines of existing algorithms and procedures but are original in their formation. These sections provide context for the original works that follow in the upcoming sections of their respective chapters.
- All other sections in Chapters 2, 3 and 4 are original unless otherwise stated

I also declare that the code produced for the purposes of this thesis are the author's own with all external libraries used cited in this thesis.

Azar Christian Nakhl
November 12, 2021

Contents

Chapter 1

Introduction

Quantum Computing has been a rapidly growing area of research since it was proposed in the 1980s by Benioff, Feynman and Manin[1, 2, 3]. The motivation underpinning this desire for a quantum simulator was that, in the words of Feynman "Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical"[2]. In the decades that followed many quantum algorithms have been proposed which solve problems ranging from chemistry[4] to finance[5] with theoretical speed and accuracy outperforming best known classical algorithms. As such, the development and analysis of quantum algorithms is a key area of research within quantum computing. Included in this is the efficient preparation of quantum states which form the starting point for many quantum algorithms[6]. However, as the capabilities of experimentally realisable quantum computing devices remains limited[7], classical simulation of quantum computers remains a key component in the analysis, validation and benchmarking of quantum algorithms. Matrix Product States (MPS) are a representation of quantum systems that have found significant application in the simulation of quantum computing as a result of their ability to represent and manipulate states of minimal entanglement in a space efficient manner[8]. In this thesis, we will analyse the performance of quantum algorithms under noisy conditions and develop novel techniques to efficiently prepare slightly entangled quantum states using MPS.

Returning to the fundamentals of quantum computing, there exists a number of ways in which a quantum model of computation may be realised. The quantum circuit model of computation is the most common model of general purpose quantum computers that is discussed and extends the familiar classical model of computation[9]. An equivalent to this is the Quantum Turing Machine[10] which extends Alan Turing's model of computation to states in a Hilbert Space under unitary evolution. Alternatively, the Adiabatic Quantum Computing[11] model is a separate model of quantum computation which uses the Adiabatic Theorem to solve problems encoded by a possibly complicated Hamiltonian through adiabatic evolution from a Hamiltonian whose ground state is simple to prepare. This model is particularly useful for optimisation type problems that may be easily mapped onto a Hamiltonian. For the entirety of this thesis, we will confine our discussion to that of the quantum circuit model of quantum computing.

1.1 The Quantum Circuit Model

In analogy with classical computation, the fundamental unit of information in the quantum circuit model of quantum computation is the qubit. The qubit is a 2-level quantum system comprising a 2-dimensional Hilbert Space with an ensemble of such systems forming a 2^n-dimensional Hilbert space. Hence, in addition to being able to store any binary string of size 2^n as per a classical computer, the quantum system may store a superposition of any number of binary strings. Acting upon these qubits, again in analogy with classical computation, are quantum gates. Unlike classical computation however where logical operations are non-reversible, quantum gates are

reversible. Lastly, measurement of any quantum mechanical system will result in a collapse of any superposition of the system and this holds for quantum computers just the same. This is starkly different to classical computation where measurement will generally not result in any disturbance of the state.

Theoretically, quantum information may be stored in some set of qubits in perpetuity in analogy with a theoretical classical bit. In practice, however, the limited coherence times of physically realisable qubits mean that this is not currently possible[7]. Furthermore, the no-cloning theorem[12] prevents the reproduction of quantum information which ultimately means that information on how to reproduce some quantum state must be stored classically. We discuss this topic further in Chapter 4 of this thesis.

1.2 Quantum Algorithms

Making use of this new model of computation there have been many algorithms proposed that can solve problems faster than classical computers. Broadly speaking there are a number of categories that most common quantum algorithms fall under[6]. The Quantum Fourier Transform is the quantum equivalent of the discrete Fourier Transform and gives rise to a number of common algorithms. This includes the Deutsch–Jozsa Algorithm, which was the first quantum algorithms proposed which provided an exponential speed-up over any possible deterministic classical algorithm, and Shor's Algorithm which can factor semi-prime numbers in $O(\log(N)^3)$. Algorithms based on amplitude amplification work by selecting and amplifying some subset of your Hilbert space and includes Grover's Algorithm which can search through an unordered dictionary in $O(\sqrt{N})$ time and is discussed extensively in Chapter 2.

Another category of quantum algorithm that has received a lot of attention recently are hybrid quantum-classical algorithms. Broadly speaking, these algorithms work by encoding some entangled trial state on a quantum device and using classical optimisation techniques to solve for the optimal state of the system. Hybrid quantum-classical algorithms include the Variational Quantum Eigensolver and the Quantum Approximate Optimisation Algorithm (QAOA) which is discussed much more extensively in Chapter 3.

Although one may naturally suspect that the properties of superposition and entanglement are the source of the supremacy of these quantum algorithms the extent to which this is the case is not yet known[13]. In general, demonstrating that an algorithm possesses true quantum supremacy is difficult as one must prove not only that it is better than existing classical algorithms, but also that it is better than any classical algorithms possible.

1.3 Experimental Realisation of Quantum Computers

The experimental realisation of quantum computing is a rapidly growing area of research and has even found some commercial success. At the forefront of this are the superconducting qubits based on the physics of Cooper pairs and Josephson junctions. Common implementations of circuit-based quantum computing such as those by IBM and Google, as well as D-Wave's adiabatic quantum computer use variations on the superconducting qubit[7] with the primary variation resulting from differences in connectivity of the qubits. Circuit based quantum devices are currently able to achieve systems on the order of 60 qubits[7, 14], with adiabatic quantum devices such as D-Wave having systems of over 1000 qubits[15]. Limited system sizes aside, the primary drawback of these quantum devices are gate infidelities and short coherence times which severely limits the algorithms that may be executed successfully. The term Noisy Intermediate-Scale Quantum (NISQ) computing has been coined to describe the current generation of quantum devices. A

common metric used to track the capabilities of NISQ era devices is Quantum Volume (QV) which tracks the maximum size of a square circuit that may be successfully executed on some given NISQ era architecture. As of 2020 IBM now has devices with a QV of 7×7.

In spite of this, in 2019 Google was able to demonstrate quantum supremacy using a 53 qubit device by sampling quantum random circuits[16], a problem which they claim would take 10,000 years on the best available classical computers available at the time. Although the problem instance for which quantum supremacy was demonstrated is not particularly useful. It demonstrated the ability for even NISQ era devices to demonstrate quantum supremacy if given the right problem.

1.4 Simulation

Given that real quantum devices are still in their infancy and unable to run even modestly sized quantum algorithms, a need exists to be able to simulate out quantum computers in order to validate the performance and characteristics of quantum algorithms. To this end, a number of different simulation techniques have been developed. Standard state vector simulators perform the unitary evolution of a quantum system using the standard linear algebra techniques used by everyday physicists to do quantum mechanics analytically[9]. For example, the equal superposition state may be represented and acted on as

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \tag{1.1}$$

$$Z\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} \tag{1.2}$$

Stabiliser simulators[17] are another type of classical simulator which work by representing a quantum state by the set of operators which map the state to itself. For example for the equal superposition state again,

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = I\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{1.3}$$

$$= X\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \tag{1.4}$$

Hence we can say that the group (X, I) stabilises the $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ state and uniquely represents the state. Then acting on the state with a Z gate as before gives the result (ZXZ, I) which stabilises the $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ state. It is a particularly useful representation for systems that involve only Clifford operators, where we recognise that $ABA^\dagger = B$[1] where A and B are elements of the Clifford group. Compared to state vectors that require $O(2^n)$ memory, this stabiliser representation requires only $O(n^2)$ memory provided that only Clifford operations are involved. Furthermore, gate execution time is $O(n)$ which is exponentially faster than ordinary state vector simulators. Measurement however requires $O(n^2)$ time. The stabiliser formalism is also particularly useful in quantum error correction where it is used to define the syndromes used to determine errors within logical qubits.

This thesis however will focus on another type of classical simulator known as the Matrix Product State (MPS) simulator based on the tensor network formalism. We will now provide you with an overview of the tensor network formalism which underpins these MPS simulations and a brief tour of quantum systems as Matrix Product States. Note that as we will be using pre-existing libraries which implement these tensor networks and all their features this introduction will only serve as a high-level overview and will omit a lot of technical detail. This information in the upcoming sections follows [18] and [19].

[1] Up to a global phase

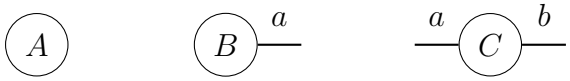


Figure 1.1: Diagrammatic representation of a scalar A, vector B_a and matrix C_{ab}

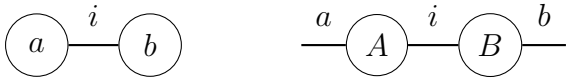


Figure 1.2: Diagrammatic representation of a Equations 1.5 and 1.6

1.5 Introduction to Tensor Networks

For the purposes of this thesis, a tensor may be thought of as an ordered bucket of scalars, and a generalisation of familiar vectors and matrices. An n-dimensional tensor is hence a bucket of scalars each indexed by n labels. Elements of a tensor are represented as $A_{u_0 u_1 \ldots u_n}$, where $u_0, u_1 \ldots u_n$ are the n labels that characterise all elements of the tensor. Diagrammatically, following [18], tensors may be represented as some given shape with lines emerging from it representing each of the n dimensions of the tensor as per Figure 1.1.

Operations on two or more tensors may be described by specifying how each of their elements interact. For example the vector dot product and matrix multiplication operations may be defined as,

$$\vec{a} \cdot \vec{b} = \sum_{i=0}^{i=N} a_i b_i = a_i b_i = c \tag{1.5}$$

$$\vec{A}\vec{B} = \sum_{i=0}^{i=N} A_{ai} B_{ib} = A_{ai} B_{ib} = C_{ab} \tag{1.6}$$

where $\vec{a}$ and $\vec{b}$ are vectors of size N, $\vec{A}$ and $\vec{B}$ are matrices of size $N \times N$ and where the third equality on each line utilises the Einstein summation convention. Diagrammatically we may represent the summation over indices by connecting lines corresponding to the indices to be summed over as per the example in Figure 1.2. This represents the most basic tensor network. These operations which combine two (or more) tensors into one are known as contractions.

1.5.1 Reshaping and Decomposing Tensors and Tensor Networks

One may reshape a tensor and redistribute its elements over a different set of indices. One may also introduce new indices of size 1 at will. This provides a mechanism by which the outer product of two tensors may be defined.

$$A_{ab} \rightarrow A_{abi} \quad B_{cd} \rightarrow B_{icd}$$
$$A \otimes B = A_{abi} B_{icd} = C_{abcd} \tag{1.7}$$

where we have introduced an index i to the matrices A_{ab} and B_{cd} of size 1. Diagrammatically, this may be represented as

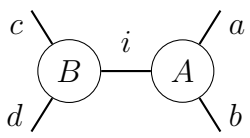


Figure 1.3: Diagrammatic representation of a Equations 1.7

The final major concept of tensor networks which we need to consider are decompositions

where a single tensor is decomposed into some tensor network. However, as we are able to reshape tensors, we do not need to develop from scratch methods to decompose general n-dimensional tensors, instead reshaping into a 2-dimensional tensor whereby standard matrix decompositions such as,

- SVD Decomposition: $A = U\Sigma V$ where U and V are unitary matrices and Σ is a diagonal matrix with real values ≥ 0 (the singular values)
- QR Decomposition: $A = QR$ where Q is some unitary matrix and R is an upper triangular matrix
- Eigendecomposition: $A = PDP^{-1}$ where P is a matrix containing the eigenvectors of A along the columns of the matrix and D is a diagonal matrix of the eigenvalues where the ith diagonal element corresponds to the ith eigenvector column in P,

after which the tensor may be reshaped back to its original form. We will see later that SVD is the best decomposition for the construction of our tensor network. Hence in the following section we will assume that the decomposition strategy is SVD.

1.5.2 Quantum Systems as Tensor Networks

Building upon this foundation, we may now construct a linear tensor network representation of a quantum system. For an ensemble of qubits of size N we typically represent the state of the system as a 1-dimensional tensor of size 2^N. We may reshape the vector to an n-dimensional tensor each of size 2 where each dimension corresponds to one qubit. We may then follow the schematic in Figure 1.4 to construct the MPS through repeated tensor decompositions. We restrict ourselves to SVD decomposition as it allows for a more straightforward analysis of the state. However, note that other decompositions may be used to construct the state albeit with perhaps some transformations required to take advantage of certain features which we will discuss.

The two types of tensors in our final MPS are,

- The qubit tensors: Each with an external bond of size 2 and an internal bond of size D. In our construction in Figure 1.4 when combined with their right bond tensor these form one of the orthonormal unitary tensors of the SVD decomposition.
- The bond tensors: Each with two internal bonds of size D representing the D non-zero singular values which result from the SVD decomposition. These tensors may be contracted into the left or right qubit tensor or left separate depending on the MPS implementation.

If other types of decompositions are used, for example, the QR decomposition, then the bond tensors would not exist as discrete entities and you would simply have each qubit tensor connected together directly with an internal bond.

The first thing benefit bestowed to us by using this representation comes about when you consider the SVD decomposition[2] of the state along some given subsystems A and B of the entire state. This decomposition may be written down as.

$$|\psi\rangle = \sum_{i}^{D} \lambda_i \left|\phi_i^A\right\rangle \left|\phi_i^B\right\rangle \tag{1.8}$$

[2]This way of framing the SVD decomposition is commonly referred to as a Schmidt decomposition in the literature. However as it is the same procedure, we will continue to refer to it as SVD

where A and B are the two bipartitions which we split the state along, $_i$ are the singular values and $\left|\phi_i^A\right\rangle$ and $\left|\phi_i^B\right\rangle$ are orthonormal basis for subsystem A and subsystem B respectively. The entropy of entanglement of the subsystems may then be written down as,

$$S(\rho_A) = S(\rho_B) = \mathrm{Tr}(\rho_A \log(\rho_A)) \tag{1.9}$$

$$\rho_A = \mathrm{Tr}_B(|\psi\rangle\langle\psi|) \tag{1.10}$$

$$= \sum_i^D |\lambda_i|^2 \left|\phi_i^A\right\rangle\!\left\langle\phi_i^A\right| \tag{1.11}$$

$$\implies S(\rho_A) = \sum_i^D |\lambda_i|^2 \log\left(|\lambda_i|^2\right). \tag{1.12}$$

Hence we can see that there is a direct correspondence to the singular values of the state's SVD decomposition and the bipartite entropy along some given subsystems of the entire state. Recognising that the bond tensors of our MPS construction also correspond to the singular values of the SVD decomposition, we see that states with low bipartite entropy will have small bond tensors. These states can then be said to be efficiently representable in MPS form as each qubit tensor will only require some low fixed number of values each. Hence the system will grow linearly with the number of qubits, unlike the state vector representation where the addition of qubits will necessarily result in an exponential increase in the memory requirements.

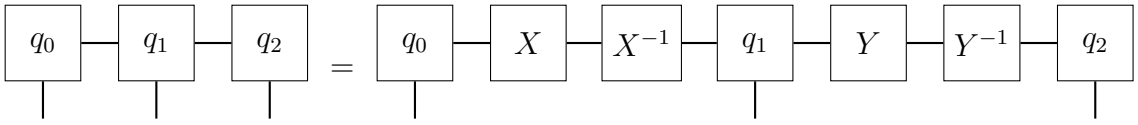
1.5.3 Truncation

In the event that the bond tensors required to reproduce a state are too large and the memory requirements too costly a procedure known as truncation may be used to provide an approximation to the state whilst reducing the overall resource requirements. Most implementations of SVD provide the singular values in descending order, this provides a natural way forward to approximate the state with as little loss in fidelity as possible given some provided maximum bond dimension. By eliminating the x smallest singular values and hence reducing the size of the bond tensor and its adjacent qubit tensors it can be shown that this provides the best approximation to the state as possible for a given bond dimension. One may think of this truncation operation as a kind of half measurement where the entanglement of the system has slightly collapsed. Hence, as we do for real measurements the entire state must be renormalised and recanonicalised.

Note that although this is a powerful feature of this simulation technique we do not use truncation at any point in this thesis. However, we do explore a similar analogue in the preparation of low depth circuits for arbitrary quantum states in Section 4.3.

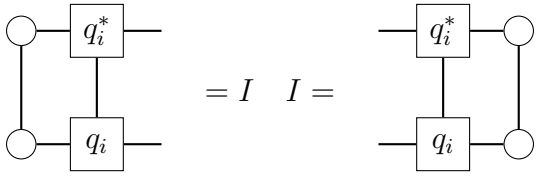
1.5.4 Canonical Form

States represented in MPS form have the following gauge freedom.



(1.13)

Hence it is typical to define canonical forms of MPS which satisfy certain conditions. The following are the typical canonicalisation conditions that one typically wishes to satisfy.



(1.14)

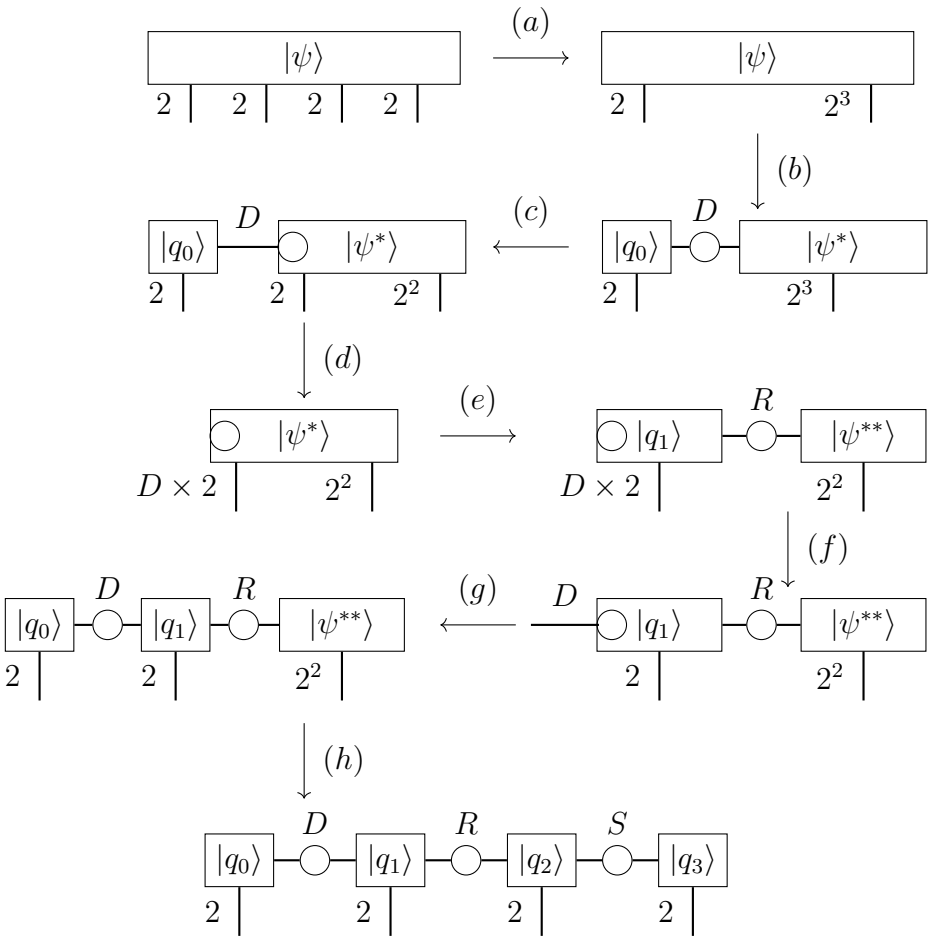


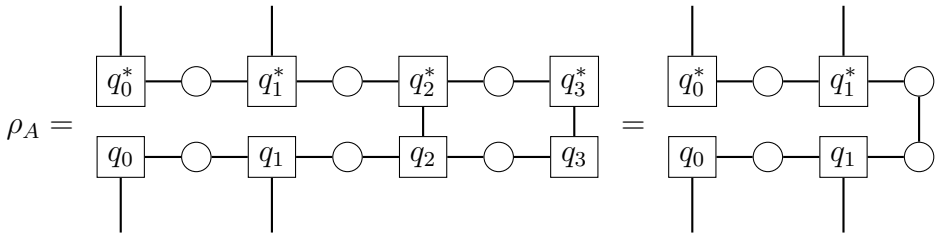
Figure 1.4: (a) The indices $q_1 \dots q_n$ are reshaped into a single index. (b) An SVD decomposition is performed resulting in a 2-tensor $|q_0\rangle$ of size $2 \times D$, a diagonal 2-tensor with D terms (the singular values) and a 2-tesnor $|\psi^*\rangle$ of size $D \times 2^{N-1}$. (c) The bond tensor is contracted into $|\psi^*\rangle$ and the 2^{N-1} sized index reshaped into two indices of size two and 2^{N-2} respectively. (d) The bond index and the index of size two are squeezed together. (e) An SVD decomposition is performed of $|\psi^*\rangle$. (f) The $D \times 2$ index is reshaped back into 2 indices of size D and 2 respectively. (g) The decomposed $|\psi^*\rangle$ tensor is returned to the original network with $|q_0\rangle$. (h) Steps (d)-(g) are repeated for $|\psi^{**}\rangle$ to obtain the final tensor network.

where the qubits at the end of the network will have the bond tensors (circles) omitted. These are known as the left and right canonicalisation conditions respectively. Note that if the MPS is formed using the procedure outlined in Figure 1.4 then the left canonicalisation condition is automatically met as the SVD decompositions which product orthogonal unitary operators were performed with left bond tensors contracted into the right. It is possible to enter the state into both left and right canonical form simultaneously by performing pairwise contractions and SVD decompositions along the entire state from left to right, and then right to left. [19] refers to this procedure as sweeping the MPS.

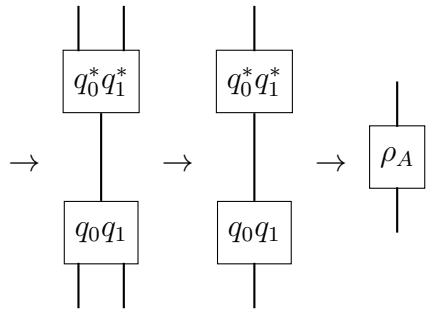
A very important consequence of this canonicalisation is that it allows for the computation of reduced density matrices, and hence measurement, locally. Recall that the reduced density matrix over some subset of qubits which form subsystem A is,

$$\rho_A = \mathrm{Tr}_B(|\psi\rangle\langle\psi|). \tag{1.15}$$

Hence for a state in MPS form we follow the procedure in Figure 1.5 to determine the reduced density matrix over the first two qubits as an example, where we see that ρ_A could be determined



(1.16)



(1.17)

Figure 1.5: Procedure to determine the reduced density matrix ρ_A for a state in MPS form

entirely from the qubit tensors (and surrounding bond tensors) in subsystem A. Hence the measurement probabilities p_i for qubits $q_i \in A$ can be determined entirely locally by inspecting the diagonal elements of ρ_A.

Sampling from the probability p_i, we can simulate measurement by collapsing the qubit(s) into the sampled state. This may be done by shrinking the qubit tensor's external index to one dimension corresponding to the measured value, throwing out the results for the non-measured value, and appropriately normalising the tensor. This is then propagated to the rest of the system by sweeping outward from the qubit. Finally, the qubit tensor's external dimension may be restored with zeros everywhere along that dimension. The shrinking of the qubit tensor to one helps reduce the overall cost required during the sweeping of the MPS as values that would otherwise be zero do not have to continue to be propagated through the system. That said this measurement operation still possesses an $O(n)$ time cost as a result of the outward sweep.

1.5.5 Unitary Operations on MPS

Having prepared the MPS of some quantum system and determined how it may be measured and stored efficiently, the final aspect that needs to be considered in order to use MPS as a simulator is the evolution of the quantum system by unitary evolution. Typically in the gate based model of quantum computation, unitary operations are relatively local, with only one or two qubit operations being available. Under the state vector simulation method, evolution of the system must occur universally regardless of the operation being performed. With MPS, local operations may be performed locally.

Single qubit operations are rather simple with the unitary operator forming a 2×2 2-tensor which may be contracted directly into the qubit tensor of the MPS as per Figure 1.6a. Larger operators are more complicated however. Assuming the operation acts on nearest neighbour qubits only, one approach is to simply contract the qubits together and squeeze their indices before contracting the $2^d \times 2^d$ 2-tensor into d-level qubit (qudit). This qudit may then be reshaped back into its original form and decomposed to reform the MPS. If the MPS was in canonical form, this approach maintains that by virtue of the fact that the contraction and decomposition

(a) A H gate applied to qubit q_0

(b) A CX gate applied to qubits q_0 and q_1

Figure 1.6: Example of one and two qubit operations on an MPS

steps maintain canonical form. An example of this procedure for a 2-qubit operation is shown in Figure 1.6b. A second approach based on Matrix Product Operators (MPOs) which are the operator analogue of MPS also exists for these multi-qubit operations. However, this approach is generally not useful for 2-qubit operations as the number of contractions required exceeds that of the approach already discussed. As our simulations seek to mimic real quantum devices as much as possible, we will not be making use MPOs for gate operations.

For multi-qubit operations which are not nearest neighbour we note that the way forward in this case is to swap the qubit tensors until they are adjacent then apply the operator after which the qubit tensors may be swapped back. To achieve one may either apply a sequence of 2-qubit SWAP gates or perform a sequence of contractions, reshapes and decompositions. Hence these non nearest neighbour operations are very costly and should generally be avoided in MPS simulations.

1.6 Tensor Network & MPS Software Packages

A number of different software packages exist which utilise MPS to simulate quantum circuits. IBM's open-source quantum computing package Qiskit[20] includes a number of classical simulators, including one which utilises MPS. In addition to this, dangMPS, a library developed at The University of Melbourne allows for the construction of large scalable MPS quantum circuits over many computing nodes within a High Performance Computing (HPC) cluster[19]. Software packages also exist which allow for the creation of arbitrary tensor networks such as open-source libraries Quimb[21] and Google's TensorNetwork[22]. In this thesis we will benchmark dangMPS, Qiskit and Quimb in Chapter 2 and additionally use all three at various points throughout the thesis as appropriate.

1.7 Outline of Thesis

In this thesis, we will use MPS to explore problems in quantum computing and quantum information theory. Specifically, we will simulate quantum algorithms under noisy conditions and approach the task of low depth initial state preparation. We will develop a noise model tailored for MPS simulators and apply it to Grover's Algorithm, hence determining the noise tolerance of the algorithm. We will then investigate a hybrid quantum-classical algorithm, the Quantum Approximate Optimisation Algorithm (QAOA) and its recursive adaptation RQAOA, investigating the susceptibility of RQAOA to Noise-Induced Barren Plateaus. We will then approach the task of determining low-depth circuits for initial quantum states and develop novel procedures which outperform best known methods for states of limited entanglement.

Chapter 2

Grover's Algorithm

Grover's Algorithm, proposed in 1996 by Lov Grover[23], is an algorithm that performs an unordered search through a dictionary in $O(\sqrt{N})$ time. Classically, no algorithm can guarantee a solution in better than $O(n)$ time, hence Grover's algorithm provides a quadratic improvement over the best known methods. Grover's Algorithm has applications including solving constraint satisfaction problems and breaking symmetric-key cryptography, among others. It is noted however that in many applications of the algorithm, such as database searching, the loading of an initial state is required which in general may overcome any speed up. In Chapter 4 we will consider this task of efficient initial state preparation.

We will introduce Grover's Algorithm and how it may be implemented on a Quantum Computer in Section 2.1 and 2.2 and use it to benchmark a number of MPS simulators in Section 2.3. We will then introduce common noise models for NISQ devices in Section 2.4.1 and determine a noise model which reflects this and suits MPS simulations. Finally, we will consider the noise tolerance of Grover's Algorithm in Section 2.5.

2.1 Outline of Algorithm

The outline of the algorithm is as follows. Suppose we have a system of size $N = 2^n$ and a desired marked state x. To begin we assume that the prior probability that any state is the desired state is equal. To reflect this we put the system into an equal superposition of all states,

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \,, \tag{2.1}$$

where i is a bitstring of a number between 1 and $N-1$

We then "mark" the desired state by applying an oracle $f(x)$ which applies a π phase rotation to state $|x\rangle$ and leaves all other states unchanged. After performing this operation the marked state will have an amplitude of $\frac{-1}{\sqrt{N}}$ with the remaining states keeping their original amplitude of $\frac{1}{\sqrt{N}}$. Sampling the system at this stage though will continue to result in a uniform probability distribution, to change this, the next step of the algorithm is to invert about the mean. Noting that the application of the oracle results in a decrease in the overall mean amplitude of the system, this inversion will result in an increased magnitude of the amplitude of the marked state as follows

$$\overline{|\psi\rangle} = \frac{1}{N} \sum_{i=0}^{N} \langle i|\psi\rangle = \frac{1}{\sqrt{N}} \tag{2.2}$$

$$\implies \overline{U_{f(x)} |\psi\rangle} = \frac{N-2}{N\sqrt{N}} \tag{2.3}$$

$$\implies \langle x| U_{inv} U_{f(x)} |\psi\rangle = \frac{2(N-2)}{N\sqrt{N}} + \frac{1}{\sqrt{N}} = \frac{3N-4}{\sqrt{N^3}} \tag{2.4}$$

where $U_{f(x)}$ is a unitary operator which encodes $f(x)$ and U_{inv} is the following operator

$$U = \begin{bmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} - 1 & \cdots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} - 1 \end{bmatrix} \tag{2.5}$$

which performs the mapping $s \mapsto \overline{U_{f(x)} |\psi\rangle} - s$ where s is some state of the system.

The application of the oracle and inversion operators is referred to as a Grover iteration and may be repeated in order to increase the probability of measuring the marked state. It is found that for large N, $\frac{\pi}{4}\sqrt{N}$ iterations are required to achieve a marked state probability of 100%.

2.2 Constructing the Circuit

In order to execute this algorithm on a simulator, we must first construct the operations of the algorithm using the standard gate set available to the simulator (or quantum device) we wish to use. Typically the gates available in quantum computing software packages are the single-qubit Pauli operators (X,Y,Z), the 2-qubit controlled-X operator (CX) in addition to the Hadamard (H), phase (S) and $\frac{\pi}{8}$ (T) operators. This gate set is universal, that is, any unitary operation that may be executed on a quantum device may be reduced down to the application of these operators. For simplicity, we will also allow ourselves the use of the arbitrary single-qubit rotation gate (U) as this is also typically available. The following is a typical gate set available on most quantum computing libraries including all three which we use in this thesis.

$$\begin{aligned} X &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & Y &= \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & Z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ H &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & S &= \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} & T &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \\ U(\theta, \phi, \lambda) &= \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda} \sin\left(\frac{\theta}{2}\right) \\ e^{i\phi} \sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)} \cos\left(\frac{\theta}{2}\right) \end{bmatrix} & CX &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned} \tag{2.6}$$

Given this, we can now follow the algorithm procedure step by step and construct the circuit which will execute the algorithm. We recognise that the Hadamard gate applied to all qubits will place the system in an equal superposition, hence satisfying the first step of the procedure,

$$H |0\rangle = \frac{1}{2}(|0\rangle + |1\rangle) \tag{2.7}$$

$$\implies H^{\otimes n} \left|0^{\otimes n}\right\rangle = \frac{1}{2^n} \sum_{i=0}^{2^n - 1} |i\rangle \,. \tag{2.8}$$

From here, we require an operator which conditionally applies a phase rotation for some given marked state. We can realise this as a controlled-Z like operation noting that we may flip any of the controls or targets from 1 to 0 by selective application of X gates at given sites. The controlled-Z gate looks as follows,

$$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} . \tag{2.9}$$

As the controlled-Z gate is not in the standard gate set, we need to determine a decomposition

for it. The Z gate performs a flip operation on the states $|+\rangle$ and $|-\rangle$, similarly to the X gate on the states $|0\rangle$ and $|1\rangle$. Furthermore, the Hadamard gate performs the transformation $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$. Given this, we may decompose the 2-qubit controlled-Z operator as $CZ_{0,1} = H_1 CX_{0,1} H_1$ where the subscripts denote the qubit(s) for which the operator is applied to.

2.2.1 Construction of multiply controlled gates

In order to run the algorithm for greater than two qubits, however, one requires the construction of a multiply controlled gate. The prototypical construction of the controlled-controlled-X gate, known as the Toffoli gate, requires 9 single-qubit gates and 6 2-qubit CX gates[9]. This is acceptable, however, 4-qubit Toffoli gates require at least 15 single-qubit gates and 16 2 qubit CX gates. Going beyond this, general decompositions can be shown to scale exponentially with the number of qubits[24].

An alternative to this is to introduce ancillary qubits into the system. This may be difficult in practice as NISQ era hardware is typically limited in the number of qubits available, however, in the context of limited quantum volume, this may in fact be the way forward. As for MPS simulations, they do not have such constraints as long as the bipartite entropy remains low. Recognising that the n-controlled-X gate performs the mapping $|q_0 q_1 \dots q_n\rangle \mapsto |q_0 q_1 \dots q_{n-1}(q_n \oplus (q_0 \wedge q_1 \wedge \dots q_{n-1}))\rangle$ we can construct an operator using $n-2$ ancillary qubits, sequentially applying logical AND between pairs of qubits

$$a_0 = q_0 \wedge q_1 = TOFF(q_0, q_1, a_0) \tag{2.10}$$

$$a_1 = a_0 \wedge q_2 = q_0 \wedge q_1 \wedge q_2 = TOFF(a_0, q_2, a_1) \tag{2.11}$$

$$\dots \tag{2.12}$$

$$a_{n-2} = a_{n-3} \wedge q_{n-1} = q_0 \wedge q_1 \wedge \dots q_{n-1} = TOFF(a_{n-3}, q_{n-1}, a_{n-2}), \tag{2.13}$$

where $q_0 \dots q_n$ represent the real qubits and $a_0 \dots a_{n-2}$ represent the ancillary qubits which are all initialised into the $|0\rangle$ state, and $TOFF(c_0, c_1, t)$ is a Toffoli gate with controls c_0, c_1 and target t. With the a_{n-2} qubit containing $q_0 \wedge q_1 \wedge \dots q_{n-1}$ we then perform a CX gate controlled on a_{n-2} and targeted on q_n to complete the construction of the n-controlled-X gate. As the Toffoli gate is Hermitian, and hence self-inverse, one may then reset the ancillary qubits by applying the Toffoli gates in reverse. Finally, one may convert the n-controlled-X gate to a n-controlled-Z gate by wrapping the q_n with Hadamard gates analogous to the construction of the 2 qubit CZ gate. An example of this ancillary construction for a 3-controlled-Z gate can be seen in Figure 2.1.

2.2.2 Inversion about the mean

Now we need to construct the operator U_{inv}. Firstly we recognise that we can write this operator down in terms of a generalised $|+^{\otimes n}\rangle$ state as follows,

$$U_{inv} = 2\left|+^{\otimes n}\right\rangle\!\left\langle+^{\otimes n}\right| - I. \tag{2.14}$$

Wrapping this operator with some Hadamard gates then gives us,

$$\begin{aligned} H^{\otimes n} U_{inv} H^{\otimes n} &= H^{\otimes n} 2\left|+^{\otimes n}\right\rangle\!\left\langle+^{\otimes n}\right| H^{\otimes n} - H^{\otimes n} I H^{\otimes n} \\ &= \left|0^{\otimes n}\right\rangle\!\left\langle 0^{\otimes n}\right| - I \qquad (2.15) \\ &= -U_{f(0^{\otimes n})}, \qquad (2.16) \end{aligned}$$

where we have recognised that Equation 2.15 is a similar construction to that of the oracle for the $|0^{\otimes n}\rangle$ state. This construction does result in a global phase shift of π radians as evidenced by the overall minus sign in Equation 2.16, however global phase has no physical consequence hence it does not matter for the successful execution of the algorithm. Hence the inversion step is executed

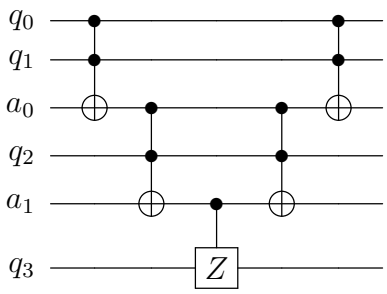


Figure 2.1: A 4-controlled-Z gate using 2 ancillary qubits

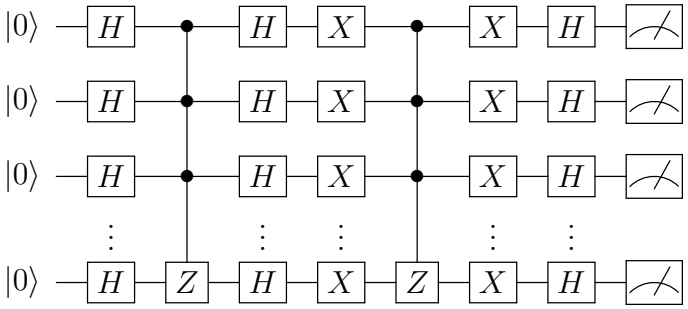


Figure 2.2: Grover's Algorithm Circuit with a $|111\ldots1\rangle$ target state

using a multiply-controlled-Z gate wrapped by some X gates to set the "marked" state to $|00\ldots0\rangle$ and then wrapped further by some Hadamard gates.

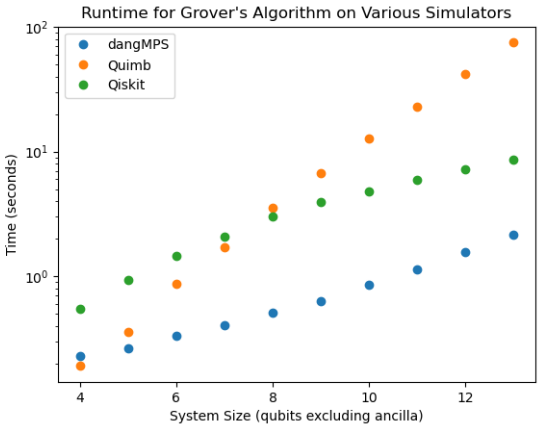


Figure 2.3: Comparing the runtime for Grover's Algorithm across various common libraries with MPS capability

Figure 2.2 shows the final construction for Grover's Algorithm that was executed on all simulators but without the ancilla qubits or the decomposition of the multiply controlled-Z gates. For simplicity, all simulations used a marked state of $|11\ldots1\rangle$ but as outlined in Section 2.1 the construction for oracles of different marked states is identical with simply some X gates wrapped around the oracle.

2.3 Comparing Different Simulators

The Grover's Algorithm circuit as prepared in Section 2.2 was set up on 3 different software packages, the internally developed dangMPS Simulator[19], IBM's Quantum Computing SDK Qiskit[20], and an open-source tensor network library Quimb[21]. In addition to the gates in Equation 2.6 all three simulators had the ability to execute controlled-Z gates and Qiskit additionally had the ability to encode Toffoli and multiply controlled gates as well, however in order to ensure the results of all three are comparable this in-built functionality was not used. The minimum system size was 4 qubits as below that one can simulate the entire algorithm trivially and without any ancillary qubits. Simulations were all run on a desktop computer with 4 cores at 3.9GHz and 16GB RAM.

Qubits	4	5	6	7	8	9	10	11	12	13
dangMPS	92.28%	99.84%	98.95%	98.30%	99.96%	99.75%	99.78%	100%	100%	100%
Qiskit	96.38%	99.91%	99.66%	99.66%	100.00%	99.91%	99.95%	100%	100%	100%
Quimb	96.16%	99.92%	99.60%	99.54%	99.99%	99.95%	99.95%	100%	100%	100%

Table 2.1: Success rate of Grover's Algorithm after 10,000 shots on 3 different MPS simulators

As expected all three simulators were capable of executing the algorithm successfully and were able to achieve a 100% success rate for all systems greater than 10 qubits[1]. The lowest success rate was found for the 4 qubit system across all 3 simulators. Table 2.1 shows the success rate across the 3 different simulators for systems up to 13 qubits

Figure 2.3 outlines the execution time for Grover's Algorithm across the three different simulators. The internally developed simulator outperforms the other simulators tested, however, the exponential growth of dangMPS and Qiskit are similar. Quimb performs exponentially worse than the other two simulators. The exponential nature of the growth can be attributed to the fact that the number of Grover iterations needed grows exponentially with the number of qubits. This underpins the fact that Grover's algorithm still grows as $O(\sqrt{N} = 2^{n/2})$, and the factor of two improvement in the exponent is simply not enough to counteract the exponential growth in the search space as you add more (qu)bits.

The speed and parallelisation ability makes dangMPS a good candidate for the execution of large circuits however its limited feature set means that its applicability is limited in scope. Qiskit has the ability to execute circuits using various simulation techniques as well as real quantum devices and has various noise models and algorithms in-built. Quimb allows for the construction and manipulation of arbitrary tensor networks and allows for easy visualisation of the network. Hence we will, at various points, rely on all three of these libraries throughout this thesis.

2.4 Noise Model

2.4.1 Noise in Quantum Devices

In order to simulate Grover's Algorithm under noisy conditions, we must first consider the different types of noise that quantum devices are prone to. NISQ era devices are subject to errors resulting from (i) gate infidelities, (ii) readout errors and (iii) decoherence resulting from interaction with the environment[25]. Single qubit gate infidelities may be modelled as a depolarising channel which can be represented as a combination of bit-flip errors and/or phase-flip errors. We can represent these errors using the Pauli operators as follows,

$$\epsilon_{\text{bit-flip}}(\rho) = p_{\text{bit-flip}} X\rho X + (1 - p_{\text{bit-flip}})\rho \tag{2.17}$$

$$\epsilon_{\text{phase-flip}}(\rho) = p_{\text{phase-flip}} Z\rho Z + (1 - p_{\text{phase-flip}})\rho, \tag{2.18}$$

where ρ is the density matrix of the quantum state and the ps define the probability of a phase-flip or bit-flip error occurring on a given qubit. Similarly, readout errors may also be modelled as bit-flips as per Equation 2.17. In addition to this, there are also two-qubit gate infidelities which may be modelled as a depolarising channel applied to the target qubit(s).

The final source of error, decoherence from environmental interactions, is the primary source of error on NISQ era devices and is a combination of thermal relaxation/excitation and phase decoherence. These interactions can be thought of as perturbations orthogonal to the (x,y) plane

[1]For the remainder of this chapter, a system of size n qubits does not include the number of ancillary qubits unless explicitly specified

and z axes of the Bloch sphere respectively. Unlike gate infidelities and readout errors, the system is guaranteed to experience decoherence hence it is a matter of when, not if, the system will breakdown. The decoherence is modelled as an exponential decay parameterised by two times T_1 and T_2 as follows,

$$\epsilon_{T1}(\rho)e^{\frac{-t}{2T_1}}\rho + (1 - e^{\frac{-t}{2T_1}})X\rho X \tag{2.19}$$

$$\epsilon_{T2}(\rho)e^{\frac{-t}{2T_2}}\rho + (1 - e^{\frac{-t}{2T_2}})Z\rho Z. \tag{2.20}$$

Table 2.2 outlines the noise characteristics of one of IBM's quantum devices. As we can see, single-qubit errors are negligible with readout and 2-qubit errors contributing a greater percentage. Noting that the average gate execution time of a gate on ibmq_montreal is 422 ns. We can compute how many consecutive gates will result in an equivalent error rate to the CX error rate as follows,

$$\begin{aligned} \text{Gates} &= -85120 \times \log(1 - 0.01842)/422 \\ &= 3.75. \end{aligned} \tag{2.21}$$

Hence we can conclude that decoherence will be the overwhelming factor that any noise model will need to account for.

Readout Error	CX Error	X Error	$\sqrt{X}$ Error	T_1 (μs)	T_2 (μs)
2.63%	1.842%	0.0497%	0.0497%	85.12	71.32

Table 2.2: Noise Characteristics of ibmq_montreal (as of 30th September 2021)[14]

2.4.2 Noise Model for MPS Simulators

Given that depolarising and readout errors have a substantially smaller equivalent noise rate compared to thermal relaxation and dephasing the noise models we will construct will only emulate these forms of error. We constructed 3 noise models, one consisting of random application of S gates, one of Z gates and the last of measurement gates. The former two simulate dephasing with the last one simulating thermal excitation/relaxation. We hypothesised that the application of measurement gates has the ancillary benefit that it simplifies the MPS by collapsing the entanglement of the state. We found that all 3 models had a comparable effect on Grover's Algorithm. Section 2.5 will present results for the measurement gate noise model only.

2.5 Results

To determine the effect of Grover's Algorithm under noise, we require a benchmark that we can use to define success. We define this as $p_{\text{marked}} \geq 0.5$. As we are not in the position to fix the success rate a priori however, we scan over a range of noise rates. Firstly though, we normalise our noise rates by determining an equivalent number of noisy gates per circuit

$$N_\epsilon = (N_G p + N)\epsilon \tag{2.22}$$

where N_G is the number of gates per Grover iteration, p is the number of Grover iterations, N is the size of the system and ϵ is the noise rate. This means that for any given circuit we can ensure that there are the same number of noisy gates present.

We ran the noisy simulations on The University of Melbourne High Performance Computing cluster Spartan[26] with 32 3.7 GHz CPU cores and 64GB RAM for system sizes between 4 and 20 qubits with variable noise rates on dangMPS. Figure 2.4 shows the success rate for Grover's Algorithm for a range of N_ϵ, between 50 and 150 per circuit. In order to achieve 50% success, we can see that we can limit our scope to $60 < N_\epsilon < 100$ per circuit. We see that there is a linear

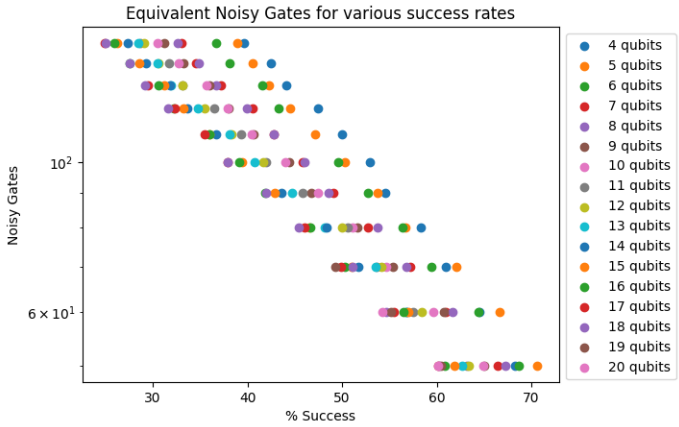


Figure 2.4: Success Rate of Grover's Algorithm for $50 \leq N_\epsilon \leq 150$

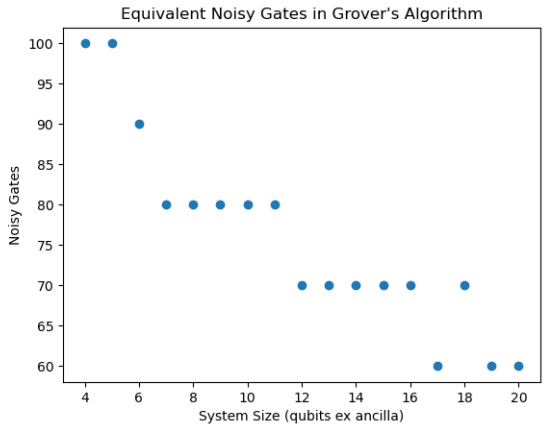


Figure 2.5: Equivalent Number of Noisy Gates, N_ϵ, for Figure 2.6

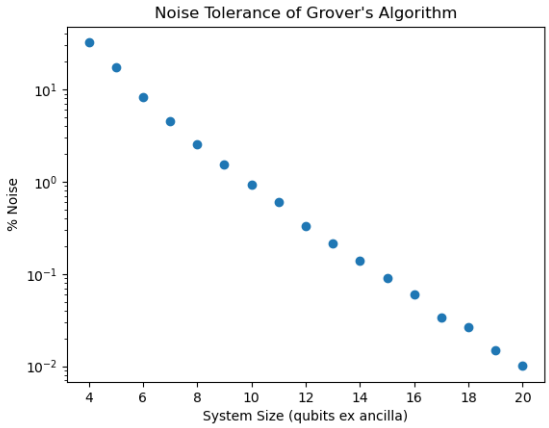


Figure 2.6: Noise Tolerance of Grover's Algorithm benchmarked as the highest noise rate possible for which $p_{\text{marked}} \geq 0.5$

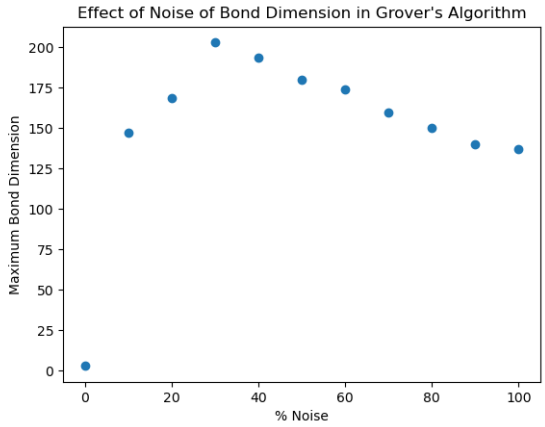


Figure 2.7: Effect of noise on bond dimension of a 10 qubit instance of Grover's Algorithm

relationship between the success of the algorithm and the number of noisy gates present in the circuit with a slight trend towards larger circuits being less tolerant towards noise as can be seen in Figure 2.5.

This result is quite devastating when it comes to the noise tolerance of Grover's Algorithm as the exponentially increasing depth of the circuit for larger systems will mean that the noise rates tolerated by the algorithm will be exponentially suppressed as can be seen in Figure 2.6.

Next, we determined the effect of our noise model on the bond dimension of the MPS. We found that the bond dimension of noise-free instances is very low with a uniform bond dimension of 2 across all qubits. As we introduced noise, however, we found that the bond dimension increased substantially as per Figure 2.7. We hypothesise that this occurs as a result of a distorted Toffoli gate that fails to reset some ancillary qubits hence generating entanglement across all qubits.

From this, we can conclude that Grover's Algorithm is a poor candidate to run on NISQ era hardware. Its low entanglement nature means that it may be possible to run very large instances of Grover's Algorithm on classical computers using MPS simulators, however, the exponential depth of the algorithm means that time will be a limiting factor even if memory is not.

Chapter 3

The Quantum Approximate Optimisation Algorithm

The Quantum Approximate Optimisation Algorithm (QAOA) is a variational quantum algorithm (VQA) designed to solve combinatorial optimisation problems[27]. It is a hybrid quantum-classical algorithm where the quantum device encodes a trial ansatz determined by the problem Hamiltonian and the classical computer varies the parameters according to some optimisation algorithm in order to determine the ground state. QAOA is particularly useful for solving Ising type problems and in particular problems in graph theory such as Max-Cut[28] and the Travelling Salesman Problem (TSP)[29]. Such combinatorial optimisation problems are considered to be NP-hard, hence best known classical approaches rely on approximation algorithms, such as the Goemans-Williamson approximation algorithm[30] for the Max-Cut problem. This classical approximation algorithm has a runtime of $O(Nm)$[31] where m is the number of edges and guarantees an approximation ratio of 0.8785[30] with any ratio above 0.9412 theorised to be NP-hard[32]. QAOA has a runtime of $O(Np)$ where p are the number of iterations needed which has been demonstrated to be sub-linear[33].

It has been shown that the parameterised ansatz of VQAs have landscapes that vanish exponentially under noisy conditions[34] creating what is coined as a Noise-Induced Barren Plateau (NIBP). We will simulate QAOA under noisy conditions and verify the presence of NIBPs for randomly generated graphs with 5 and 10 nodes respectively in Section 3.2. In addition to this, we will investigate a non-local adaptation to QAOA known as recursive QAOA (RQAOA) and explore the susceptibility to NIBPs for this algorithm in Section 3.3. Lastly, we will exploit the low connectivity of graphs of low bounded degree to run large instances of RQAOA using fewer qubits in Section 3.4.

3.1 Outline of Algorithm

3.1.1 Constructing the Ansatz

Consider a general constraint satisfaction problem with N constraints. The objective function of said problem will be of the form

$$C(x) = \sum_{i=1}^{N} C_i(x) \tag{3.1}$$

where x is an instance and $C_i(x) = 0$ if the constraint is satisfied by the instance and 1 otherwise. We then wish to minimise $C(x)$, that is, we wish to ensure that as many constraints are simultaneously satisfied by some given instance x as possible. Mapping this onto a problem we can solve on a quantum device, we construct the Hamiltonian for this objective function in the computational

(Z) basis as follows

$$H_P = \sum_i C_i \left|i\right\rangle\!\left\langle i\right| \tag{3.2}$$

where the is are bitstrings that encode each of the possible instances. We recognise that this Hamiltonian is diagonal in the computational basis and also note that it will typically involve only up to quadratic terms in Z. We can time evolve this Hamiltonian in the usual way[12],

$$H_P(t) = e^{-iH_P t/\hbar} \tag{3.3}$$

$$H_P(\alpha_j) = e^{-i\alpha_j \sum_i C_i |i\rangle\langle i|} \tag{3.4}$$

where, noting that the time-independent Hamiltonian H_P is strictly real, we have replaced time with some angle α_j. We now introduce a "mixer" Hamiltonian

$$H_M = -\sum_{i=1}^{N} X_i \tag{3.5}$$

$$H_M(t) = e^{-itH_M/\hbar} \tag{3.6}$$

$$H_M(\beta_j) = e^{i\beta_j \sum_{i=1}^{N} X_i} \tag{3.7}$$

with ground state $\left|s\right\rangle = H^{\otimes N}\left|0^{\otimes N}\right\rangle$. At this point, it is easy to see how we can solve this problem adiabatically. As the ground state of H_M is easy to prepare, we may set up a Hamiltonian to slowly interpolate from our mixer Hamiltonian to our problem Hamiltonian as follows,

$$\bar{H}(t) = (1-t)H_M + tH_P. \tag{3.8}$$

Provided we evolve slowly enough, by the adiabatic theorem[12], we will remain in the eigenstate of Equation 3.8 at all times. Hence at $t = 1$ our system will be in the eigenstate of H_P and hence we would have solved our constraint satisfaction problem.

However, we wish to map our problem onto the gate based model of quantum computation. Hence we restrict ourselves to the time evolved operators in Equations 3.7 and 3.4 and construct the following ansatz inspired from the adiabatic algorithm

$$\left|\alpha_j, \beta_j\right\rangle_p = \prod_{j=1}^{p} H_M(\beta_j) H_P(\alpha_j) \left|s\right\rangle \tag{3.9}$$

where $p \geq 1 \in \mathbb{Z}$. The expectation value of the objective function is then

$$S(p, \alpha_j, \beta_j) = \left\langle \alpha_j, \beta_j | H_P | \alpha_j, \beta_j \right\rangle_p \tag{3.10}$$

which may be minimised by some optimisation technique. It is obviously true that

$$\min(S(p, \alpha_j, \beta_j)) \geq \min(S(p+1, \alpha_j, \beta_j)) \tag{3.11}$$

and it is shown in [27] that,

$$\lim_{p\to\infty} \min(S(p, \alpha_j, \beta_j) = \min_{x\in\mathcal{H}} C(x). \tag{3.12}$$

Hence this ansatz is an appropriate choice for unconstrained binary optimisation problems.

3.1.2 Classical Optimisation

To determine for the α_j and β_j which minimise Equation 3.10 we rely on classical optimisation. As the gradient is not easily accessible we are constricted to gradient-free optimisation techniques such as the Nelder-Mead simplex method[35], COBYLA[36] or SPSA[37].

Nelder-Mead optimisation works by forming an n-dimensional simplex (with $n+1$ vertices) and traversing the cost landscape via reflection, expansion, contraction and pivoting of the simplex. This technique works well in general and is commonly used for VQAs. However, pivoting/shrinking the simplex can become a costly operation for large n. An alternative optimisation is the Con-

strained Optimisation by Linear Approximation (COBYLA) which similarly evaluates the cost function at the vertices of an n-dimensional simplex (commonly referred to as the "trust region") and linear interpolation is used to approximate the cost function within the trust region which can then be shrunk accordingly. COBYLA is a good algorithm for jagged cost landscapes and does not involve pivoting of the entire simplex hence it is not costly in that regard. Other optimisers such as Simultaneous Perturbation Stochastic Approximation (SPSA) exist which take a similar approach to simulated annealing and randomly perturb the parameters of the cost function. Stochastic methods such as SPSA generally require more iterations to converge but are generally quite resilient to poor local minima and noisy landscapes.

We will be using the ScipPy[38] implementation of these classical optimisation algorithms in our simulations.

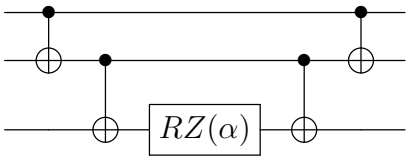
3.1.3 Executing the Algorithm

Given the QAOA ansatz and an appropriate classical optimiser, the steps to solve some given problem are as follows

1. Determine a Hamiltonian H_P in the computational basis that appropriately encodes the objective function you wish to optimise. This will typically be a linear combination of products of Z_i terms, where i denotes the qubit acted on by the Z operator

2. Construct a circuit that encodes the ansatz in Equation 3.9 with some fixed p

   - e.g. $Z_iZ_jZ_k$ couplings may be encoded as

   $e^{i\alpha Z_iZ_jZ_k} =$ 

   (3.13)

   extending and contracting in the usual way (similar to the MCZ gates of Chapter 2) for greater or smaller couplings
   - $e^{-i\alpha X} = RX(\alpha)$ and $e^{-i\alpha Z}$ can be implemented using the $U(2\theta, \pi/2, 3\pi/2)$ and $U(0, 0, \lambda)$ (up to a global phase) respectively using the definition of $U(\theta, \phi, \lambda)$ in Equation 2.6.

3. Classically optimise α_j, β_j with some appropriate gradient-free optimiser

4. Execute the circuit from Step 2 with the optimal angles and measure the final solution in the computational basis

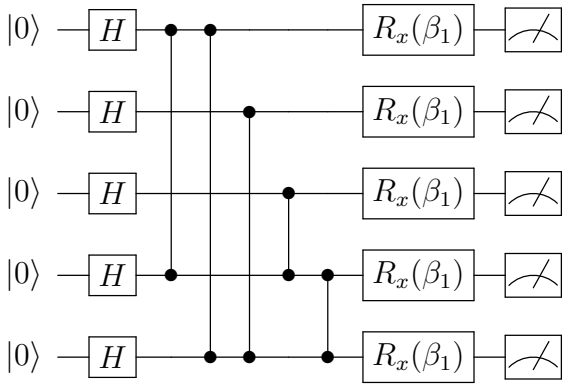
Note that it can be demonstrated that if p does not increase as a function of the problem size n then there exists a classical algorithm that can determine the optimal parameters for the problem[27]. This will be discussed further in Section 3.3.

3.1.4 An example: Max-Cut

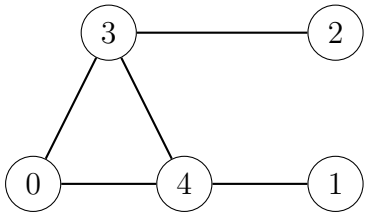
A common problem in graph theory that can be solved using QAOA is the Max-Cut problem. The Max-Cut problem involves the formation of two disjoint sets of vertices with as many connected edges as possible. Hence we can write down the Hamiltonian for a generic graph as follows,

$$H = \frac{1}{2} \sum_{(i,j)\in E} Z_iZ_j - I \tag{3.14}$$

a)



c)



b)

$$= ZZ(\alpha_1) = \begin{bmatrix} e^{-i\alpha/2)} & 0 & 0 & 0 \\ 0 & e^{i\alpha/2)} & 0 & 0 \\ 0 & 0 & e^{i\alpha/2)} & 0 \\ 0 & 0 & 0 & e^{-i\alpha/2)} \end{bmatrix}$$

d)

$$H = \frac{1}{2}\Big(Z_0Z_3 - I\Big) + \frac{1}{2}\Big(Z_0Z_4 - I\Big) + \frac{1}{2}\Big(Z_1Z_4 - I\Big) + \frac{1}{2}\Big(Z_2Z_3 - I\Big) + \frac{1}{2}\Big(Z_3Z_4 - I\Big)$$

Figure 3.1: (a) the $p = 1$ QAOA circuit encoding the Hamiltonian in (d). (b) The construction of the ZZ Ising coupling gate which makes up the mixer Hamiltonian. (c) An example graph of 5 nodes generated using Erdős–Rényi with $p = 0.5$. (d) the Hamiltonian encoding Max-Cut problem as per Equation 3.14 for the graph in (c)

where E is the set of edges in some graph G. We recognise that if i and j are in the same set then $Z_iZ_j = 1$ and no term will be added to the Hamiltonian, otherwise a -1 term will be added. Hence minimising this Hamiltonian will solve the Max-Cut problem for the graph G. We demonstrate this for an example graph of 5 nodes in Figure 3.1. For the remainder of this chapter, we will confine ourselves to solving the Max-Cut Hamiltonian, however, our analysis is equally applicable to any Hamiltonian that may be solved using QAOA or it's variants unless otherwise specified. For simplicity, we will also omit all factors of I in our simulations without loss of generality.

3.2 Noise-Induced Barren Plateaus

The ability to correctly optimise a Hamiltonian is the key to running VQAs such as QAOA successfully. Principally, having a training landscape with sufficiently large gradients is necessary in order to find the global minimum. For certain types of VQAs such as those with global cost functions it has been demonstrated that the gradient of the training landscape vanishes exponentially in n[39]. This does not affect QAOA, however, it has been shown that noise is another factor that may induce barren plateaus in the training landscape[34]. It has been shown that for any given ansatz and expectation operators, the gradient of the training landscape vanishes exponentially as $O(2^{L\log_2(q)})$ where L is the number of layers (i.e. p in our formulation of QAOA) and $q < 1$ is a noise parameter. Likewise, it has been demonstrated that the cost function concentrates exponentially towards that of the equal superposition state. Recognise that this is true regardless of the specific value of q, which will only ever affect the severity of the NIBP.

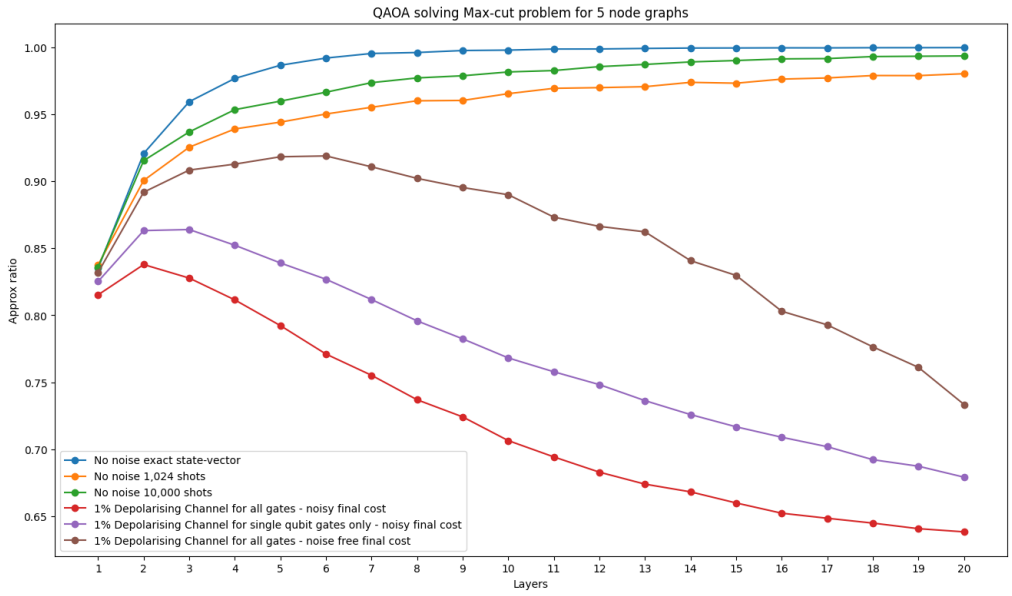


Figure 3.2: QAOA solving the Max-Cut problem on 100 random 5 node graphs

3.2.1 NIBPs in Max-Cut

To verify the presence of NIBPs in QAOA we follow [34] and solve the Max-Cut problem on 100 randomly generated graphs. We do so under both no-noise conditions and under a 1% depolarising channel. The graphs were generated with the Erdős–Rényi model[40] using the NetworkX library[41] with $p = 0.5$ being the probability of an edge being formed. Each instance of QAOA was repeated 10 times, with the best result recorded as the final result for a given graph. This allows for sufficient statistics to be collected. The COBYLA optimiser was used as the size of the systems was sufficiently small. The circuits were prepared using Qiskit's MPS simulator backend and run on the Spartan HPC with one CPU core per graph.

To measure the success of a given instance we define the approximation ratio of a given optimised instance of QAOA to be

$$\text{Approximation Ratio} = \frac{S(p, \alpha_j, \beta_j)}{\min_{x \in \mathcal{H}} C(x)} \tag{3.15}$$

where the minimum of $C(x)$ can be found classically for these small instances by determining the minimum eigenvalue of H_P, which we did using NumPy[42]. The results can be seen in Figure 3.2.

We see that QAOA instances suffer from NIBPs for even low depth instances regardless as to whether the noise applies to single or multiple qubit operators and regardless of noisy or noise-free final cost. Extending beyond the noise models analysed in [34], we also ran instances that sampled from the final no-noise quantum state, in these instances we expect the standard error of our measurement to be

$$\begin{aligned} \frac{\Delta C}{\bar{C}} &= \frac{\sigma}{\bar{C}\sqrt{N}} \\ &= \frac{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(C_i - \bar{C})^2}}{\bar{C}\sqrt{N}} \end{aligned} \tag{3.16}$$

which we find to be on the order of about 1.6% for 1024 shots. As we can see though, this kind of noise although having an impact on the overall final approximation ratio does not induce any kind of barren plateau. This makes sense as shot noise can be viewed as a form of measurement noise

which can be shown to have only have a linear decrease in n on the gradient of the landscape[34].

3.2.2 The Cost landscape of QAOA

We include a visual demonstration of the exponentially vanishing gradient resulting from NIBPs in Figure 3.3. We see that although the general landscape remains broadly the same, we recognise that the gap between the peaks and troughs is far smaller. Note that for this particular instance the approximation ratio determined with noise is 0.42 compared to 0.94 in the noise-free case.

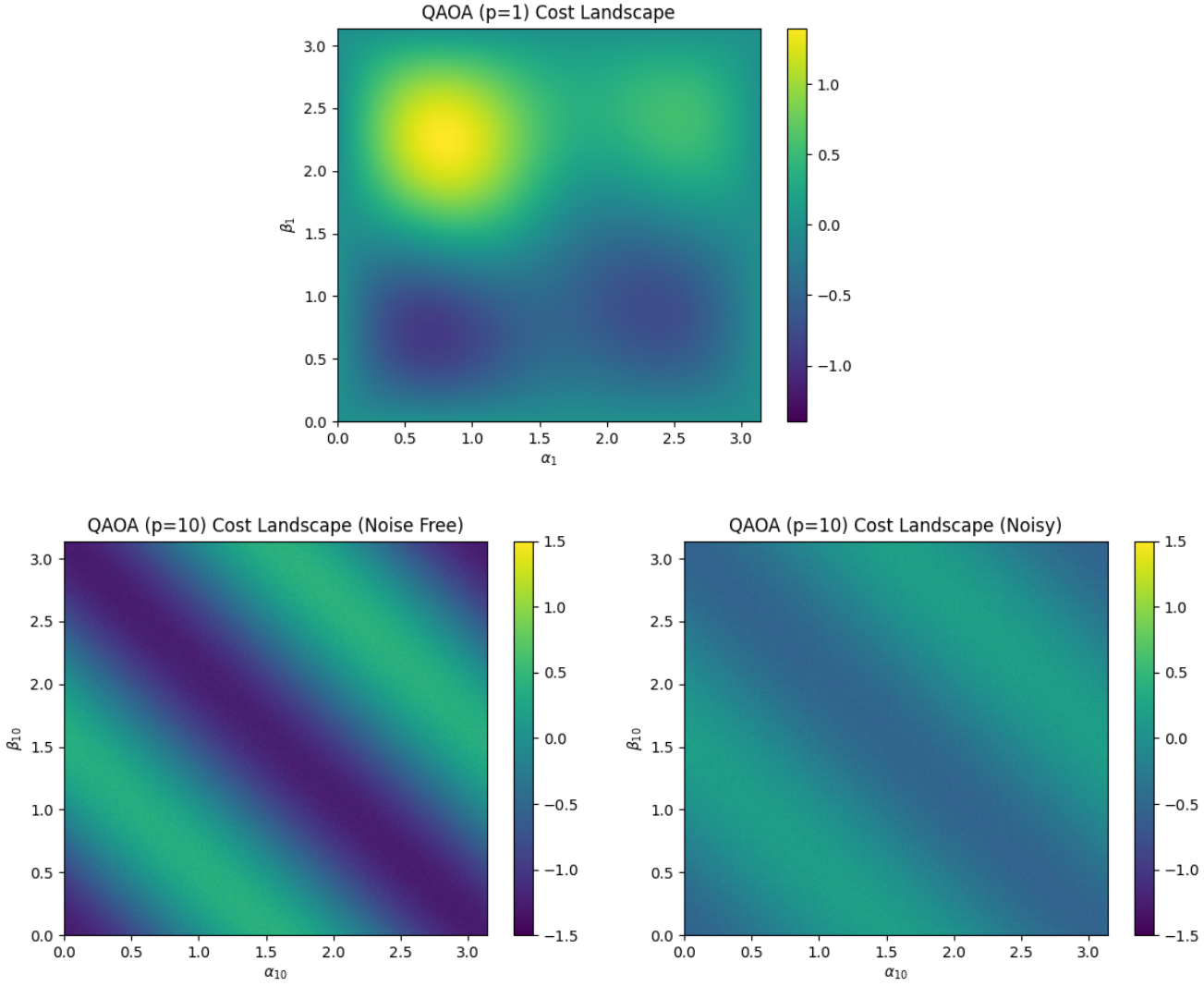


Figure 3.3: QAOA Cost landscape for the example in Figure 3.1 for $p = 10$. $\alpha_1 \dots \alpha_1 0$ and $\beta_1 \dots \beta_9$ are fixed to their optimal values

3.3 Recursive QAOA

We hinted earlier that QAOA has some limitations which may hamper any hopes of quantum advantage. Specifically, it is noted that the locality of the QAOA Hamiltonian means that for problems with some bounded degree there exists an upper bound on the approximation ratio that may be achieved[33].

A Hamiltonian is said to be local if each term involves $O(1)$ interactions only. This is separate from geometric locality where each interaction is physically proximal, e.g. Z_iZ_{i+1}. It is clear that QAOA is most likely local with respect to its Hamiltonian, however in general it won't be geometrically local. For local Hamiltonians on some bounded degree graph, it is found that each term has a sphere of influence of d^p and that p must have a lower bound of $O(\log(n))$ in order to beat the best known classical optimisations algorithms[33]. Functionally this means that the effective runtime of QAOA is likewise $O(\log(n))$. Note that this does not mean that an $O(\log(n))$ depth circuit will guarantee a solution, just that the solution will be closer than any classical optimisation algorithm.

3.3.1 Outline of Algorithm

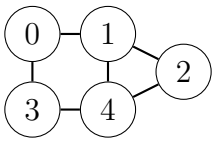
An alternative to standard QAOA which is non-local is proposed by [43] known as Recursive QAOA. It is constricted to $\mathbb{Z}_2$ symmetric Hamiltonians (i.e. of the form $\sum_{i,j\in E} Z_iZ_j$) and works as follows.

1. Prepare and optimise a standard level-p QAOA circuit for some given problem as per Section 3.1.3
2. For the problem Hamiltonian $H_P = \sum_{i,j\in E} Z_iZ_j$ and optimised α_i, β_i determine which $|\langle\alpha_i, \beta_i|Z_iZ_j|\alpha_i, \beta_i\rangle|$ is the largest
3. Fix these qubits to be correlated, $Z_i = Z_j$, if $\langle\alpha_i, \beta_i|Z_iZ_j|\alpha_i, \beta_i\rangle > 0$, or anti-correlated, $Z_i = -Z_j$, if $\langle\alpha_i, \beta_i|Z_iZ_j|\alpha_i, \beta_i\rangle < 0$ and eliminate Z_i from H_P
4. Repeat 1-3 with the new H_P on $n-1$ qubits
5. Once the problem size hits some threshold n_c stop and determine the solution to the n_c sized system classically (e.g. via brute force)
6. The solution to the n_c sized H_P can be propagated backwards to determine the solution to the full H_P.

This can be further adapted to Ising Hamiltonians with $n_{\text{interactions}} > 2$ as per [43] and can additionally be adapted to handle individual Z_i terms by the introduction of an ancillary qubit. We note that the run time for this circuit for $p = 1$ is $O(n)$ which appears to be worse than standard QAOA if indeed p being constant is sufficient. However, RQAOA vastly outperforms QAOA for all $p \in O(\log(n))$ in terms of correctness of solution. As the depth requirements appear to be low for RQAOA, it shows promise as a suitable algorithm to be run on NISQ era hardware.

3.3.2 An example

By way of example let's consider the following 5 node graph



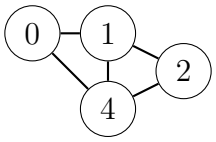
For Max-Cut the problem Hamiltonian takes the following form,

$$H_P = \frac{1}{2}(Z_0Z_1 - I) + \frac{1}{2}(Z_1Z_2 - I) + \frac{1}{2}(Z_2Z_4 - I) + \frac{1}{2}(Z_3Z_4 - I) + \frac{1}{2}(Z_1Z_4 - I) + \frac{1}{2}(Z_0Z_3 - I). \tag{3.17}$$

Performing QAOA as per Section 3.1.3 we find that

$$\text{argmax}_{i,j\in E}|\langle\alpha,\beta|Z_iZ_j|\alpha,\beta\rangle| = Z_0Z_3 \tag{3.18}$$

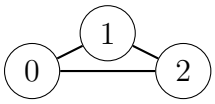
with $\langle\alpha,\beta|Z_0Z_3|\alpha,\beta\rangle \approx -0.47$. Hence we can fix this edge and eliminate one of the nodes as follows



Setting $Z_3 = \text{sign}(\langle\alpha,\beta|Z_0Z_3|\alpha,\beta\rangle)Z_0$, the Hamiltonian then becomes,

$$H = \frac{1}{2}(Z_0Z_1 - I) + \frac{1}{2}(Z_1Z_2 - I) + \frac{1}{2}(Z_2Z_4 - I) - \frac{1}{2}(Z_0Z_4 + I) + \frac{1}{2}(Z_1Z_4 - I). \tag{3.19}$$

Running QAOA on this reduced system we find that Z_0Z_4 is the most correlated bond with $\langle\alpha,\beta|Z_0Z_4|\alpha,\beta\rangle \approx 0.58$. Eliminating node 4 then gives the following system



$$H = (Z_0Z_1 - I) + \frac{1}{2}(Z_1Z_2 - I) + \frac{1}{2}(Z_2Z_0 - I). \tag{3.20}$$

The solution to this system is straightforward. It's $Z_1 = 1$, $Z_0 = Z_2 = -1$ (or vice-versa as the system is $\mathbb{Z}_2$ symmetric). Hence we can propagate our answer backward to determine the solution for the full graph,

$$Z_0 = -1, Z_1 = 1, Z_2 = -1 \tag{3.21}$$

$$Z_4 = Z_0 = -1 \tag{3.22}$$

$$Z_3 = -Z_0 = 1. \tag{3.23}$$

Which plugging into Equation 3.17 gives,

$$H_P = \frac{1}{2}(-1-1) + \frac{1}{2}(-1-1) + \frac{1}{2}(1-1) + \frac{1}{2}(-1-1) + \frac{1}{2}(-1-1) + \frac{1}{2}(-1-1) \tag{3.24}$$

$$= -4, \tag{3.25}$$

which is the correct ground state of that Hamiltonian.

3.3.3 Noise in Recursive QAOA

The question that has not yet been explored widely in the literature is, how susceptible is RQAOA to noise. To investigate this we performed a similar procedure to what was undertaken in Section 3.2.1. We generated 100 random connected 10 node graphs and ran both noisy and noise-free instances of RQAOA and QAOA using a Max-Cut Hamiltonian. The noise model was a 1% depolarising channel on both one and two-qubit gates and we repeated each graph instance 10 times as before. The simulations were run under the same environment and classical optimiser as Section 3.2.1. The results can be seen in Figure 3.4

As standard QAOA is a subroutine of RQAOA we expected to see some manifestation of the NIBP as seen in the previous section, however as we can see there is only a minimal drop off in the approximation ratio for large p. Additionally, we recognise that for the Max-Cut problem in particular we do not require p to be particularly large in order to achieve an approximation ratio of approximately 1. A possible explanation for this minimal drop off in approximation ratio is that because each QAOA subroutine only needs to determine the most maximally correlated term, our results will be more typical of a noisy training with noise-free final cost simulation than that of a completely noisy algorithm. This can also be seen in Figure 3.3 where we see that the

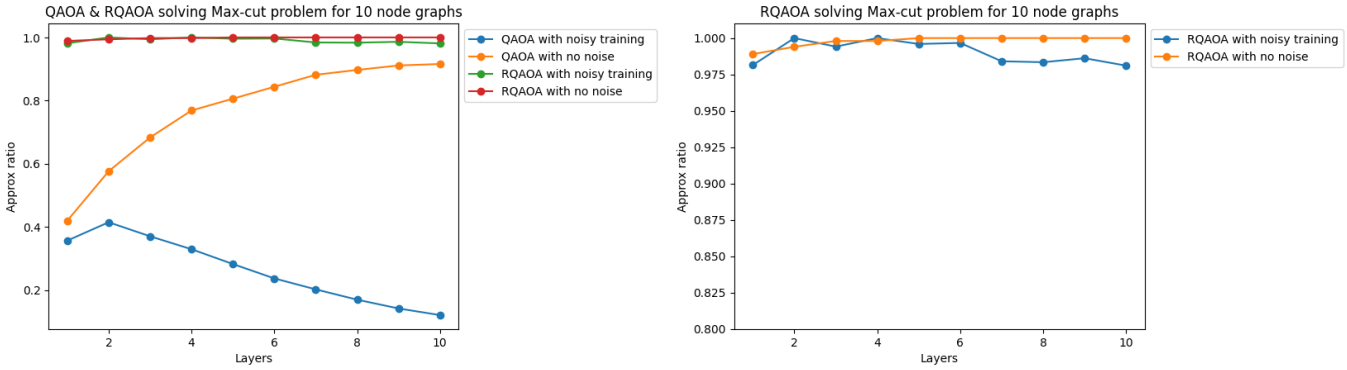


Figure 3.4: Average Approximation Ratio for RQAOA and QAOA for 100 10 node connected graphs as per Section 3.3

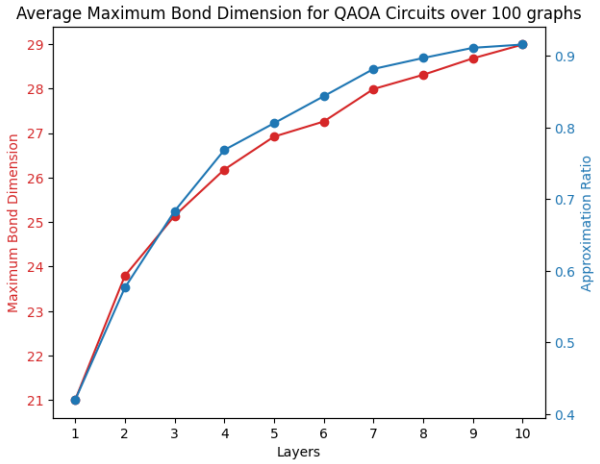


Figure 3.5: Average Maximum Bond Dimension for 10 qubit QAOA instances as per Section 3.3.3

minima are at the same location regardless of noise. This successful result aside, we still recognise that the overall trend is still indicative of a NIBP of some sort and it is likely that more complex Hamiltonians will exhibit poor performance as a result of NIBPs more readily than what we have observed with the Max-Cut problem.

3.3.4 Bond Dimension of QAOA and RQAOA Circuits

As we have seen, the success of QAOA is intimately tied to how much of the graph each interaction term "sees". We expect then, that the minimum entanglement required to successfully solve a graph instance is tied to the connectivity of the graph, and low entanglement instances of QAOA will not be successful for highly connected graphs in particular. To that end, we keep track of the maximum bond dimension for each QAOA instance run on 10 node graphs as per the previous section with results presented in Figure 3.5. Note that for RQAOA, the results as to the maximum bond dimension are identical to that of standard QAOA as it is a subroutine algorithm. As we can see, the approximation ratio for standard QAOA and the maximum bond dimension line up almost exactly which re-affirms the result that the QAOA circuit must see the whole graph in order to successfully determine the ground state of the problem Hamiltonian. This result also suggests that QAOA is not easily simulatable on MPS simulators at scale.

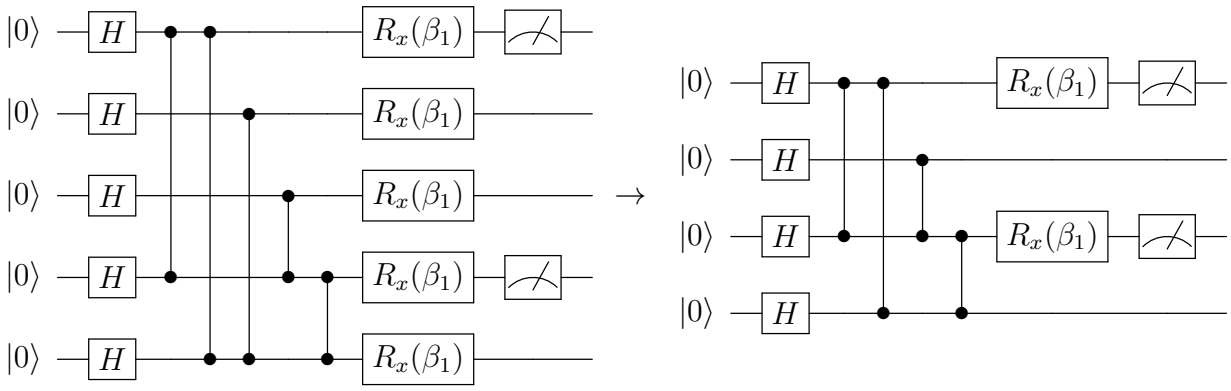


Figure 3.6: Circuit from Figure 3.1 where we want to only measure the Z_1Z_3 term of the cost function

3.4 Tailoring RQAOA to NISQ era devices

We have seen that RQAOA is resilient to the sort of noise we may expect in NISQ era devices. Hence our attention turns to strategies that would enable RQAOA to be run on NISQ era devices. In particular, the qubit requirements for larger systems is particularly prohibitive. With this in mind, we will explore techniques that may be combined to reduce the overall size of the circuits that need to be executed for a given problem instance.

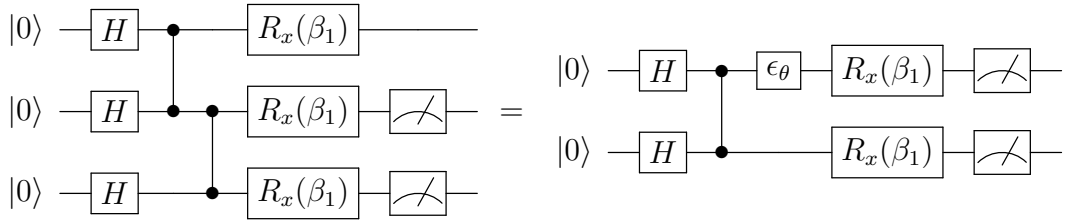
3.4.1 Removing Qubits outside the lightcone

The first technique relies on the fact that for any given term in our Hamiltonian we require only the measurement of the qubits for which that term directly acts on. As such qubits that are not in the lightcone of the qubits that we are measuring may be removed entirely.

The lightcone of a given qubit is defined to be the set of gates directly or indirectly correlated with the qubit of interest. For example in Figure 3.6 we see that qubit 2 has no gates which are correlated with the output of qubits 1 and 3 which are to be measured to determine the Z_1Z_3 coupling. Hence that qubit may be removed from the system entirely. It is important to recognise that within each $H_P(\alpha_i)$ each of the ZZ terms commute, hence even though it may look like the Z_1Z_5 term is indirectly correlated by virtue of the fact that qubit 5 is indirectly correlated, this is not the case as we can shift that term to the end of the circuit without loss of generality. This procedure is particularly useful for graphs of low bounded degree. We provide an example of this procedure in Section 3.4.3.

3.4.2 Leaf Node elimination

In addition to the procedure above we may also eliminate leaf nodes by recognising that



(3.26)

where the ϵ_θ gate is some dephasing channel of the form,

$$\epsilon_\theta(\rho) = (1 - \sin^2\theta)I - (1 - \cos^2\theta)Z\rho Z^\dagger. \tag{3.27}$$

To show that Equation 3.26 is true, consider the action of the ZZ coupling on the first two qubits,

$$ZZ(\theta)\rho(ZZ(\theta))^\dagger = \frac{1}{2}\begin{bmatrix} e^{2i\theta} & 1 & 1 & e^{2i\theta} \\ 1 & e^{-2i\theta} & e^{-2i\theta} & 1 \\ 1 & e^{-2i\theta} & e^{-2i\theta} & 1 \\ e^{2i\theta} & 1 & 1 & e^{2i\theta} \end{bmatrix}. \tag{3.28}$$

Taking the partial trace over the second qubit gives

$$\rho_1 = \frac{1}{2}\begin{bmatrix} e^{2i\theta} + e^{-2i\theta} & 2 \\ 2 & e^{2i\theta} + e^{-2i\theta} \end{bmatrix} \tag{3.29}$$

$$= \begin{bmatrix} \cos(2\theta) & 1 \\ 1 & \cos(2\theta) \end{bmatrix} \tag{3.30}$$

$$= \begin{bmatrix} \cos^2\theta - \sin^2\theta & 1 \\ 1 & \cos^2\theta - \sin^2\theta \end{bmatrix} \tag{3.31}$$

which concurs with Equation 3.27. Hence Equation 3.26 is true.

This construction works well for low p instances of QAOA where we recognise that the leaf node must have a distance of $\geq p$ away from the leaf node in order to be removable. Note that the dephasing parameter θ must also be tuned in line with the other parameters of the QAOA ansatz.

3.4.3 Reduced Circuit Sizes for 3-regular graphs

To demonstrate the performance of these techniques, we generated 100 random 3-regular graphs using the NetworkX library and constructed the 100 qubit Max-Cut QAOA circuits for each. We then ran the above procedures to reduce the circuit size for each term in their respective Hamiltonians. We can see that for low degree graphs we are able to solve the Max-Cut problem using relatively only a small number of qubits. These sized circuits are easily achievable on current day NISQ era devices which are of the order of 65 qubits[14]. Combined with the high noise tolerance and success rate for low p of RQAOA it does appear to be possible to solve the Max-Cut problem using real quantum hardware.

p	1	2	3
Lightcone Removable	93.89	85.59	72.24
Leaf Node Elimination	3.90	8.20	12.69
Total Removable	96.99	93.79	84.93
Total Remaining	3.01	6.21	15.07

Table 3.1: Average number of qubits removable for terms in the Hamiltonian of 100 node 3-regular graphs

3.5 Summary

In this chapter, we demonstrated the noise tolerance of the QAOA algorithm and confirmed the existence of Noise-Induced Barren Plateau at relatively low depths. We then investigated an adapted form of QAOA known as recursive QAOA and determined the noise tolerance of this relatively new algorithm, a process that has not been undertaken before. Having demonstrated remarkable noise tolerance we then provided a pathway to execute QAOA circuits and in particular low p RQAOA on NISQ era hardware by exploiting the locality of the QAOA problem Hamiltonians to exclude as many gates and qubits as possible. The techniques presented resulted in a strong 97% decrease in the number of qubits required for $p = 1$ in 3-regular graphs. This demonstrates a path forward to run instances of RQAOA on NISQ era devices.

Chapter 4

Quantum State Preparation

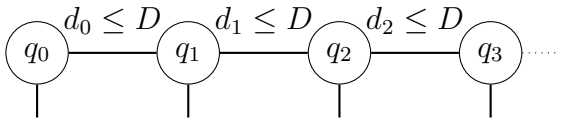
Loading quantum states is a key component of quantum algorithms such as Grover's Algorithm which we encountered in Chapter 2, as well as the HHL algorithm for solving linear systems of equations[44] and options pricing algorithms used in finance[5]. Despite being a key task, the best known techniques for determining circuits to load a state require exponential depth in general [45, 46]. Extending upon a procedure for Quantum State Tomography using MPS presented in [47] we will seek to develop algorithms which outperform current best known methods for quantum state generation of in particular low entanglement states.

We will present the baseline procedure as given in [47] in Section 4.1 and extend this procedure by applying stricter bounds on the size of the unitary operators generated in Section 4.2. Going further we develop a greedy algorithm in Section 4.3 which seeks to perform a similar procedure to that in Section 4.1 but with unitary operators of size no greater than 4×4. Finally, we present alternative circuit ansatzes in Section 4.4 which provide low depth variations on the procedures presented in Sections 4.1-4.3. We demonstrate the performance of each of these procedures in Section 4.6 for Gaussian, Random and W-States.

4.1 Outline of Baseline Procedure

This baseline procedure is based on the work of Cramer et al[47] and was originally proposed for the purposes of quantum state tomography. The procedure sequentially disentangles some arbitrary system $|\psi\rangle$ by constructing unitary operators of size $\kappa = \lceil \log(D) \rceil + 1$, where D is the maximum bond dimension in the MPS representation of the state.

Begin by considering the MPS form of some quantum system $|\psi\rangle$ formed as per Section 1.5.2.



(4.1)

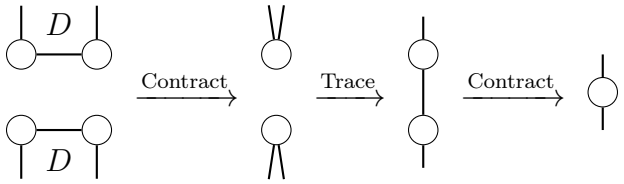
We may form the reduced density matrix over the first κ sites locally by first ensuring the MPS is in canonical form as per Section 1.5.4 and performing the operations in Equations 1.16-1.17 to form ρ_{red}. The eigendecomposition of ρ_{red} is,

$$\rho_{\text{red}} = P \Gamma P^{-1} \tag{4.2}$$

$$= \sum_{i=1}^{D} \lambda_i |\phi_i\rangle\langle\phi_i| . \tag{4.3}$$

where P is a matrix of eigenvectors, Γ is a diagonal matrix of descending eigenvalues, and λ_i and $|\phi_i\rangle$ are the ith eigenvalue and eigenvectors in descending order respectively. We then argue that the rank of this reduced density matrix must be bounded by $D = 2^{\kappa-1}$. To see this, consider the

following formulation of the partial trace of a system of two qudits



(4.4)

where one qudit is formed by the contraction of the first κ sites, with the remaining sites forming the other qudit. We note that the contractions in Equation 4.4 are ordinary matrix multiplications. We know that for ordinary matrices that,

$$\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B)). \tag{4.5}$$

Recalling that our internal bonds are in fact diagonal matrices with D non-zero elements we can then conclude that the matrices resulting from the contractions have a rank bounded by D. Hence our final reduced density matrix is also bounded by D. It's important to note here that our rank bound for the reduced density matrix is strictly dependent on the right internal bond of the κth site. This will be important in Section 4.2 where we use this fact to determine a construction with possibly smaller unitaries.

The implication of this result is that there must then exist some density matrix with the same eigenvalues as ρ_{red} but with one fewer qubit. Hence we may construct some operator to set one of the qubits to state $|0\rangle$ in ρ_{red}. To do so, take the eigenvectors $|\phi_i\rangle$ and extend them to form a basis of dimension 2^κ, using this basis to construct the operator $U = \sum_{i=1}^{2^\kappa} |i\rangle\langle\phi_i|$ where $|i\rangle$ is the ith computational basis vector. The action of this operator on the first κ sites will be,

$$U\rho_{red}U^\dagger = \sum_{i=1}^{2^\kappa} |i\rangle\langle\phi_i| \sum_{j=1}^{2^{\kappa-1}} \lambda_j |\phi_j\rangle\langle\phi_j| \sum_{k=1}^{2^\kappa} |\phi_k\rangle\langle k| \tag{4.6}$$

$$= \sum_{i=1}^{2^\kappa} \sum_{j=1}^{2^{\kappa-1}} \sum_{k=1}^{2^\kappa} \lambda_j \delta_{ijk} |i\rangle \langle k| \tag{4.7}$$

$$= \sum_{i=1}^{2^{\kappa-1}} \lambda_i |i\rangle\langle i| . \tag{4.8}$$

Recognising that all terms $i > 2^{\kappa-1}$ are zero we can conclude that the first qubit has been successfully set to zero. Having applied U onto the MPS one can then shrink the MPS by one qubit and repeat the procedure until all qubits have been set to $|0\rangle$. One can then apply the inverse of these unitary operations in reverse order from the $\left|0^{\otimes N}\right\rangle$ state to reconstruct $|\psi\rangle$. An example final circuit for constructing a four qubit arbitrary state in given in Figure 4.1. We provide the complete procedure which we name the Matrix Product State Circuit Generator (MPSCG) in pseudocode in Algorithm 1.

An important note to make is that there is a degree of freedom in constructing the complete basis which will form the unitary operators which zero out the qubits. We used the software package NumPy[42] to determine the eigensystem of ρ_{red} which incidentally provides a complete basis for Hermitian matrices which we used for all procedures in this thesis. An extension to this thesis could possibly include taking advantage of this degree of freedom to make more optimal choices for U.

4.1.1 Decomposing Unitary Gates

The MPSCG procedure generates unitary gates, which although may be applied to MPS and state vector simulators with ease, would not be executable on real quantum devices without

Algorithm 1 MPS Circuit Generator (MPSCG)

Input: MPS $|\psi\rangle = |q_0 q_1 \dots q_n\rangle$ in left canonical form
Output: List of unitaries $U^{\dagger}_{i,(i+1),\dots(i+\kappa)}$

function CONSTRUCT UNITARY($|\psi\rangle$, i, κ)
    $\rho_{red} \leftarrow \sum_{j=i}^{i+\kappa-1} \langle q_j|q_{j+1}\rangle\, |q_i\rangle\langle q_{i+\kappa}|$ ▷ where $|q_i\rangle$ contain their right bond tensor
    $\{|\phi_i\rangle\} \leftarrow \{|\phi_i\rangle\, |\rho_{red}\, |\phi_i\rangle = \lambda_i\, |\phi_i\rangle\}$ ▷ Assume these are ordered in descending order of λ_i
    $\{|\phi_i\rangle\} \leftarrow \{|\phi_i\rangle\} \cup \{|\phi_i\rangle \mid \langle\phi_i|\phi_j\rangle = \delta_{ij} \forall\, |\phi_j\rangle\}$ ▷ See Note on extending to a complete basis
    $U_{i,i+1,\dots(i+\kappa)} \leftarrow \sum_{i=1}^{2^\kappa} |i\rangle\langle\phi_i|$
    return U_i
end function
R $\leftarrow$ MAXIMUM BOND DIMENSION($|\psi\rangle$) ▷ May be determined by inspection of the $|q_i\rangle$
$\kappa \leftarrow \lceil \log_2(R) \rceil + 1$
for $|q_i\rangle \in |\psi\rangle$ **do**
    $U_{i,i+1,\dots(i+\kappa)} \leftarrow$ CONSTRUCT UNITARY($|\psi\rangle$, i, κ)
    $|\psi\rangle \leftarrow U_{i,i+1,\dots(i+\kappa)}\, |\psi\rangle$
end for

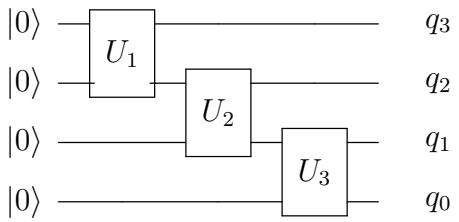


Figure 4.1: Circuit to construct the state $|\psi\rangle = |q_0 q_1 q_2 q_3\rangle$ with $\kappa = 2$

further decomposing them into single-qubit rotations and CX gates. For two-qubit unitaries this procedure is well established and can be done using no more than three CX gates and seven single qubit rotations for any $U \in U(4)$[48]. At three qubits, one may use a generalisation Cartan's KAK decomposition, known as the Khaneja-Glaser decomposition to create a circuit consisting of no more than 40 CX gates and 98 single-qubit rotations[49]. In general, it is found that arbitrary n-qubit unitary gates may be decomposed into elementary gates with $O(4^n)$ CX gates[24]. This is obviously impractical, hence we must either determine a more efficient unitary decomposition or limit the size of the unitaries which we generate. In this thesis, we will go down the latter route and devise algorithms with the aim of minimising the size of the unitaries generated as much as possible. We will defer to Qiskit's unitary decomposition based off [45] to decompose all the unitaries we generate.

4.2 Varying κ

The MPSCG procedure works well for states that may be represented with an MPS of low bounded bond dimension, such as the W-State which has a bond dimension of 2 everywhere. However, as soon as the bond dimension for some given state increases slightly anywhere along the MPS, the procedure becomes considerably worse as the size of the unitaries along the entire state increases. To try and address this problem, in particular for states with localised entanglement, we adapt the procedure to apply a tighter upper bound on the rank of the reduced density matrix

of some subsystem of the MPS. Recognising that from the prior section the rank of the reduced density matrix is bounded not by the overall maximum bond dimension but by the internal bond to the right of the $i+\kappa$ site, we can construct an algorithm to set our κ dynamically. In Algorithm 2 we outline the steps needed to execute this variation on the MPSCG procedure.

Algorithm 2 Variable κ Quantum State Generator (VMPSCG)

Input: MPS $|\psi\rangle = |q_0 q_1 \dots q_n\rangle$
Output: List of unitaries $U^\dagger_{i,i+1,\dots(i+\kappa)}$
$i \leftarrow 0$
while $\langle 0|q_j\rangle \neq 0 \forall\, |q_j\rangle \in |\psi\rangle$ **do**
  if $\langle 0|q_i\rangle \neq 0$ **then**
    $\kappa \leftarrow 1$
    while SCHMIDT RANK($|q_{i+\kappa-1}\rangle$, $|q_{i+\kappa}\rangle$) $< \lceil \log_2(\kappa) \rceil + 1$ **do**
      $\kappa \leftarrow \kappa + 1$
    end while
    $U_{i,i+1,\dots(i+\kappa)} \leftarrow$ CONSTRUCT UNITARY($|\psi\rangle$, i, κ) ▷ As per Algorithm 1
    $|\psi\rangle \leftarrow U_{i,i+1,\dots(i+\kappa)} |\psi\rangle$
  end if
  $i \leftarrow i + 1$
end while

Note that this algorithm, like MPSCG, is deterministic. You are guaranteed to recreate any state $|\psi\rangle$ perfectly. This variation on the algorithm will be useful for states with highly localised entanglement, however it will still generate some large unitaries which is not preferable. We will name this variation the Variable (κ) MPSCG (VMPSCG). We present results for this procedure for Gaussian and Random states in Section 4.6.2.

4.3 Fixing κ to 2

The VMPSCG procedure presents an improvement over the baseline MPSCG procedure, however, the generation of unitaries larger than 4×4 is still not preferable as they require considerably more elementary gates than two-qubit unitary operators. To remedy this we will propose a greedy algorithm that may require numerous passes over the MPS but will only generate two-qubit unitaries.

For this procedure, consider the reduced density matrix over the first $\kappa = 2$ sites,

$$\rho_{red} = \sum_{i=1}^{2^\kappa} \lambda_i |\phi_i\rangle\langle\phi_i| . \tag{4.9}$$

Unlike in Section 4.1, ρ_{red} is full rank matrix. Hence applying the same operator $U = \sum_i^{2^\kappa} |i\rangle\langle\phi_i|$ will result in the following reduced density matrix,

$$U\rho_{red}U^\dagger = \sum_{i=1}^{2^\kappa}\sum_{j=1}^{2^\kappa}\sum_{k=1}^{2^\kappa} |i\rangle\langle\phi_i| \left(\lambda_j |\phi_j\rangle\langle\phi_j|\right) |\phi_k\rangle\langle k| \tag{4.10}$$

$$= \sum_{i=1}^{2^\kappa}\sum_{j=1}^{2^\kappa}\sum_{k=1}^{2^\kappa} \delta_{ijk}\lambda_j |i\rangle\langle k| \tag{4.11}$$

$$= \sum_{i=1}^{2^\kappa} \lambda_i |i\rangle\langle i| \tag{4.12}$$

$$= \lambda_1 |00\rangle\langle 00| + \lambda_2 |01\rangle\langle 01| + \lambda_3 |10\rangle\langle 10| + \lambda_4 |11\rangle\langle 11| . \tag{4.13}$$

Unfortunately we have been unable to successfully set the first qubit to the $|0\rangle$ with this operator, but we can show that this is the closest one can get to the $|0\rangle$ state on the first qubit by application of a two-qubit operator.

It has been shown in [50] that for a Hermitian matrix A with diagonal entries $h_1, h_2 \dots h_n$ and eigenvalues $\lambda_1, \lambda_2 \dots \lambda_n$ that

$$(h_1, h_2, \dots, h_n) \prec (\lambda_1, \lambda_2, \dots, \lambda_n) \tag{4.14}$$

where $\prec$ denotes majorisation and is defined as both of the following conditions being true,

$$\sum_i^k x_i \leq \sum_i^k y_i \tag{4.15}$$

$$\sum_i^n x_i = \sum_i^n y_i, \tag{4.16}$$

for some $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_n)$, and some $k = 1, 2 \dots n$. Now, noting that the probability of measuring $|0\rangle$ on the first qubit is

$$\langle 00|\rho_{red}|00\rangle + \langle 01|\rho_{red}|01\rangle = (\rho_{red})_{00} + (\rho_{red})_{11} = h_1 + h_2 \tag{4.17}$$

prior to applying the operator U. After the application of U this becomes,

$$\langle 00|U\rho_{red}U^\dagger|00\rangle + \langle 01|U\rho_{red}U^\dagger|01\rangle = \lambda_1 + \lambda_2. \tag{4.18}$$

Finally, as we can see from Equation 4.14,

$$h_1 + h_2 \leq \lambda_1 + \lambda_2 \tag{4.19}$$

Hence we may conclude that application of the operator U will get you as close to setting the first qubit to $|0\rangle$ as possible. Given this result we construct a greedy algorithm, as per Algorithm 3, which sets every qubit as close as possible to $|0\rangle$ through the application of U, repeating until some threshold minimum fidelity in met. We will name this procedure Fixed (κ) MPSCG (FMPSCG).

Algorithm 3 Fixed κ Quantum State Generator (FMPSCG)

Input: MPS $|\psi\rangle = |q_0 q_1 \dots q_n\rangle$, κ, Threshold fidelity F
Output: List of unitary, layer tuples $(U^\dagger_{i,j\dots k}, l)$

```
l ← 0
while |⟨ψ|U_i^†|0^⊗N⟩|^2 < F do
    for |q_i⟩ ∈ |ψ⟩ do
        (U_{i,(i+1),...(i+κ)}, l) ← Construct Unitary(|ψ⟩, i, κ)        ▷ As per Algorithm 1
        |ψ⟩ ← U_{i,(i+1),...(i+κ)} |ψ⟩
    end for
    l ← l + 1
end while
```

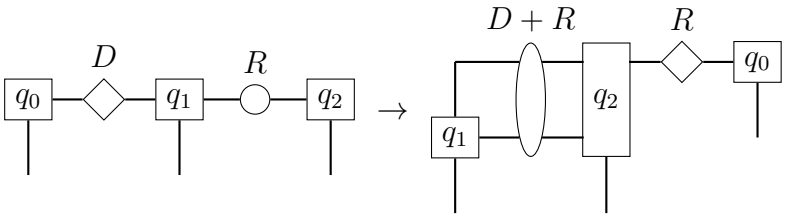
It is hoped that this procedure reduces the overall depth of the state preparation circuit as although the procedure produces more unitaries than MPSCG and VMPSCG which produce strictly $O(n)$ unitaries, the cost of generating unitaries of size greater than 4×4 is prohibitive which this procedure avoids. We demonstrate the performance of this procedure in terms of state fidelity, circuit depth and CX count for Gaussian and Random states in Section 4.6.3.

Although not investigated in this thesis, this procedure may be hybridised with VMPSCG, i.e. vary κ but with some upper bound. This will in general harm the overall fidelity of the state generated and will hence still necessitate multiple passes over the MPS, however, it may result in overall fewer layers than pure FMPSCG.

4.4 A low depth approach

It is natural at this point to consider the possibility of parallelising these procedures to generate even lower depth circuits, especially in the case of states which are well served by the baseline MPSCG procedure such as W-States. One such optimisation we introduce is that given the ability to start at either end of the MPS to undertake any of the procedures in Sections 4.1-4.3, it is possible then to do both in parallel and meet in the middle. This halves the depth of the circuits generated. We will name this type of circuit "V-Shaped" MPSCG (VSMPSCG). However, we can go even further and generate circuits with $O(\log(n))$ depth. To do so we zero out every κ qubit simultaneously and then work in jumping over any qubits that have already been zeroed out. We will name this type of circuit "Low Depth" MPSCG (LDMPSCG). Example circuits for both these procedures can be found in Figure 4.2.

Fundamentally these circuits work because of the result in Section 4.3 which applies equally to any reduced density matrix, not just those formed by adjacent qubits. That said it is important to consider the rank of the reduced density matrix formed by these non-edge qubits as it may inform the fidelity lost by using these circuits. The simplest way to approach this is to consider a 3 qudit system where the central qudit is the subsystem which we wish to determine the reduced density matrix. Then consider the remapping of the left qudit to the right of the system such that the qudit of interest is then located at the edge of the MPS,



$$\bigcirc = \begin{bmatrix} \lambda_0 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \rightarrow \bigcirc\!\!\!\!\bigcirc = \begin{bmatrix} \mathbf{I}_D & & & \\ & \lambda_0 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}, \square = \begin{bmatrix} \mathbf{I}_D & 0 \\ 0 & |q_0\rangle \end{bmatrix}$$

where we have increased the size of the q_1q_2 bond tensor and the q_2 qubit tensor to "pass through" the q_0q_1 through q_2. Hence all sites on the right partition have had their internal bond dimension increase by D. This is of course an upper bound for the rank of the matrix. Most implementations of site swapping, including the ones used by the software packages used in this thesis implement the swapping of sites through the contraction of the region of interest followed by a decomposition with the indices of the swapped sites moved around. This allows for a possibly smaller internal bond dimension to arise after swapping. As we will see in the Results section, this is often what happens in practice for slightly entangled states.

Note that, as we are seeking only 2-qubit unitaries we will fix $\kappa = 2$ and use an FMPSCG like algorithm with some threshold fidelity. Although in theory one is free to apply larger κ values based on the bond dimension of the MPS. We present the results for both these alternative circuit ansatzes in Section 4.6.5 for Gaussian, Random and W states.

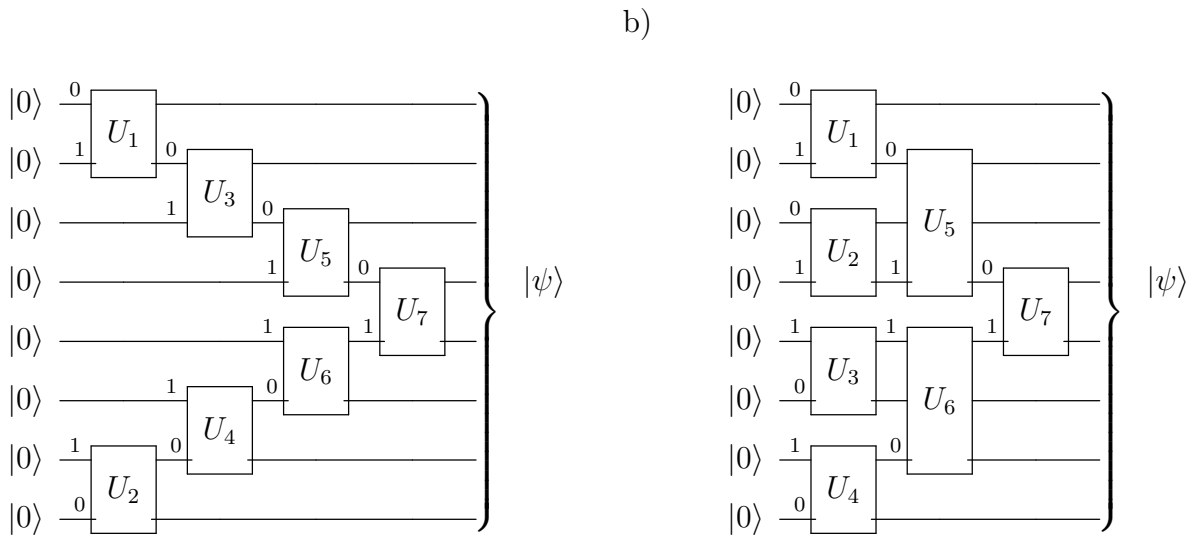


Figure 4.2: Example 8 qubit circuits for the 2 low depths procedures in Section 4.4. (a) VSMPSCG procedure with $\kappa = 2$ (b) LDMPSCG procedure with $\kappa = 2$. Note that the numbers next to the unitary gates denotes the qubit ordering of the operator

4.5 Tailoring Circuits to NISQ era devices

The final problem we will consider in this thesis is the tailoring of these state preparation circuits to given hardware layouts. Although typical circuit diagrams make it appear as though any qubit may interact with any other, in real quantum devices this is seldom the case. In general, each qubit may only be connected to a few others in its local vicinity and swap gates are required if you wish to interact one distant qubit with another. SWAP operations however are costly, requiring 3 CXS each, hence avoiding these SWAP operations would be ideal. Hence we must map our states onto these hardware layouts and determine some procedure to sequentially zero out each of the qubits.

As we have been only considering 1-dimensional tensor networks in this thesis, this connectivity will not correspond to the connectivity of a real quantum device in general. However, the result from Section 4.3 permits us to construct a unitary from any reduced density matrix that we wish. That said, it is still prudent to pay attention to the bond dimensions that will result from these constructions. Extending from Section 4.4 it is clear that moving qubits results in a linear increase in the bond dimensions to the right (or left) of the system of interest. Hence we can conclude that at worst the bond dimension for a reduced density matrix over q_i sites will be $\sum_{i \in q_i}(L_i + R_i)$ where L_i is the bond dimension to the left of qubit i, and R_i is the bond dimension to the right of qubit i.[1]

Given this, we have permitted ourselves a significant degree of freedom in determining the structure of the circuit. As it is typical for quantum hardware layouts to contain leaf nodes, this appeared to be a natural starting point for our procedure. For simplicity, we restrict ourselves $\kappa = 2$. Our proposed algorithm will begin at some given leaf node and attempt to zero it out, eliminating it from the graph, then continue by picking another leaf node from the reduced graph, working in until all nodes have been attempted to be zeroed out. In the event that the graph being processed does not have any leaf nodes, one edge will be cut at random. As per the procedure in Section 4.3, it may be necessary to do several passes over the original graph to successfully meet

[1]This is a loose upper bound. In general, only the leftmost qubit may have a contribution from both both the left and right bipartition

some threshold fidelity. An outline of the procedure (which we will call Connectivity Constrained MPSCG (CCMPSCG)) is provided in Algorithm 4. We also provide an example circuit for the hardware layout of IBM's 7 qubit Lagos device in Figure 4.3.

In general, the circuit depth generated by this algorithm will be O(n), which is a regression from the low depth procedures discussed above. However this does not account for the SWAP's that would be required for the other procedures. Ultimately, this procedure serves as a proof of concept for tailored quantum state preparation circuits which if successful can give rise to other procedures which utilise the properties of the graphs to generate lower depth circuits.

Algorithm 4 Connectivity Constrained Quantum State Generator (CCMPSCG)

Input: MPS $|\psi\rangle = |q_0 q_1 \dots q_n\rangle$, $G = (V, E)$, Threshold fidelity F
Output: List of unitary, layer tuples $(U_{i,j}^{\dagger}, l)$

$l \leftarrow 0$
while $\left|\langle\psi|U_i^{\dagger}|0^{\otimes N}\rangle\right|^2 < F$ **do**
  $G = (V, E) \leftarrow$ Copy(G)
  while Is Empty(G) = false **do**
    for $i \in V$ **do**
      if Degree$(G, i) = 1$ **then**
        Continue ▷ i.e. If no leaf nodes, then last node is selected
      end if
    end for
    $i, j \leftarrow$ Edge(G, i).first
    $|\psi^*\rangle \leftarrow$ Remap$(|\psi\rangle, i, j)$ ▷ Remap such that i, j are next to each other
    $(U_{i,j}, l) \leftarrow$ Construct Unitary$(|\psi^*\rangle, i, 2)$ ▷ As per Algorithm 1
    $|\psi\rangle \leftarrow U_{i,j}|\psi\rangle$
    Remove Node(G, i)
  end while
  $l \leftarrow l + 1$
end while

4.6 Results

To demonstrate the efficacy of the various procedures outlined in the previous sections we will generate circuits for a number of different states commonly encountered in Quantum Computing and compare circuit depths, gate counts and state fidelities between each of the procedures in addition to the state initialisation procedure found with Qiskit[51]. We used Quimb to create the MPS of the states and to compute the reduced density matrices. NumPy was used to determine the eigensystems of the reduced density matrices. Unitary decomposition and the execution of final circuits was completed using Qiskit. The states generated were;

- Gaussian States: These are encountered in Options Pricing Algorithms[5] where one requires the underlying distribution of the asset price at maturity as an input. We expect that these states are slightly entangled but should still be constructible using most procedures at a relatively low depth. These states are of the form

$$|\psi\rangle(\mu, \sigma) = \sum_{i=0}^{2^N} \frac{e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}}{\sigma\sqrt{2\pi}} |i\rangle \tag{4.20}$$

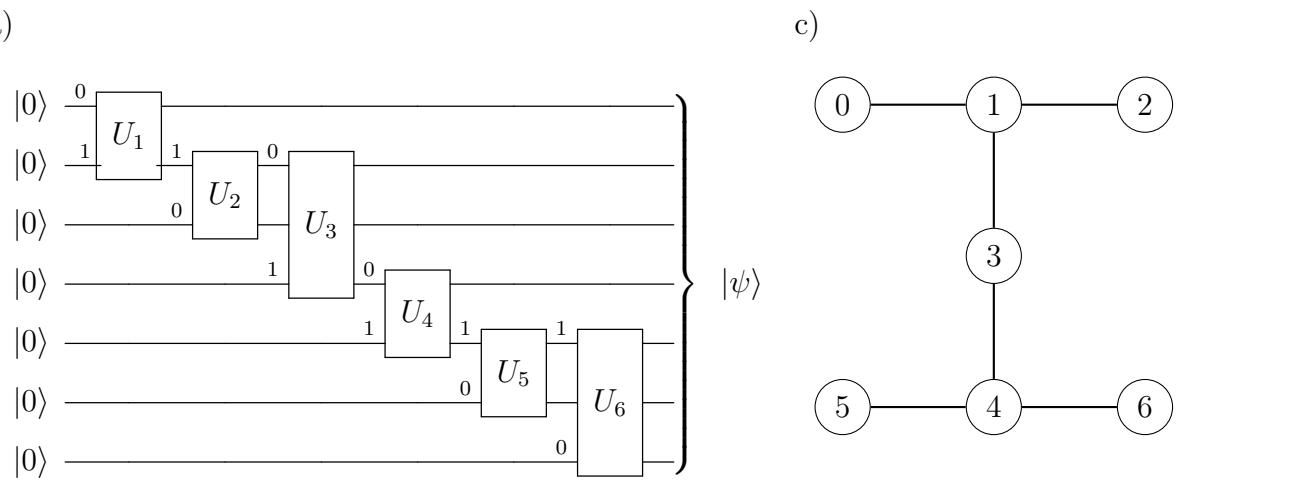


Figure 4.3: (a) A circuit generating state $|\psi\rangle$ which respects the qubit connectivity of ibm_lagos. (b) The qubit layout and connectivity of IBM's 7 qubit device ibm_lagos

with μ being the average and σ being the standard distribution as is typical. For all our simulations we set $\mu = 2^N/2$ and $\sigma = 2^N/3$, this results in a rather wide distribution with the expectation that most states should have a significant probability of being measured. An example of this state is provided in Figure 4.4a

- W-States: This state represents a distinct kind of multipartite entanglement. It is of the form
$$|\psi\rangle = \frac{1}{\sqrt{N}}(|10\ldots0\rangle + |010\ldots0\rangle + \cdots + |0\ldots1\rangle) \tag{4.21}$$
The important thing to note about the W-state is that it has a bipartite entropy of 2 between any two subsystems of the state. Hence this state should be constructible with all procedures easily. An example of this state is provided in Figure 4.4b

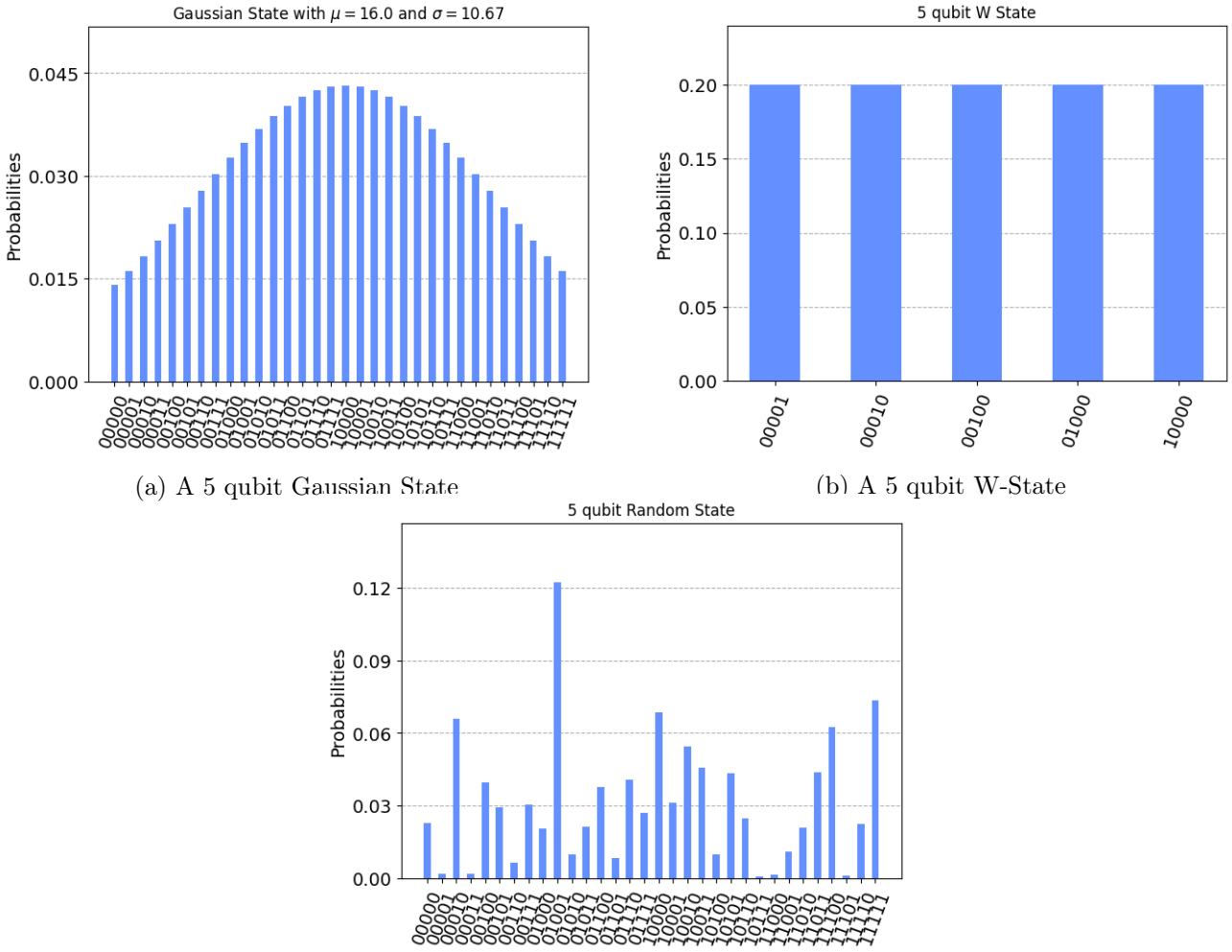
- Random States: These states are generated randomly using Qiskit. We anticipate that these states are highly entangled and will not be easily constructible. An example of this state is provided in Figure 4.4c

4.6.1 MPSCG

Firstly we analyse the performance of the MPSCG procedure outlined in Section 4.1 for the three states specified above. We expect the fidelity of the circuits generated to be 100% as the procedure is deterministic. This is verified Figure 4.5a.

Next, we analyse the size of the unitaries for the three states. We see in Figure 4.5b that as expected the random states required the largest unitaries, with W States requiring only requiring 4×4 sized unitaries regardless of the overall size of the system. As discussed in Section 4.1.1 the decomposition of arbitrary gates of size $N \times N$ requires an exponential number of elementary gates, Figure 4.5c illustrates this correspondence with the increase in κ for random states corresponding directly to the exponential growth in the circuit depth.

The primary result of interest, circuit depth compared to Qiskit can be seen in Figure 4.5d. We see that W-States may be executed with an exponentially smaller depth compared to Qiskit. This is not surprising as W-States, having an entanglement entropy of 2 everywhere, are known to be a good fit for this type of procedure as seen in [47]. Gaussian and Random States appear to scale



(a) A 5 qubit Gaussian State

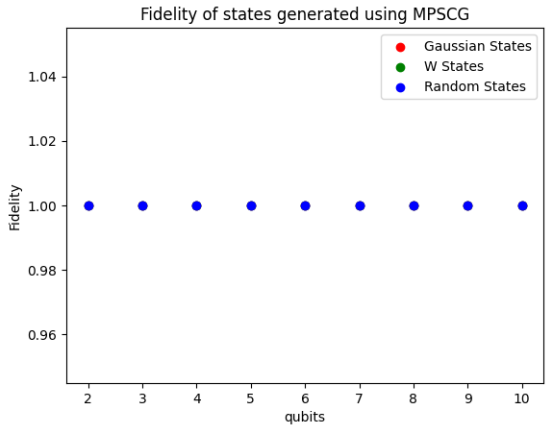
(b) A 5 qubit W-State

(c) A 5 qubit Random State

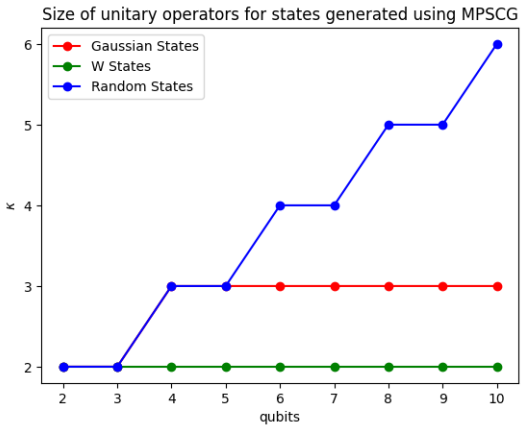
Figure 4.4: Examples of Gaussian, W and Random States that we will be determining circuits for

similarly to Qiskit, but with a larger constant factor meaning that overall circuit depth and gate counts are larger than Qiskit. This is not an entirely unexpected result, with our VMPSCG and FMPSCG procedures designed to address this. Results for both of these procedures for Gaussian and Random states are found in Sections 4.6.2 and 4.6.3 respectively.

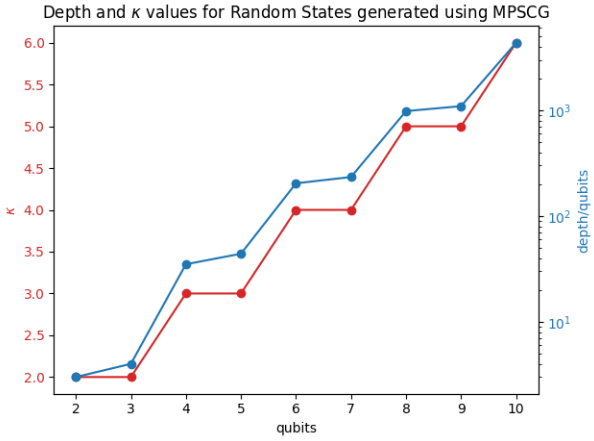
Finally, we analyse the correlation between the CX and $U(\theta, \phi, \lambda)$ gate counts and the circuit depth for Gaussian states generated using both Qiskit and MPSCG as per Figure 4.6.4. We recognise that in the case of MPSCG the depth of the circuit is closely correlated with both CX and single-qubit rotation gate counts, whereas with Qiskit it appears that the depth is more closely correlated to the CX gate counts. The natural explanation here is likely to be related to the "block-wise" locality of the MPSCG procedure which results in at most a distance of κ between the target and the control of any given gate. This would permit some greater degree of parallelisation of single-qubit operations far from the CX gate. This would not be the case for Qiskit's procedure which may in general result in CX gates spanning the entire circuit hence preventing any compression of single-qubit gates and hence resulting in the CX gate being a bigger driver of overall circuit depth ultimately resulting in correlations which we observe.



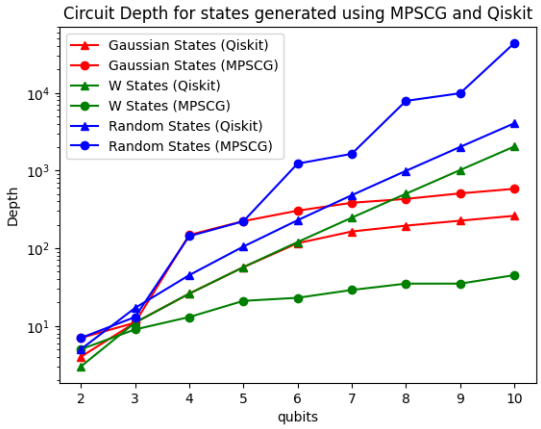
(a) Fidelity of states generated using MPSCG



(b) κ values for states generated using MPSCG



(c) κ, circuit depth correspondence for states generated using MPSCG



(d) Circuit Depths for MPSCG and Qiskit

Figure 4.5: Baseline Statistics for the MPSCG procedure on Gaussian, W and Random States

4.6.2 VMPSCG

In our first attempt to improve upon the state preparation construction for Gaussian states, and to a lesser extend random states, we now consider the VMPSCG procedure. Just like the MPSCG procedure from the previous section, this procedure is also deterministic hence all circuits generated have a fidelity of 100%. To quantify the improvement in the size of the unitary matrices we present the average κ values for the two states compared to MPSCG in Figure 4.7a. We see an improvement to the average κ in VMPSCG compared to MPSCG for both states. Considering the overall circuit depths as per Figure 4.7b we note that this does indeed result in a decrease compared to the standard MPSCG procedure for almost all system sizes. However, we recognise that this procedure is still worse than Qiskit for both Gaussian and random states. This is an expected result for random states, however, we had hoped that this procedure would've resulted in stronger performance for Gaussian states which have only modest entanglement. Given this result, we conclude that having any unitary operator of more than 2 qubits is too detrimental to the circuit depth without any tailored decomposition strategies. The path forward to achieve results that outperform Qiskit may then involve fixing $\kappa = 2$ as per the upcoming section.

We also include the CX gate counts for this procedure as well as the standard procedure and Qiskit in Figure 4.7d. As expected, we see a pretty close correspondence to the circuit depth.

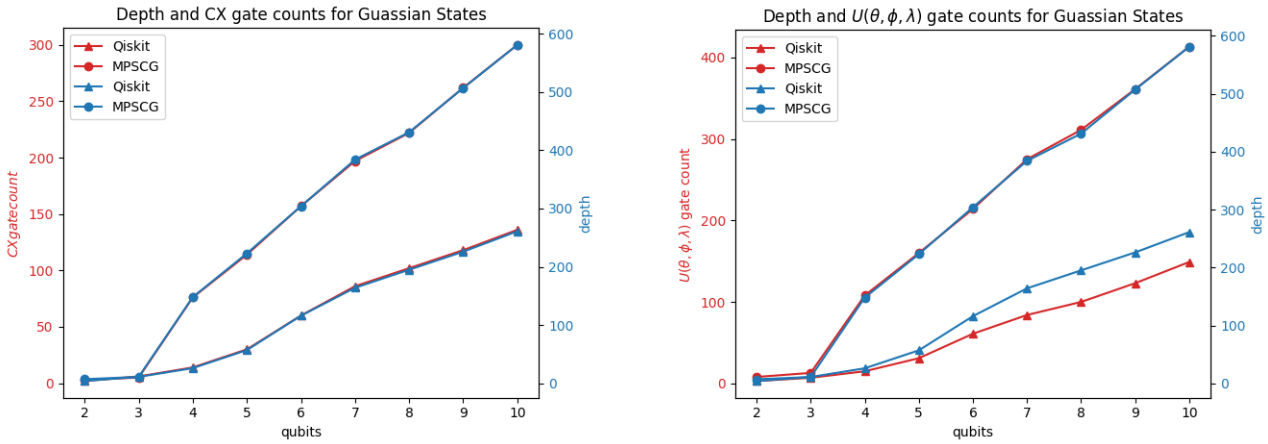


Figure 4.6: Number of CX (left) and $U(\theta, \phi, \lambda)$ (right) gates and circuit depths for MPSCG and Qiskit procedures

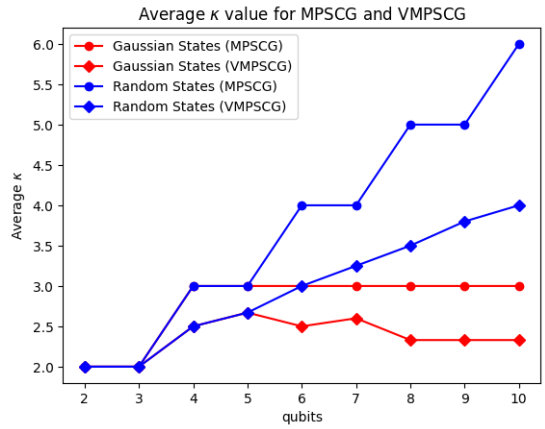
Furthermore, we see that the κ, circuit depth correlation we noted in the previous section holds true with this variable procedure as well, as can be seen in Figure 4.7c. We note however that the correlation appears to be closer than it was for MPSCG which can be likely attributed to the tighter bounds on the size of the unitaries which this procedure imposes.

Although this procedure does not outperform Qiskit we still recognise that it outperforms the standard MPSCG, and at worst will be equal to it. Hence, we suggest that this procedure supplants MPSCG for those requiring a deterministic state preparation algorithm.

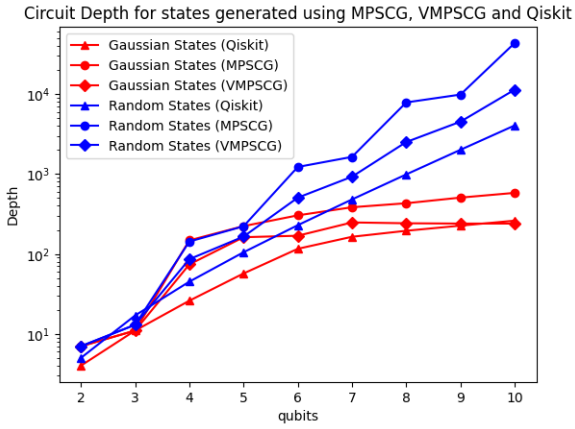
4.6.3 FMPSCG

Having exhausted our deterministic procedures, we now attempt our greedy FMPSCG algorithm on Gaussian and random states. We apply the algorithm for Gaussian and Random States with a threshold fidelity of 99% and present the final fidelities for both states in Figure 4.8a. We see that the fidelity of the Gaussian state is always close to 100% whereas for Random States we note that the fidelity approaches the minimum threshold of 99%. Looking at the number of layers needed to reach this threshold fidelity in Figure 4.8b we see that for Gaussian states the number of layers remains fixed at one throughout, whereas for random states we see an exponential increase in the number of layers as expected.

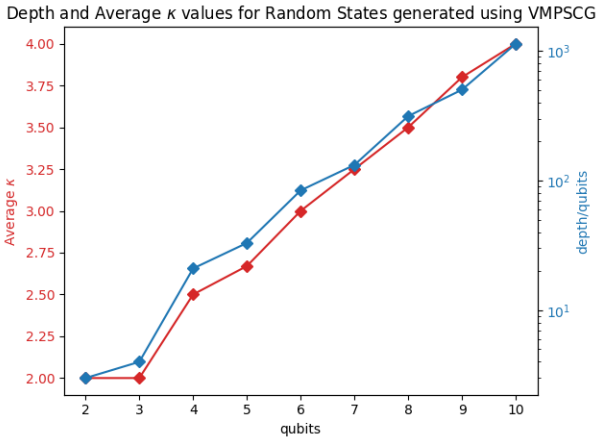
We present the overall circuit depths compared to Qiskit in Figure 4.8c. As hinted at from the number of layers, we see that Gaussian states have a significantly reduced depth compared to the previously discussed procedures including Qiskit. We also see that for Random states, despite the exponential growth in the number of layers and circuits depths persisting, FMPSCG is likewise outperforming all previously discussed procedures including Qiskit. Taking a look at the CX gate counts in Figure 4.8d however we see that Qiskit requires fewer CX gates overall for random states. This suggests that the FMPSCG generated circuits are far more dense, which we can justify theoretically as increasing the number of layers in an FMPSCG circuit increases the depth of the circuit by κ. Hence a doubling of the number of unitary operations only results in an increase of two in the circuit depth (in terms of the U_i operators). That said, as we note from Section 2.4.1 the primary driver of noise in NISQ era devices is the limited coherence time of the device more so than the gate infidelities. Hence an improvement in the circuit depth at the cost of a higher gate count may still be beneficial. However, a more robust analysis would need to be undertaken here to confirm this.



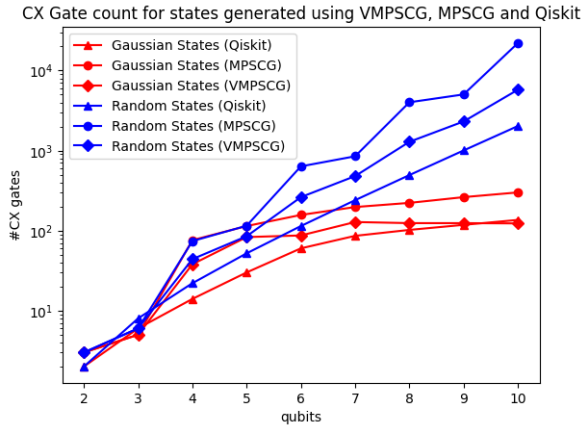
(a) Average κ values for MPSCG and VMPSCG



(b) Circuit Depths for MPSCG, VMPSCG and Qiskit



(c) κ, circuit depth correspondence for states generated using VMPSCG



(d) CX Gate counts for MPSCG, VMPSCG and Qiskit

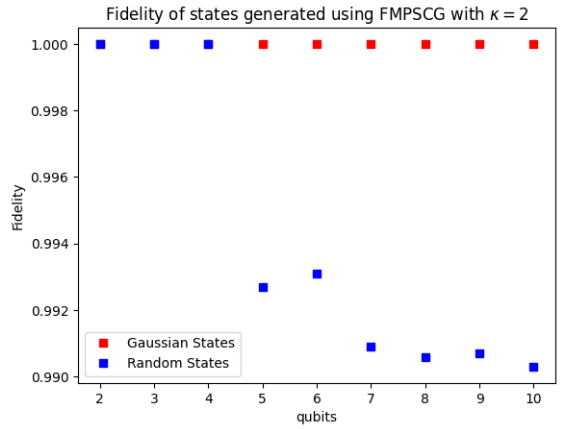
Figure 4.7: Baseline Statistics for the VMPSCG procedure on Gaussian and Random States

Overall this procedure has resulted in a significant improvement over Qiskit for Gaussian states in terms of both circuit depth and CX gate counts, demonstrating the benefit of this algorithm for slightly entangled states.

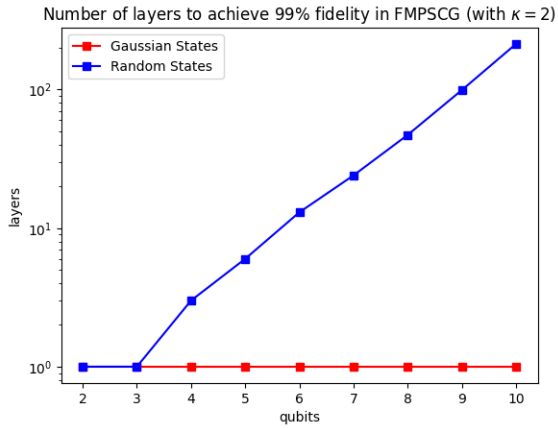
4.6.4 LDMPSCG & VSMPSCG

Next, we discuss the two low depth procedures formulated in Section 4.4. We will analyse the performance of all three states we have introduced in this chapter as they all could benefit from these procedures. As in the previous section the fidelity is set to a minimum of 99%. Firstly, we observe in Figure 4.9a that the fidelity of the states produced is comparable to the best procedures encountered thus far, with no notable decline across any of the states and only minimal difference between the two low depth procedures.

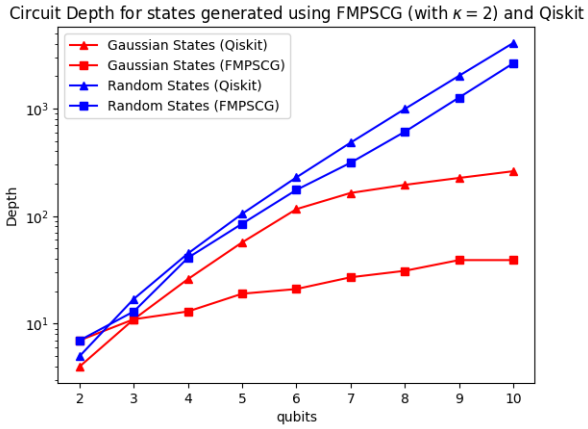
Analysing the circuit depth of Gaussian and W States for the low depth procedures as per Figure 4.9c, we see that both LDMPSCG and VSMPSCG outperform the best procedures encountered thus far, with LDMPSCG performing the best overall. This means that for Gaussian and W States we have constructed a circuit that scales as $O(\log(n))$ which is possibly better than any known construction for these states. Analysing the CX gate count for these circuits as per Figure 4.9d we see that all procedures regardless of depth or the overall number of layers appear to



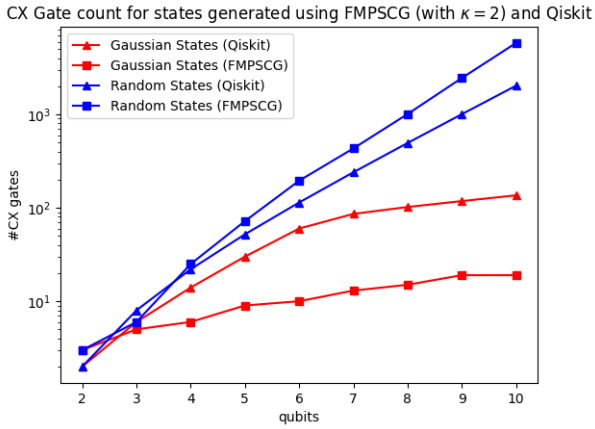
(a) Fidelity of Gaussian and Random States when run with a 99% threshold



(b) Number of layers required to achieve 99% fidelity



(c) Circuit Depths for FMPSCG procedure compared to Qiskit



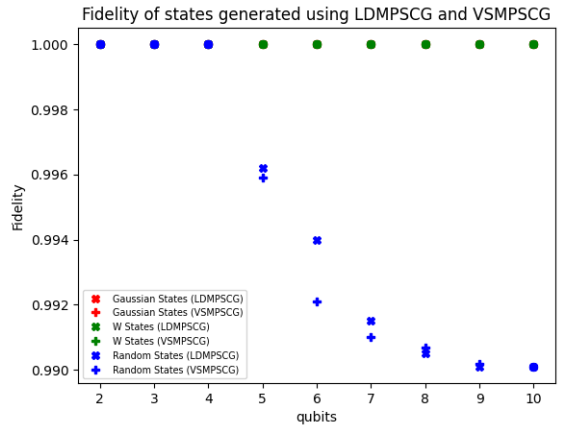
(d) CX Gate counts for FMPSCG compared to Qiskit

Figure 4.8: Baseline Statistics for the FMPSCG procedure ($\kappa = 2$) on Gaussian and Random States

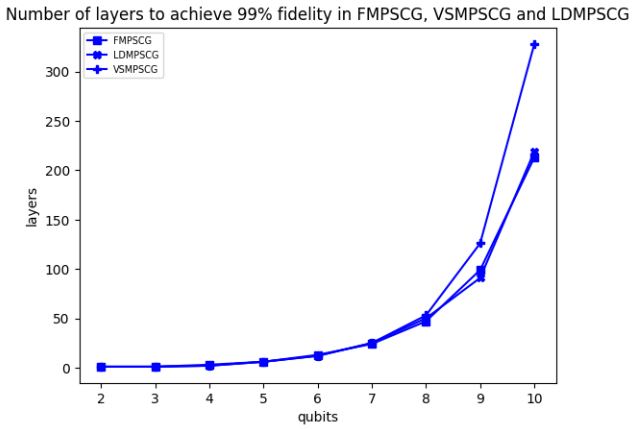
have a similar gate count. This is particularly interesting for random states using VSMPSCG as the number of layers appears to be significantly greater than other procedures. The likely reason for this are the optimisation strategies in Qiskit's transpiler which was used to decompose the U_i operators.

For Random states, we plot the number of layers for the two low depth procedures compared to FMPSCG in Figure 4.9b. We observe that the number of layers for VSMPSCG procedure grows faster than FMPSCG and LDMPSCG which have a similar number of layers. This was not an expected result. We however believe the reason for this result is the symmetry of the circuit ansatz along the central bipartition. Random states obviously do not possess this symmetry. Gaussian (with $\mu = 2^N/2$) and W States however do have this symmetry and hence are seemingly unaffected by the symmetry implied in this ansatz. Analysing the depths of the circuits we see that for random states the LDMPSCG procedure has a significantly greater depth than FMPSCG and VSMPSCG in spite of the relatively fewer layers compared to VSMPSCG in particular. This is a result of the fact that for FMPSCG and VSMPSCG the depth increases as $O(\kappa l + n)$ whereas for LDMPSCG the depth increases as $O(nl)$, where l is the number of layers. Hence, despite the number of layers being lower the increase in the number of qubits continues to drive the depth higher.

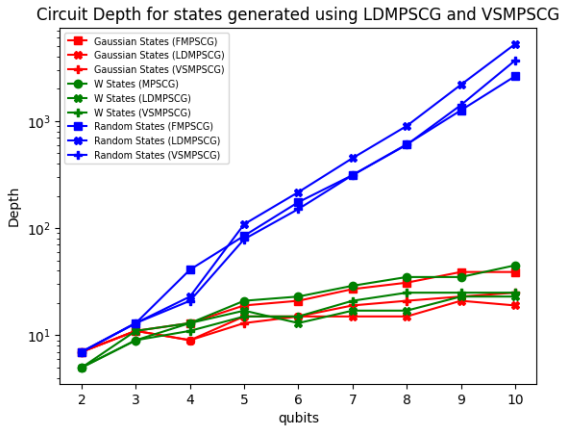
Analysing the ratio of CX and $U(\theta, \phi, \lambda)$ gates to circuit depth for the low depth, fixed κ



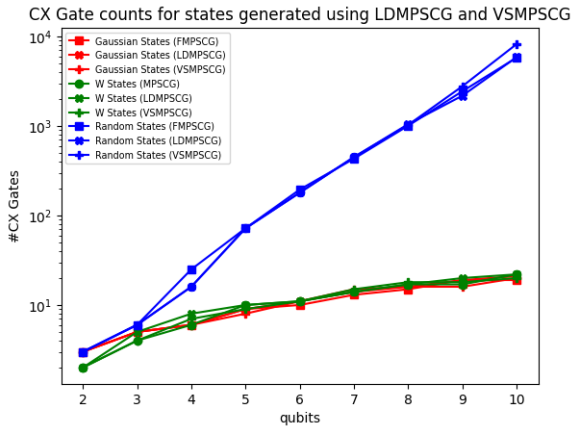
(a) Fidelity of Gaussian, Random and W States when run with a 99% threshold



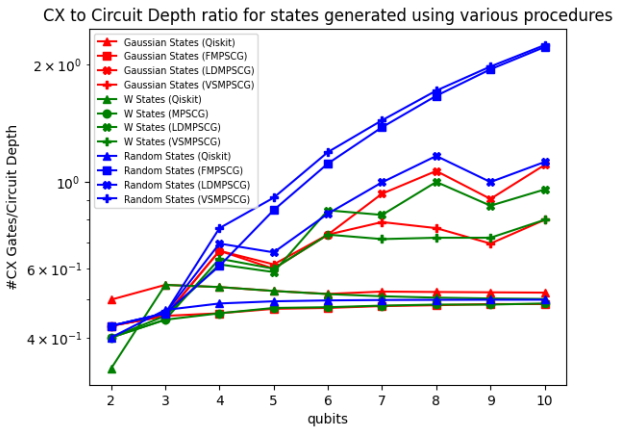
(b) Number of layers required to achieve 99% fidelity for random states



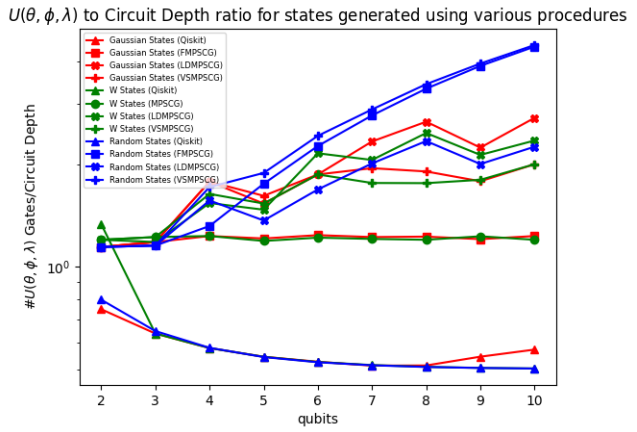
(c) Circuit Depths for proposed low depth procedures



(d) CX Gate counts for proposed low depth procedures



(e) CX to Circuit Depth ratio



(f) $U(\theta, \phi, \lambda)$ to Circuit Depth ratio

Figure 4.9: Baseline Statistics for the LDMPSCG and VSMPSCG procedures with $\kappa = 2$ on Gaussian, Random and W States

and Qiskit procedures in Figures 4.9e and 4.9f allows us to gain some insight as to the density of the circuits generated. Having low gate to depth ratios suggests that the circuit is of a lower density with many operations not occurring in parallel. We see that the low depth procedures are

very dense compared to Qiskit and low layer FMPSCG. This reaffirms the result in Figure where we concluded that Qiskit's circuits are being bounded by long-distance CX gates which prevent the condensing, and hence parallelisation, of the circuit. We also note that in spite of the low parallelisation capability of LDMPSCG for a large l, its circuit density still appears comparable to $l = 1$ VSMPSCG which is considerably greater than Qiskit and $l = 1$ FMPSCG.

Overall we can conclude that these low depth procedures work particularly well for circuits with low bipartite entropy which would otherwise be reproducible with FMPSCG with $l = 1$. In this case, the circuits generated are of depth $O(\log(n))$ which is significantly fewer than those generated by Qiskit or standard MPSCG. We recognise, however, that these circuit ansatz may not be ideal for states which require $l \gg 1$, and VSMPSCG in particular is not suited for states which are asymmetric in this instance.

4.6.5 Connectivity Constrained Procedure

Lastly, we consider the connectivity constrained procedure for Gaussian and W-States. We omit Random States from the discussion here as we recognise that in spite of improvements in the circuit depths with the procedures discussed thus far, the overall depth required to reproduce random states is far too high to be practicable on any NISQ era device. We generate circuits for two quantum devices, the 7 qubit ibm_lagos and the 16 qubit ibmq_guadalupe. The hardware layout for both of these can be found in Figure 4.11. As with FMPSCG we ran the algorithm with a threshold fidelity of 99% and we present the results for the 2 devices compared to the best procedure encountered thus far (LDMPSCG) in Table 4.1. The main question that we seek to answer is whether this connectivity constrained circuit generation algorithm outperforms the other procedures after being transpiled for a given hardware layout. To achieve this, we transpiled the LDMPSCG and CCMPSCG circuits using Qiskit with the coupling map for the two devices set. As we see, for the 7-qubit device we only see a negligible improvement if any in the depth of the circuit, with the primary advantage of the CCMPSCG procedure being the CX gate count. For the 16 qubit device, however, we see a more substantial improvement in the depth of the circuit. Note that the transpilation procedure of Qiskit did give inconsistent results from one run to the next when provided with a coupling map. Hence we have taken the best result produced but do note that the overall variation was small, being at most 2-5 gates for any given circuit.

In Figure 4.10 we provide histograms of the Gaussian and W States produced with the CCMPSCG procedure as run on an MPS simulator. This illustrates the overall high fidelity of the states that are produced. Overall though, in spite of some improvement in the depth of the circuit, the benefits do appear to be rather minimal for the circuit ansatz that we have constructed. That said we believe that more sophisticated procedures should be able to produce circuits with more significantly reduced depth. As we saw in the previous section, however, one must be careful to consider the symmetry of the ansatz, and in particular for this connectivity constrained procedure, that any symmetry in the qubit layout does not result in a similarly symmetrised circuit ansatz.

4.7 Summary

In summary, we have developed novel procedures to determine circuits for arbitrary quantum states extending upon the existing framework of MPS assisted quantum state tomography. In addition to successfully producing $O(n)$ depth circuits for W states, a known result. We extended the procedure to produce circuits of $O(\log(n))$ and was able to demonstrate successful reproduction of W States and Gaussian states with over 99% fidelity. We also demonstrated reproduction of random states to a fidelity greater than 99% with arbitrary unitary gates no larger than 4×4.

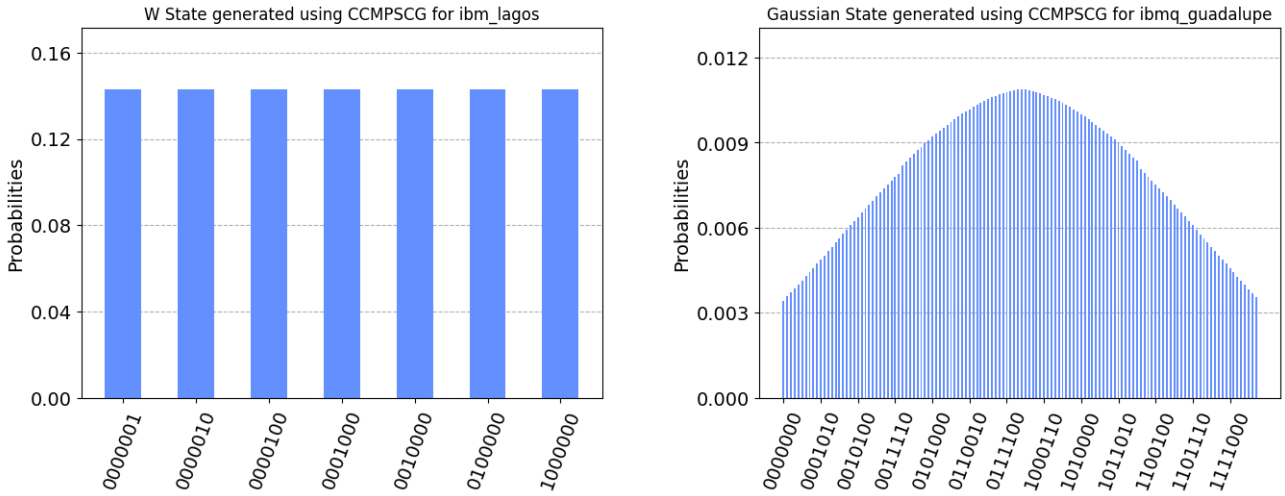
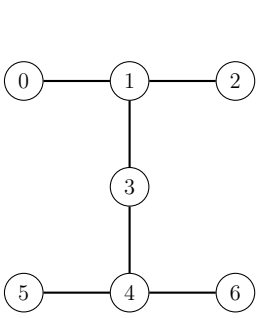
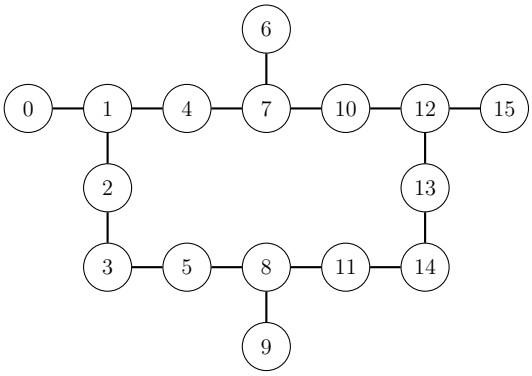


Figure 4.10: Histogram of Gaussian and W States produced using CCMPS for ibm_lagos



(a) 7 qubit ibm_lagos



(b) 16 qubit ibmq_guadalupe

Figure 4.11: Qubit layout of 2 IBMQ devices which we benchmarked our connectivity constrained procedure against

		CX	$U(\theta, \phi, \lambda)$	Depth
ibm_lagos	Gaussian	47 13	35 33	41 27
	W State	35 15	35 37	34 31
ibmq_quadalupe	Gaussian	235 36	84 88	74 41
	W State	220 35	84 86	75 39

Table 4.1: Gate counts and depths for circuits generated using LDMPSCG (in red) and CCMPSCG (in blue)

We found that the overall decomposition of these circuits in terms of CX and $U(\theta, \phi, \lambda)$ produced depths no larger than those generated using Qiskit's own state decomposition procedure. Lastly, we developed a proof of concept where circuits may be constructed which strictly obey some given qubit connectivity and demonstrated the successful reproduction of Gaussian and W-States to $O(n)$ with this constraint. This provides a framework in which for the development of low depth circuit generation for NISQ era devices with limited qubit connectivity.

Chapter 5

Conclusion

In this thesis, we used Matrix Product States (MPS) as a tool to explore problems in quantum computing and quantum information theory. To that end, we analysed the performance of quantum algorithms under noisy conditions and addressed the task of preparing quantum states on gate based quantum computers.

Starting with quantum algorithms we explored the classical simulatability and noise tolerance of Grover's Algorithm, the Quantum Approximate Optimisation Algorithm (QAOA) and recursive QAOA (RQAOA). We found that Grover's Algorithm was particularly simulatable as its relatively low bipartite entropy allows for the construction of large tensor networks without encountering an exponential increase in memory requirements. Regarding its noise tolerance, however, it was determined that the algorithm was very susceptible and would not be suitable to run on Near Intermediate-Scale Quantum (NISQ) devices. Simulating out large instances of Grover's Algorithm using MPS would be the natural extension beyond this thesis.

We verified the existence of Noise-Induced Barren Plateaus (NIBP) in QAOA and then analysed the recently proposed RQAOA, investigating whether or not it is susceptible to NIBPs in the same manner as its QAOA subroutine. We found that RQAOA was far less susceptible to NIBP when solving the Max-Cut problem compared to standard QAOA and hence it is a far better candidate algorithm to run on NISQ era devices. To that end, we then proposed techniques that may be utilised to strongly reduce the size of the circuits that need to be run for the RQAOA optimisation. We found that for 3-regular graphs one may achieve a 97% reduction in the number of qubits required (for large n) for level 1 RQAOA. Having demonstrated a path forward to execute RQAOA on a real quantum device, the next step here would be to complete this exercise and attempt successful simulation of RQAOA on existing quantum hardware.

We then considered the generation of circuits for arbitrary quantum states and determined new procedures which can reproduce states with limited bipartite entanglement with low depth to high fidelity. These procedures were benchmarked using Gaussian, Random and W states. We found that Gaussian and W States were reproducible with circuits of $O(\log(n))$ depth. This outperforms current best known circuits which are of $O(n)$, and in the case of W states, strongly outperforms Qiskit's state initialisation procedure which requires exponential depth. We further extended this procedure to determine circuits that obey the qubit connectivity of a given quantum device, minimising the number of swaps that would otherwise be required. Going forward, running these hardware specific circuits on the quantum device itself would be the natural extension to this thesis. We also believe that considering more complex tensor networks beyond the 1-dimensional MPS would possibly give rise to more efficient circuit constructions for a given hardware connectivity. Higher dimensional tensor networks may also allow for the simulation of more complex quantum algorithms, however with these higher dimensional tensor networks contraction strategy becomes a far more complicated task compared to its 1-dimensional counterpart.

Bibliography

[1] P. Benioff, Journal of Statistical Physics **22**, 563 (1980).

[2] R. P. Feynman, International Journal of Theoretical Physics **21**, 467 (1982).

[3] Y. Manin, Sovetskoye Radio, Moscow **128**, (1980).

[4] I. Kassal, S. P. Jordan, P. J. Love, M. Mohseni, and A. Aspuru-Guzik, Proceedings of the National Academy of Sciences **105**, 18681 (2008).

[5] N. Stamatopoulos, D. J. Egger, Y. Sun, C. Zoufal, R. Iten, N. Shen, and S. Woerner, Quantum **4**, 291 (2020), arXiv: 1905.02666.

[6] A. Montanaro, npj Quantum Information **2**, 15023 (2016).

[7] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, Annual Review of Condensed Matter Physics **11**, 369 (2020), arXiv: 1905.13641.

[8] G. Vidal, Physical Review Letters **91**, 147902 (2003).

[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, Cambridge, 2010).

[10] Deutsch, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences **400**, 97 (1985).

[11] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, arXiv:quant-ph/0001106 (2000), arXiv: quant-ph/0001106.

[12] J. J. Sakurai and J. Napolitano, *Modern quantum mechanics*, 2nd ed ed. (Addison-Wesley, Boston, 2011).

[13] R. Jozsa and N. Linden, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **459**, 2011 (2003).

[14] IBM Quantum, 2021.

[15] D-Wave Systems, 2021.

[16] F. Arute *et al.*, Nature **574**, 505 (2019).

[17] S. Aaronson and D. Gottesman, Physical Review A **70**, 052328 (2004), arXiv: quant-ph/0406196.

[18] R. Orus, Annals of Physics **349**, 117 (2014), arXiv: 1306.2164.

[19] A. Dang, Ph.D. thesis, The University of Melbourne, 2017.

[20] M. Treinish *et al.*, Qiskit/qiskit: Qiskit 0.30.0, 2021.

[21] J. Gray, Journal of Open Source Software **3**, 819 (2018).

[22] C. Roberts, A. Milsted, M. Ganahl, A. Zalcman, B. Fontaine, Y. Zou, J. Hidary, G. Vidal, and S. Leichenauer, arXiv:1905.01330 [cond-mat, physics:hep-th, physics:physics, stat] (2019), arXiv: 1905.01330.

[23] L. K. Grover, arXiv:quant-ph/9605043 (1996), arXiv: quant-ph/9605043.

[24] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl, Physical Review A **93**, 032318 (2016), arXiv: 1501.06911.

[25] K. Georgopoulos, C. Emary, and P. Zuliani, arXiv:2101.02109 [quant-ph] (2021), arXiv: 2101.02109.

[26] B. MEADE, L. LAFAYETTE, G. Sauter, and D. TOSELLO, 0 Bytes (2017).

[27] E. Farhi, J. Goldstone, and S. Gutmann, arXiv:1411.4028 [quant-ph] (2014), arXiv: 1411.4028.

[28] *Complexity and approximation: combinatorial optimization problems and their approximability properties*, softcover reprint of the hardcover 1. ed. 1999, 2. corr. printing ed., edited by G. Ausiello (Springer, Berlin, 2003).
[29] *The traveling salesman problem: a computational study*, *Princeton series in applied mathematics*, edited by D. L. Applegate (Princeton University Press, Princeton, 2006), oCLC: ocm83853510.
[30] M. X. Goemans and D. P. Williamson, Journal of the ACM (JACM) **42**, 1115 (1995), publisher: ACM New York, NY, USA.
[31] Y. Haribara, S. Utsunomiya, K.-i. Kawarabayashi, and Y. Yamamoto, arXiv:1501.07030 [quant-ph] **911**, 251 (2016), arXiv: 1501.07030.
[32] J. Håstad, Journal of the ACM **48**, 798 (2001).
[33] E. Farhi, D. Gamarnik, and S. Gutmann, arXiv:2004.09002 [quant-ph] (2020), arXiv: 2004.09002.
[34] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, arXiv:2007.14384 [quant-ph] (2021), arXiv: 2007.14384.
[35] J. A. Nelder and R. Mead, The Computer Journal **7**, 308 (1965).
[36] M. J. Powell, Mathematics Today-Bulletin of the Institute of Mathematics and its Applications **43**, 170 (2007), publisher: Citeseer.
[37] S. Bhatnagar, H. L. Prasad, and L. A. Prashanth, *Stochastic recursive algorithms for optimization: simultaneous perturbation methods*, No. 434 in *Lecture notes in control and information sciences* (Springer Verlag, London ; New York, 2013), oCLC: ocn794710408.
[38] P. Virtanen *et al.*, Nature Methods **17**, 261 (2020).
[39] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Nature Communications **12**, 1791 (2021), arXiv: 2001.00550.
[40] P. Erdös and A. Rényi, *The Structure and Dynamics of Networks* (Princeton University Press, ADDRESS, 2011), pp. 38–82.
[41] A. Hagberg, P. Swart, and D. S Chult, Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (unpublished).
[42] C. R. Harris *et al.*, Nature **585**, 357 (2020).
[43] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, Physical Review Letters **125**, 260505 (2020), arXiv: 1910.08980.
[44] A. W. Harrow, A. Hassidim, and S. Lloyd, Physical Review Letters **103**, 150502 (2009).
[45] V. Shende, S. Bullock, and I. Markov, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**, 1000 (2006).
[46] P. Niemann, R. Datta, and R. Wille, in *2016 IEEE 46th International Symposium on Multiple-Valued Logic (ISMVL)* (IEEE, Sapporo, Japan, 2016), pp. 247–252.
[47] M. Cramer, M. B. Plenio, S. T. Flammia, D. Gross, S. D. Bartlett, R. Somma, O. Landon-Cardinal, Y.-K. Liu, and D. Poulin, Nature Communications **1**, 149 (2010), arXiv: 1101.4366.
[48] F. Vatan and C. Williams, Physical Review A **69**, 032315 (2004), arXiv: quant-ph/0308006.
[49] F. Vatan and C. P. Williams, arXiv:quant-ph/0401178 (2004), arXiv: quant-ph/0401178.
[50] X. Zhan, *Matrix Inequalities* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2002), Vol. 1790, pp. 17–25, series Title: Lecture Notes in Mathematics.
[51] V. V. Shende, S. S. Bullock, and I. L. Markov, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**, 1000 (2006), arXiv: quant-ph/0406176.