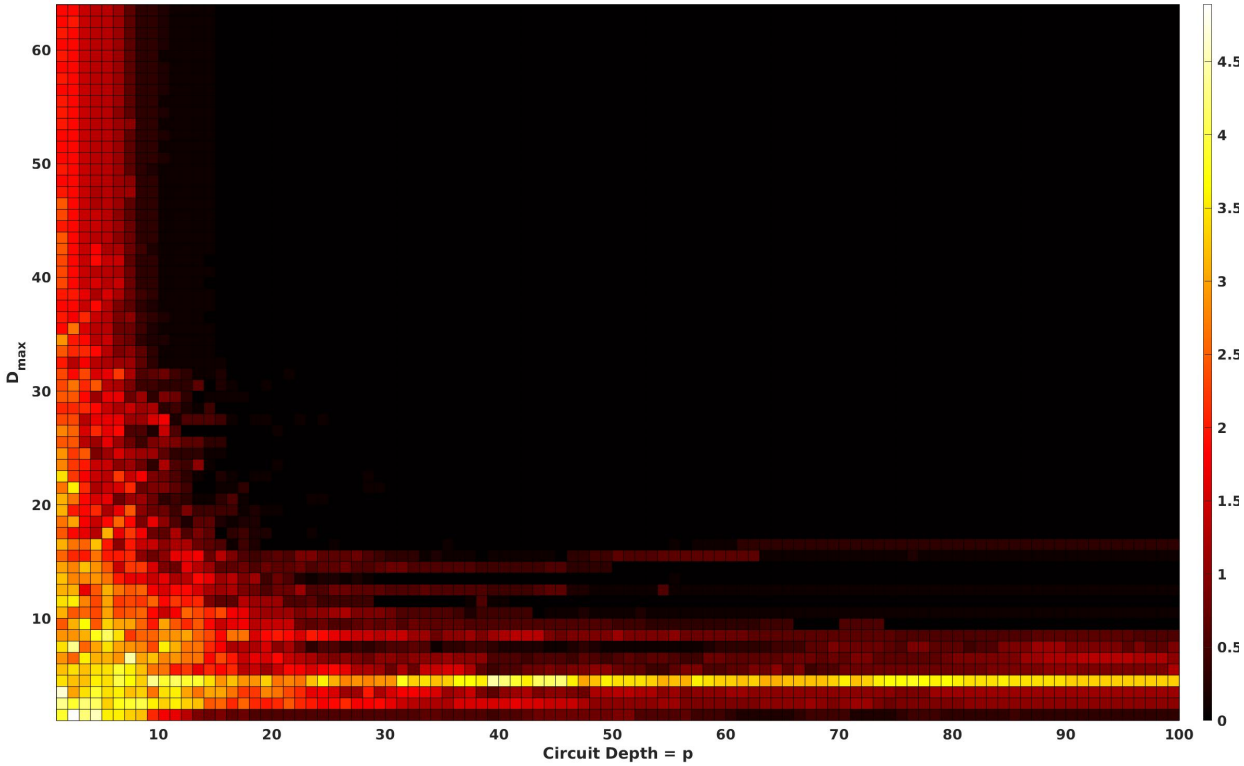


QIOA: Quantum Inspired Optimisation Algorithm

A Tensor Network based quantum-inspired classical algorithm for optimization problems

Master's thesis in Erasmus Mundus Joint Masters in Nanoscience and Nanotechnology, Specialising on Quantum Computing

Rishi Sreedhar

Department of Microtechnology and Nanoscience
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Funded by the Erasmus+ Programme of the European Union

Master's thesis 2020:MCCX04

QIOA
Quantum Inspired Optimisation Algorithm

A Tensor Network based approach to a Quantum Inspired Classical Optimization Algorithm

RISHI SREEDHAR

Funded by the
Erasmus+ Programme
of the European Union

Department of Microtechnology and Nanoscience
Division of Applied Quantum Physics
Göran Johansson Group
Chalmers University of Technology
Gothenburg, Sweden 2020

QIOA: Quantum Inspired Optimisation Algorithm
A Tensor Network based approach to Quantum Inspired Classical Algorithms
RISHI SREEDHAR

Promoter: Prof Göran Johansson,
Department of Microtechnology and Nanoscience,
Chalmers University of Technology.

Co-Promoter: Prof Bart Sorée,
Department of Electrical Engineering,
Katholieke Universiteit Leuven.

Examiner: Prof Thilo Bauch,
Department of Microtechnology and Nanoscience,
Chalmers University of Technology.

Daily Supervisor: Andreas Josefsson Ask,
Department of Microtechnology and Nanoscience,
Chalmers University of Technology.

Master's Thesis 2020:MCCX04
Department of Microtechnology and Nanoscience
Division of Applied Quantum Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A Hamming map of circuit-depth p vs Bond dimension $D_{\max}$ showing the number of character mismatches between predicted solution and actual solution.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

QIOA: Quantum Inspired Optimization Algorithm
A new Tensor Network based quantum inspired classical algorithm
to solve optimization problems

RISHI SREEDHAR
Department of Microtechnology and Nanoscience
Chalmers University of Technology &
Katholieke Universiteit Leuven

Abstract

The current era of Quantum computing has been aptly named Noisy Intermediate-Scale Quantum era, or NISQ era [1], owing to the highly promising, yet still imperfect state of available quantum hardware. The applicability of these NISQ devices available today is an area of research attracting increasing amounts of attention, as the field of quantum computing grows in popularity. Quantum-classical variational algorithms such as the Quantum Approximate Optimisation Algorithm [2], Variational Quantum Eigensolver [3], and Variational Autoencoder [4] are promising attempts in this direction, aiming to extract maximum value from NISQ devices. These algorithms aim to minimise the number of operations on the quantum processor by incorporating classical support-routine overheads.

The Quantum Approximate Optimisation Algorithm, also known as QAOA [2], is one such hybrid algorithm designed to solve optimisation problems. In this thesis, we study the QAOA using the framework of tensor networks and matrix product states (MPS). We make approximations on the quantum aspects of QAOA by limiting the amount of entanglement in QAOA circuits. Then, we analyse the effect of these approximations on its performance, by applying this approximated method on multiple Max-Cut and Exact-Cover problems of different sizes.

In the MPS formulation, it is the exponential increase of entanglement that makes exact simulations of highly entangled quantum circuits, such as QAOA circuits, classically infeasible. Our results suggest that it could still be possible to extract the solutions of the optimisation problems QAOA tries to solve, even from an approximated MPS-based QAOA-implementation with restricted entanglement. This added restriction in entanglement growth makes our implementation classically tractable. Hence, with this thesis, we propose a classical version of QAOA, which we are calling Quantum Inspired Optimisation Algorithm, QIOA, that is inspired by the original QAOA. We have validated the performance of QIOA on problems of small sizes, and further studies are required to better understand its limitations.

Keywords: Tensor Networks, QAOA, Quantum-inspired algorithms, Optimisation problems.

Acknowledgements

I am deeply grateful to my mentors Andreas Josefsson Ask and Pontus Vikstål for their limitless support and countless discussions. I learned about Tensor Networks from Andreas and QAOA from Pontus. Moreover, it was Pontus's vector-matrix implementation of QAOA that our tensor network-based calculations were benchmarked against. We also sourced the optimum parameters used in chapter 9, Figure 3.1, and the different problem instances described in chapter 4, from Pontus. I am also extremely grateful to my promoter Prof Göran Johansson for his invaluable feedback. His insights helped immensely in not only getting more clarity on the data obtained but also were pivotal in finding ways to improve upon the methods used. I'm also indebted to Prof Giulia Ferrini, Shahnawaz Ahmed, Marika Svensson, Laura García-Álvarez, the other master's students and colleagues at AQP for the multiple discussions and clarifications throughout last year. The entire Applied Quantum Physics group members have been extremely kind to me, and I am thankful for their support. Our program coordinator Prof Thilo Bauch has been an amazing pillar of encouragement throughout this program. It is difficult to express how reassuring it is to know that he is there to turn to when things go south. Thanks, Thilo. I am also thankful to Prof Bart Sorée, Prof Guido Groeseneken, Elke Delfosse and all other mentors and teachers from KU Leuven who have facilitated in helping me be where I am today. I'm also thankful to the EU and their Erasmus+ program for my scholarship, without which I couldn't have attended this amazing master's program. I have also been fortunate to have made so many great friends here at MC2 without whose support, encouragement, company and stimulating discussions, this program would have been less colourful. Special thanks to Ananthu Pullukattuthara Surendran for all his help throughout. My Erasmus batchmates have been another source of constant encouragement, support and discussions that I'm really grateful for. Also, thanks to David Frisk for the thesis template.

Finally, none of this would have been possible if not for the sacrifices and support of my family, and no finite set of words will do justice to how grateful I am for that. This thesis is dedicated to Eenippally.

Rishi Sreedhar, Gothenburg, October 2020

Contents

List of Figures

List of Tables

Part I

Introduction

1
Introduction

" There's Plenty of room at the Top [6]. "

The entire field of Quantum Computing is said to have been born from the now ubiquitous lecture given in 1959 by the famous physicist Prof Richard Feynman, titled "There is plenty of room at the bottom" [7]. While the ideas he had presented during that talk may not have much correspondence with the field of quantum information and quantum computing today [8], his 1984 conference paper titled "Quantum Mechanical Computers" [9] holds many parallels with the field, and was much ahead of its time. Quantum computing is considered by many as a complete paradigm shift on how we process information. Quantum computers operate on principles fundamentally different from digital transistor based classical computers because, unlike classical computers, they make use of quantum mechanical principles such as superposition and entanglement to directly work with information.

Arguably, the field of quantum computing only started getting wide-spread public attention after the invention of the famous Shor's Algorithm [10] in 1994. This is a polynomial-time quantum algorithm for solving the integer factorization problem [11]. Until then, finding the factors of large numbers were considered to be a computationally impossible problem, so much so that one of the most popular encryption protocols used to safeguard almost all modern internet transactions, the RSA protocol, was formulated based on this assumption [12]. Many believe this to be just a glimpse into the disruptive potential of quantum technologies [13]. This belief is further validated by recent breakthroughs in the field, such as the achievement of Quantum Supremacy by Google [14] where they performed a calculation using their 53-qubit superconducting quantum processor that would have taken the best classical supercomputing hardware existing today hundreds of years to complete. However, since Google's supremacy announcement, multiple research groups, with IBM in the lead [15], have come up with counter-arguments to Google's claims that the said calculation would take hundreds of years when performed classically. Nevertheless, everyone agrees that the future of quantum computing looks bright.

This constant tug between what is and is not feasible for quantum and classical hardware has been characterised as one of the signatures of the NISQ era [1] that we are in. This is because today, we are right at the quantum-classical boundary. This is also evidenced by multiple occasions where a proposed quantum algorithm

outperforms all existing classical algorithms for a while. However, this new algorithm leads to more insights on the problem it is trying to solve, which inspires better classical algorithms with improved performance which dethrone the initially proposed quantum algorithm. The QAOA [2] itself, which is one of the main topics of this thesis, has also been subject to this cycle (See chapter 3 for details). Another instance is when, in 2018, Ewin Tang, a bachelor's student then, came up with a quantum-inspired classical algorithm [16] which provided the same exponential speedup as a then-popular quantum algorithm for recommendation systems [17]. Since then, even just in the last two years, there have been multiple quantum inspired classical algorithms such as [18], [19], [20], [21], to name a few, that have continued to push the quantum-classical boundary.

In this thesis, we propose one such quantum inspired algorithm, called Quantum Inspired Optimisation Algorithm, or QIOA, that is inspired by the original QAOA. QIOA is born from tensor network based simulations of the original QAOA. Using tensor networks, we studied the performance of an approximated version of QAOA where the amount of entanglement within the quantum register is limited from growing exponentially. In the framework of matrix product states (MPS), which is the class of tensor networks we use, it is an exponential growth of entanglement within the registers that makes exact simulations of quantum circuits classically intractable. Our results show that in the instances we simulated, one is able to extract the solutions of problems QAOA is trying to solve, even from an approximated MPS based version of QAOA in which the entanglement has no exponential growth. This hints at the possibility that our MPS based implementation of QAOA is a new classical algorithm that takes inspiration from QAOA. However, at this point, QIOA is still only a promising avenue of exploration, and more extensive studies on bigger system sizes are required before making definitive claims.

Finally, this puts into perspective the epigraph used in this introduction, "There's Plenty of room at the Top [6]". This is the title of a paper discussing the future of classical algorithms in the face of a quantum future and a future of stalling advances in classical hardware developments. In this paper, the authors claim that the advances in classical computing would be at the top of the stack and quantum inspired classical algorithms are sure to carve a niche for themselves in this stack.

2
Tensor Networks

This introductory chapter gives a brief insight into the vast field of tensor networks. Note that this chapter only attempts to introduce the readers to the minimum amount of knowledge required to follow the methods and results discussed in this thesis. Most of the material presented in this chapter is sourced from extensive papers on the field such as [22], [23], [24], [25], [26]. Especially from [22] by Prof. Ulrich Schollwöck.

2.1 Tensors

We begin by describing a tensor, which is the most basic unit of a tensor network. There exist multiple, equally correct interpretations of a tensor, where each definition corresponds with the application they are used for. However, one common feature among all these definitions is the concept of tensor rank r, which is a number associated with all tensors. Some of the most popular and useful definitions of a tensor are listed below:

1. **As High Dimensional Matrices:** In this perspective, a tensor of rank r is an $r-$dimensional matrix. Hence, a rank 0 tensor is a scalar, rank 1 tensor a vector, rank 2 tensor a matrix and so on.

2. **As an element of an r-dimensional vector space:** Here, an r ranked tensor is treated as an element of an r dimensional vector space. Hence, if we denote our tensor as T, then $T \in \mathbb{C}^{d_1 \times d_2 \times \dots d_r}$ where d_i is the size of the i^{th} dimension. Now, if we represent the basis vectors of each dimension d_i as $\{|l_i\rangle\}$, then, we can expand our tensor T using these bases as follows:

$$T = \sum_{\forall\ l_1,l_2,\dots l_r} T_{l_1,l_2,l_3\dots l_r} \times |l_1\rangle|l_2\rangle \dots |l_r\rangle. \tag{2.1}$$

   This is the perspective we use when describing the wavefunctions of quantum states, as a tensor. This is pretty straight forward because Equation 2.1 looks exactly like the expansion of a wavefunction in some basis. For us, this would mean that a quantum state $|\psi\rangle$ which represents the wavefunction of an n-qubit quantum register would be a tensor of rank n. This is because $|\psi\rangle \in (\mathbb{H}^2)^{\otimes n}$, where $\mathbb{H}^2$ is a single-qubit Hilbert space.

3. **As a set of Linear maps from one space to another:** One way to define a linear map between two spaces, is to define how the basis states of the first

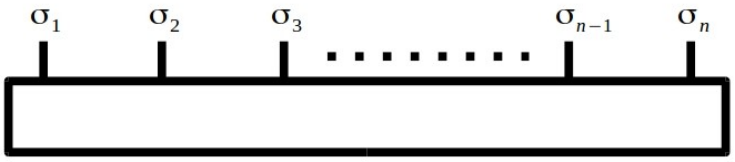
space, get mapped onto the basis states of the second space. Using the same mathematical definitions used in point 2, we can hence see a tensor T of rank r as a linear map from $\mathbb{C}^{d_1 \times d_2 \times \ldots d_k} \Longrightarrow \mathbb{C}^{d_{k+1} \times d_{k+2} \times \ldots d_r}$ for any $k \in \{1, 2, \ldots r\}$.

$$T(|l_1\rangle|l_2\rangle \ldots |l_k\rangle) = \sum_{\forall\ l_1, l_2, \ldots l_k, l_{k+1}, \ldots l_r} T_{l_1, l_2, \ldots l_k, l_{k+1}, \ldots l_r} \times |l_{k+1}\rangle|l_{k+2}\rangle \ldots |l_r\rangle. \quad (2.2)$$

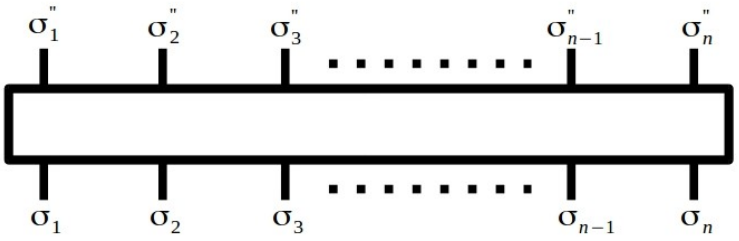
Here, the tensor T maps basis states $|l_1\rangle, \ldots |l_k\rangle$ to basis states $|l_{k+1}\rangle, \ldots |l_r\rangle$ through the scalar $T_{l_1, l_2, \ldots l_k, l_{k+1}, \ldots l_r}$. Depending on the number k separating the two spaces, T can simultaneously define a total of $r-1$ linear maps and hence is seen as a set of linear maps. In our case, we can use such a perspective to represent quantum gates. An n-qubit quantum gate acts on an n-qubit quantum register o produce another n-qubit quantum register. Thus, such a gate can be viewed as a rank $2n$ tensor mapping one rank n tensor onto another rank n tensor.

2.2 Pictorial Representation

Tensor Networks is a complete pictorial representation of many-body quantum systems, and other highly correlated systems. Here, continuing from point 2 and 3 from the above discussion, we present the pictorial representation of quantum states and quantum gates when viewed as a tensor. Any tensor T of rank r can be pictured as a closed shape with r legs sticking out of it. Each leg corresponds with labels pointing to their own respective spaces, where the labels take values from $1, \ldots d_i$, where d_i is the dimension of that subspace.



(a) An n-qubit Quantum State as a tensor.



(b) An n-qubit Quantum Gate as a tensor.

Figure 2.1: Pictorial representation of Quantum states and Gates in Tensor Network formalism.

In Figure 2.1a, each leg indexed by σ_i is called a physical index, and correspond to the state of the i^{th} qubit. This is also another advantage of tensor representations where we have means of directly addressing individual qubits in the full register. In an equivalent vector representation of an n-qubit register with a $2^n \times 1$ vector, addressing individual qubits are not that straightforward. Figure 2.1b depicts a rank $2n$ tensor which pictorially represents an n-qubit quantum gate. Here, this rank $2n$ tensor maps the set of physical indices $\sigma_1, \sigma_2, \sigma_3, \ldots \sigma_n$ to a possibly different set $\sigma_1^{"}, \sigma_2^{"}, \sigma_3^{"}, \ldots \sigma_n^{"}$. A 2-qubit gate would hence be a rank 4 tensor, in this perspective.

2.3 Tensor Operations

In this section, we briefly go over the two basic tensor operations, both mathematically, and pictorially.

2.3.1 Contraction

A contraction between indices of two tensors is a generalisation of matrix multiplication in higher dimensions. Mathematically, if we have two tensors T and M of ranks r and k respectively, then contracting a pair of indices, one from each tensor, would result in a new tensor TM with a rank of $r+k-2$. The contracted indices are completely summed over and no longer a part of this new tensor TM. Furthermore, both the indices have to be of the same dimension for a contraction to be possible. Mathematically this is explained in Equation 2.3 where s is the common index to be contracted in both T and M. An example is depicted pictorially in Figure 2.2 where both T and M are rank 6 tensors

$$\begin{aligned} T &= \sum_{\forall\ l_1,l_2,\ldots s,\ldots l_r} T_{l_1,l_2\ldots s,\ldots l_r} \times |l_1\rangle|l_2\rangle \ldots |s\rangle \ldots |l_r\rangle \\ M &= \sum_{\forall\ q_1,q_2,\ldots s,\ldots q_k} M_{q_1,q_2,\ldots s,\ldots q_k} \times |q_1\rangle|q_2\rangle \ldots |s\rangle \ldots |q_k\rangle \\ TM &= \sum_{\substack{\forall\ l_1,l_2,\ldots l_r \\ \forall\ q_1,q_2,\ldots q_k}} \left(\sum_s T_{l_1,l_2\ldots s,\ldots l_r} \times M_{q_1,q_2,\ldots s,\ldots q_k} \right) \times |q_1\rangle \ldots |q_k\rangle \times |l_1\rangle \ldots |l_r\rangle. \end{aligned} \tag{2.3}$$

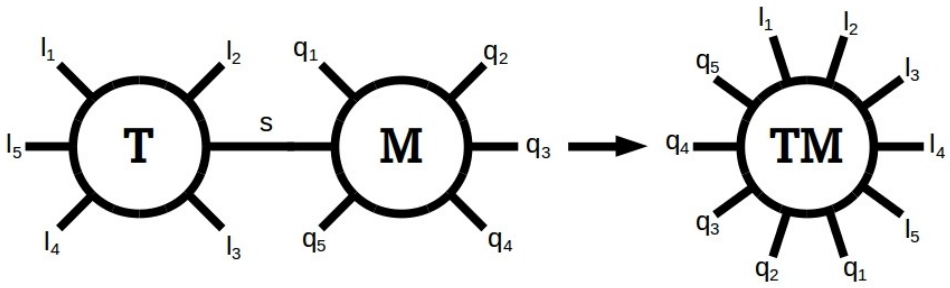


Figure 2.2: Pictorial depiction of a contraction operation between two rank 6 tensors T and M to produce a new rank 10 tensor TM.

Both Equation 2.3 and Figure 2.2, convey exactly the same idea. Nevertheless, comparing between them we can see how much more simple and easy it is to handle high-dimensional data by using tensor network notations. One just needs to contract over the common edges, and represent a new tensor with all the other remaining edges on it.

2.3.2 Tensor Products

Tensor Product operations are imperative in Quantum Mechanics and allow the Hilbert space of a multi-partite system to be expressed in terms of Hilbert spaces of its individual components. In the language of tensors, a tensor product between tensors T and M of ranks r and k results in a new tensor TM who has rank $= r + k$

$$\begin{aligned} T &= \sum_{\forall\ l_1,l_2,\ldots l_r} T_{l_1,l_2,\ldots l_r} \times |l_1\rangle|l_2\rangle \ldots |l_r\rangle \\ M &= \sum_{\forall\ q_1,q_2,\ldots q_k} M_{q_1,q_2\ldots q_k} \times |q_1\rangle|q_2\rangle \ldots |q_k\rangle \\ TM &= \sum_{\substack{\forall\ l_1,l_2,\ldots l_r \\ \forall\ q_1,q_2,\ldots q_k}} T_{l_1,l_2,\ldots l_r} \times M_{q_1,q_2,\ldots q_k} \times |q_1\rangle \ldots |q_k\rangle \times |l_1\rangle \ldots |l_r\rangle. \end{aligned} \tag{2.4}$$

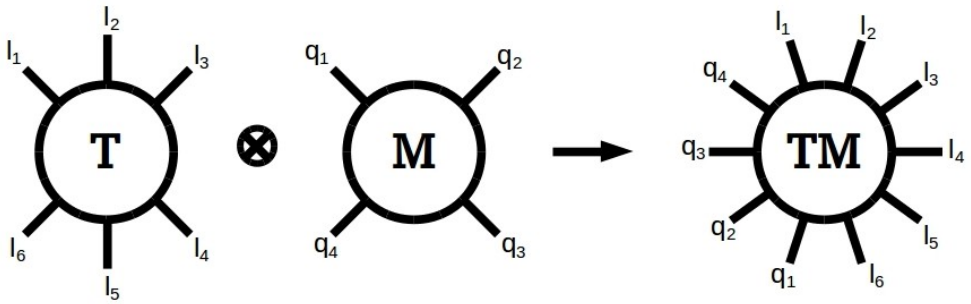


Figure 2.3: Pictorial depiction of a Tensor Product operation between a rank 6 tensor T and a rank 4 tensor M to produce a new rank 10 tensor TM.

2.4 Singular Value Decomposition

The singular value decomposition method, SVD for short, from linear algebra, is arguably the most important method required in the construction of an MPS representation of a quantum state $|\psi\rangle$. For any rectangular $m \times n$ matrix A, SVD guarantees that there exists a decomposition of A such that:

$$A = U \cdot S \cdot V^{\dagger}.$$

The dimensions of matrices U, S, and V are ($m \times \min(m,n)$), ($\min(m,n) \times \min(m,n)$), and ($\min(m,n) \times n$), where $\min(m,n)$ represents the minimum number between m and n. Matrix S here is a singular matrix with all other elements are

equal to 0 except the non-negative diagonal entries s_{aa}, also known as singular values. The singular values are generally arranged from increasing to decreasing order in S. That is, $S_{11} > S_{22} > S_{33} \cdots > S_{rr}$. The number of non-zero singular values in S is also called the Schmidt rank of the original matrix A. Also, if $m < n$, then the matrix U is unitary, that is $U \cdot U^{\dagger} = I$ and V is a rectangular matrix. If instead $m > n$, then V is a unitary. Both U and V are unitaries if $m = n$. This is depicted in Figure 2.4

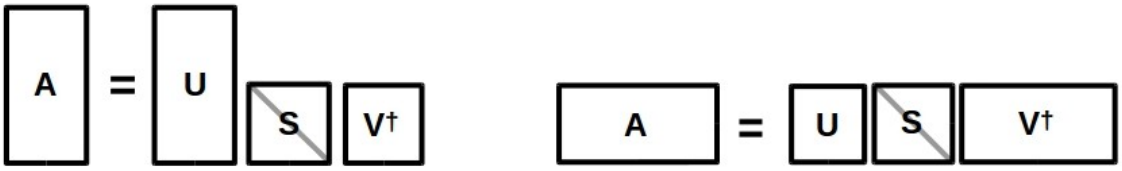


Figure 2.4: Pictorial depiction of SVD

Singular value decomposition is of significant practical importance in scenarios where one is interested in finding the most optimal approximation of matrix A with Schmidt rank r, by another matrix A' with a lower Schmidt rank r'. In that case, one can calculate A' by first calculating the SVD of A, and then replacing the original singular matrix S, whose Schmidt rank is r, with an approximated singular matrix S' which is the same as S with the only difference that all but the largest r' singular values are set to 0. Hence,

$$A' \;=\; U \cdot S' \cdot V^{\dagger}.$$

2.5 Schmidt Decomposition

The theorem for Schmidt decomposition implies that for any composite quantum state $|\psi\rangle$ that is an element of a tensor product space $\mathbb{H}_A \otimes \mathbb{H}_B$ can be expressed as

$$|\psi\rangle \;=\; \sum_{a=1}^{r} \lambda_a \cdot |u_a\rangle \otimes |v_a\rangle.$$

where $\{\ |u_1\rangle, |u_2\rangle, \ldots |u_r\rangle\ \}$ are orthonormal sets $\subset \mathbb{H}_A$ and $\{\ |v_1\rangle, |v_2\rangle, \ldots |v_r\rangle\ \}$ are orthonormal sets $\subset \mathbb{H}_B$. Here, r is the Schmidt rank associated with this quantum state $|\psi\rangle$. This can be easily derived as shown below using the singular value decomposition described above. Any pure state $|\psi\rangle \in \mathbb{H}_A \otimes \mathbb{H}_B$ can be expanded using tensor products of the respective bases in $\mathbb{H}_A$ and $\mathbb{H}_B$. Then, we could perform an SVD on the resultant coefficient matrix Ψ_{ij} and finally obtain the transformed bases $\{|u_a\rangle\}$ and $\{|v_a\rangle\}$, along with the singular values s_a. That is, $|\psi\rangle \;=\; \sum_{i,j} \Psi_{ij} \times |i_A\rangle \otimes |j_B\rangle$, where, $\Psi_{ij} \;=\; \sum_{a}^{\min(N_A, N_B)} U_{i,a} \cdot S_{aa} \cdot V^{\dagger}_{a,j}$. Now,

$$
\begin{aligned}
|\psi\rangle &= \sum_{i,j} \left(\sum_{a}^{\min(N_A,N_B)} U_{i,a} \cdot S_{aa} \cdot V^{\dagger}_{a,j} \right) \times |i_A\rangle \otimes |j_B\rangle \\
&= \sum_{a}^{\min(N_A,N_B)} S_{aa} \cdot \left(\sum_{i} U_{i,a} \cdot |i_A\rangle \right) \otimes \left(\sum_{j} V^{\dagger}_{a,j} \cdot |j_B\rangle \right) \qquad (2.5) \\
&= \sum_{a=1}^{\min(N_A,N_B)} \lambda_a \cdot |u_a\rangle \otimes |v_a\rangle.
\end{aligned}
$$

Here, N_A and N_B are the dimensions of $\mathbb{H}_A$ and $\mathbb{H}_B$ respectively. From Equation 2.5 we see that the Schmidt rank [27] of a bi-partite quantum state $|\psi\rangle$ is the same as the Schmidt rank of the coefficient matrix Ψ_{ij}. Moreover, the Schmidt coefficients $\{\lambda_a\}$ are nothing but the singular values of Ψ_{ij}. The Schmidt coefficients $\{\lambda_a\}$ is used as a measure of how entanglement one part of the system is with another. One such measure, the von Neumann entropy is defined as:

$$\text{Entropy}_{A|B}(|\psi\rangle) = -\sum_{a=1}^{r} |\lambda_a|^2 \log_2(|\lambda_a|^2).$$

Main insights derived from the above discussion are:

1. The maximum possible Schmidt rank for a bipartite quantum state is the minimum of the dimensions of the two partitions. That is, $r = \min(N_A, N_B)$. This point will be of significance when we are talking about bond-dimensions below in subsection 2.6.2.

2. If Schmidt rank $r = 1$, we only have a product state with no entanglement. Having $r > 1$ is a necessary and sufficient condition for having an entangled state.

3. To obtain a less entangled state $|\psi'\rangle$ which is the closest approximation to state $|\psi\rangle$, one only need to reduce the Schmidt rank of $|\psi\rangle$ from r to r' by setting the smallest $r - r'$ Schmidt weights, $\lambda_{r'+1}, \ldots \lambda_r$, to 0.

2.6 Matrix Product States

Intuitively, we know that a product state of n qubits, only require at max $2n$ parameters to define it. However, when representing states as vectors, we would still require to have a $2^n \times 1$ vector to define any quantum state, irrespective of its nature of entanglement. Intuitively, we can see that this is an overkill. Matrix product states, among other things, are a way of mathematically capturing that intuition.

A matrix product state, or MPS for short, is a linearized representation of a many-body quantum state. Although any quantum state can be represented exactly as

an MPS, the number of parameters required to express fully entangled systems grows exponentially with system size. (This will be proved below.) However, there exist analytical proofs that the ground states of one-dimensional quantum systems with gapped Hamiltonians (Hamiltonians with a finite energy difference between the ground state and first excited state) having only local interactions, can be efficiently represented as an MPS [28]. In this section, we briefly go over its derivation and describe how an MPS representation can give insights into the nature of entanglement in quantum systems.

2.6.1 MPS Derivation

Any quantum state $|\psi\rangle$ composed of n qubits can be expanded using a basis which is the tensor product of the individual local site bases $\{|\sigma_i\rangle\}$. That is,

$$|\psi\rangle \;=\; \sum_{\sigma_1,\sigma_2,\ldots\sigma_n} C_{\sigma_1,\sigma_2,\sigma_3,\ldots\sigma_n} \times |\sigma_1\rangle \otimes |\sigma_2\rangle \otimes |\sigma_3\rangle \otimes \cdots \otimes |\sigma_n\rangle.$$

Since we are dealing with qubits here, each local space $\{|\sigma_i\rangle\}$ is of dimension 2 (simply replace 2 by d when dealing with qudits with d levels). Hence, the full coefficient vector has an exponentially growing length of 2^n components. We can reshape this $2^n \times 1$ coefficient vector into a $2 \times 2^{n-1}$ matrix and call it Ψ such that:

$$\Psi_{(\sigma_1),(\sigma_2,\sigma_3,\ldots\sigma_n)} = C_{\sigma_1,\sigma_2,\sigma_3,\ldots\sigma_n}.$$

Next, we perform an SVD on this $2 \times 2^{n-1}$ matrix Ψ and absorb the singular matrix S into $V^\dagger$ to obtain $C_{a_1,(\sigma_2,\sigma_3,\ldots\sigma_n)}$.

$$\begin{aligned} C_{\sigma_1,\ldots\sigma_n} &= \Psi_{(\sigma_1),(\sigma_2,\sigma_3,\ldots\sigma_n)} \\ &= \sum_{a_1}^{r_1} U_{\sigma_1,a_1} \cdot \left(S_{a_1,a_1} \cdot V^\dagger_{a_1,(\sigma_2,\sigma_3,\ldots\sigma_n)}\right) \\ &= \sum_{a_1}^{r_1} U_{\sigma_1,a_1} \cdot \left(C_{a_1,(\sigma_2,\sigma_3,\ldots\sigma_n)}\right). \end{aligned} \tag{2.6}$$

Here r_1 is the Schmidt rank of the bipartition of $(\mathbb{H}^2)^{\otimes n}$ into $(\mathbb{H}^2) \otimes (\mathbb{H}^2)^{\otimes(n-1)}$. Now we proceed exactly the same way by reshaping $C_{a_1,(\sigma_2,\sigma_3,\ldots\sigma_n)}$ into an $(r_1 \cdot 2) \times (2^{n-2})$ matrix $\Psi_{(a_1,\sigma_2),(\sigma_3,\sigma_4,\ldots\sigma_n)}$, and perform an SVD on that. After that, we absorb the Singular matrix S into $V^\dagger$ again, and repeat this process till the last node. That is,

$$
\begin{aligned}
C_{\sigma_1,\ldots\sigma_n} &= \sum_{a_1}^{r_1} U_{\sigma_1,a_1} \cdot \Big(C_{a_1,(\sigma_2,\sigma_3,\ldots\sigma_n)}\Big) \\
&= \sum_{a_1}^{r_1}\sum_{a_2}^{r_2} U_{\sigma_1,a_1} \cdot U_{a_1,\sigma_2,a_2} \cdot \Big(S_{a_2,a_2} \cdot V^{\dagger}_{a_2,(\sigma_3,\sigma_4,\ldots\sigma_n)}\Big) \\
&= \sum_{a_1}^{r_1}\sum_{a_2}^{r_2} U_{\sigma_1,a_1} \cdot U_{a_1,\sigma_2,a_2} \cdot \Big(C_{a_2,(\sigma_3,\sigma_4,\ldots\sigma_n)}\Big) \\
&\quad\vdots \\
&= \sum_{a_1}^{r_1}\sum_{a_2}^{r_2}\cdots\sum_{a_{n-1}}^{r_{n-1}} U_{\sigma_1,a_1} \cdot U_{a_1,\sigma_2,a_2}\ldots U_{a_{n-2},\sigma_{n-1},a_{n-1}} \cdot \Big(C_{a_{n-1},(\sigma_n)}\Big).
\end{aligned}
\tag{2.7}
$$

Finally, we see that the tensors U are all rank 3 tensors (after assigning dummy indices of dimension 1 to the first and last tensors), where the indices σ_i are called physical indices, as they point to the respective local Hilbert spaces. The indices a_k are referred to as virtual indices, as they exist only within the MPS representation. To further clarify this distinction, we reshape U_{a_{i-1},σ_i,a_i} into a set of 2 $a_{i-1} \times a_i$ matrices indexed by σ_i, such that $U_{a_{i-1},\sigma_i,a_i} = M^{\sigma_i}_{a_{i-1},a_i}$. Finally putting everything together,

$$
\begin{aligned}
C_{\sigma_1,\ldots\sigma_n} &= \sum_{a_1,a_2,\ldots a_{n-1}} M^{\sigma_1}_{1,a_1} \cdot M^{\sigma_2}_{a_1,a_2} \cdot M^{\sigma_3}_{a_2,a_3}\ldots M^{\sigma_n}_{a_{n-1},1} \\
&= M^{\sigma_1} \cdot M^{\sigma_2} \cdot M^{\sigma_3}\ldots M^{\sigma_n},
\end{aligned}
\tag{2.8}
$$

which gives,

$$
|\psi\rangle = \sum_{\sigma_1,\sigma_2,\ldots\sigma_n} M^{\sigma_1} \cdot M^{\sigma_2}\ldots M^{\sigma_n} \times |\sigma_1\rangle \otimes |\sigma_2\rangle \otimes \cdots \otimes |\sigma_n\rangle. \tag{2.9}
$$

Equation 2.8 is also from where Matrix Product States get their name, as each state coefficient $C_{\sigma_1,\ldots\sigma_n}$ is expressed as a product of matrices $M^{\sigma_i}_{a_{i-1},a_i}$ in an MPS. While the mathematical derivations provided above are cumbersome, they provide exact insights onto the inner workings of matrix product states. However, the same derivation can also be expressed pictorially, using the tensor network notations introduces above. Recall that an n-qubit quantum register is a rank n tensor and represented as shown in Figure 2.1a. Thus, the operations depicted in Equation 2.6 and Equation 2.7 are represented using Tensor Network notations in Figure 2.5. Here, however, unlike in the derivation, we have not absorbed the singular matrices S into the $V^{\dagger}$ as was done in the mathematical derivations. They have been left as is, in the form of red diamonds, in the figure. This is because although in the above discussions, and in Figure 2.5, we have started performing the SVD operation from the left, we could have just as correctly started these operations from the right,

by absorbing the singular matrices S into U. In the former case, we are left with what is called left-canonicalised MPS, while the latter leaves us with right-canonical MPS. The choice between left and right canonicalization is arbitrary and related to gauge-freedoms.

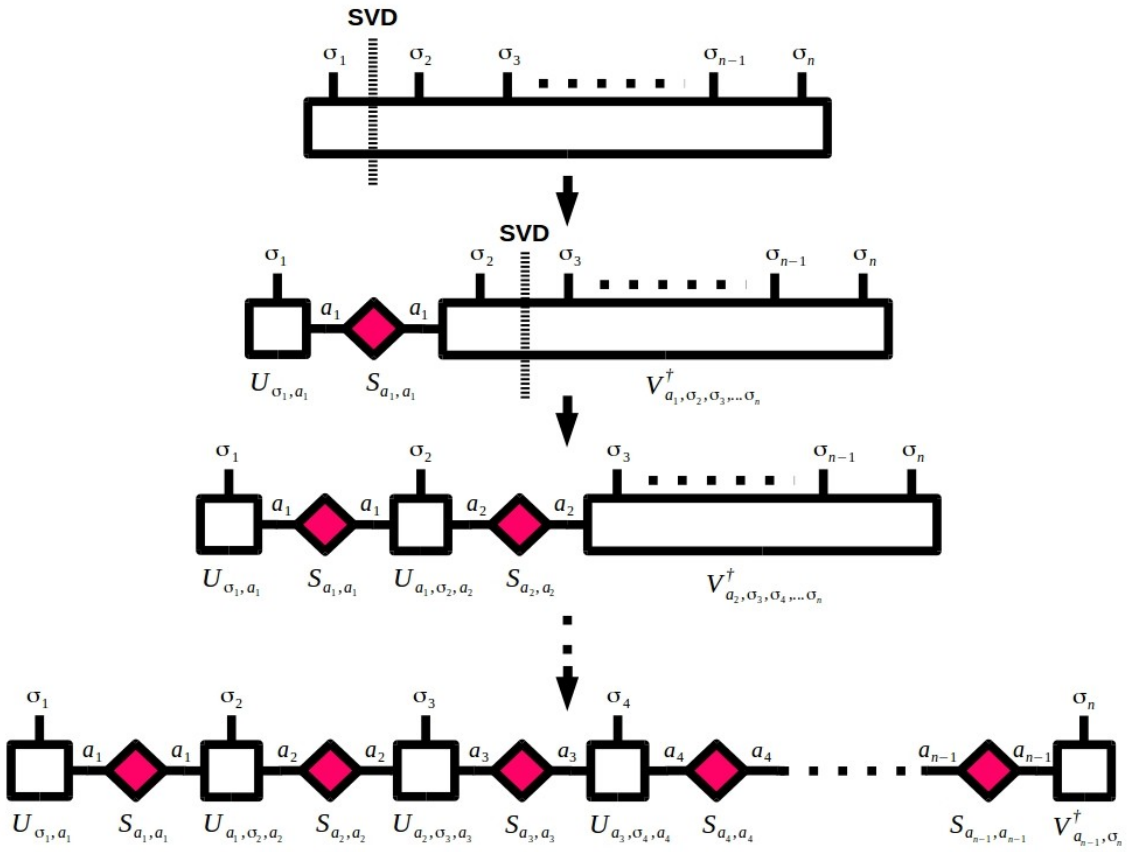


Figure 2.5: Conversion of a rank n Tensor into an n-register MPS.

2.6.2 Bond Dimension and Entanglement

Till now, it may not seem like we have done much to our original quantum state other than some heavy mathematical operations. However, putting together the above three discussions, section 2.4 on SVD, section 2.5 on the Schmidt decomposition, and subsection 2.6.1 on MPS derivation, we can now get a much clearer picture.

Firstly, note that in Equation 2.6, r_1 is the Schmidt rank of the bipartition of $(\mathbb{H}^2)^{\otimes n}$ into $(\mathbb{H}^2) \otimes (\mathbb{H}^2)^{\otimes(n-1)}$. Hence, from section 2.5, we know that $r_1 \leq \min(N_A, N_B)$, where partition $A = \mathbb{H}^2$ and partition $B = (\mathbb{H}^2)^{\otimes(n-1)}$. This makes $N_A = 2$ and $N_B = 2^{n-1}$, making $r_1 \leq 2$. Choosing $r_1 = 2$ would mean using all available degrees of freedom from the Hilbert space of partition A. Proceeding onward with SVD along the chain, we can see that $r_2 \leq 2r_1$, $r_3 \leq 2r_2r_1$ and so on, until one reaches the middle of the chain. The Schmidt rank increases till the middle because till then, when starting from the left, $N_A < N_B$. Furthermore, note that the Schmidt rank r_i is also the number of retained Singular values from the Singular matrix S_i

appearing on creating a partition between physical indices σ_i and σ_{i+1}. Recalling from section 2.5, this is of particular physical relevance because the retained singular values dictate the amount of retained entanglement within the bipartition.

This Schmidt rank r_i is also known as the Bond-Dimension d_i between the tensors representing qubits i and $i+1$. The term bond-dimension would be used throughout the rest of this thesis and is one of the most important ideas raised in this chapter. Bond-dimensions are so named because the virtual index connecting the tensors of the two qubits i and $i+1$ is also known as a bond in tensor network notation. Furthermore, the bond-dimensions connecting two bipartitions also tell us the number of correlated degrees of freedom within them. How well a matrix product state can represent a given quantum state also depends a lot on the nature of these bond-dimensions. For instance, continuing from the above discussion on the growth of Schmidt ranks r_i (or equally the growth of bond-dimension d_i), we see that in a case where we do not make any approximations, d_i grows exponentially along the chain. That is, $d_1 = 2$, $d_2 = d1 \times 2 = 2^2$, $d_3 = d_2 \times 2 = 2^3$ etc. This goes on till one reaches the middle of the chain. Here, depending on if the chain has an odd or even number of qubits, we see different behaviour. That is,

1. If we have an even-numbered qubit register where $n = 2m$ for some $m \in \mathbb{N}$, then the largest possible Schmidt rank is obtained through an equal partition of the full Hilbert space $(\mathbb{H}^2)^{\otimes 2m}$ into $(\mathbb{H}^2)^{\otimes m} \otimes (\mathbb{H}^2)^{\otimes m}$. In such a case, the maximum possible bond dimension is held by bond d_m which is the bond connecting the two partitions of equal size m, and $d_m = 2^m$ which is also to say, $d_m = 2^{n/2}$. Also note that there is only one bond d_m that holds this highest value, as there is only one such possible partition.

2. Now on the other hand if we had an odd number of qubits in our register such that $n = 2m + 1$ for some $m \in \mathbb{N}$, then we no longer have a single bond with the highest possible bond-dimension. This is because in this scenario, there exists a central tensor, and partitions on both the left and right of that node provide the highest bond-dimensions. Here too the maximum bond-dimension is equal to 2^m. However, there are now two bonds who have this dimension, d_m and d_{m+1}.

The significance of this above discussion on odd and even qubit registers will become clear in chapter 9, section 9.5, when describing some results for odd numbered qubit registers. In both case of odd and even n, we see that the maximum bond-dimension is the same, and proportional to 2^m, where m is as defined above. The mathematical operation that allows one to extract m from n is to take the floor value of $n/2$. The floor of a number n, denoted by $\lfloor n \rfloor$ is the largest integer below that number. For instance, $\lfloor 1.89 \rfloor = 1$. Thus, for both odd and even cases, $m = \lfloor n/2 \rfloor$. Thus the highest possible bond-dimension of an n qubit MPS register is given by,

$$D_{\text{highest}} = 2^{\lfloor n/2 \rfloor}. \tag{2.10}$$

Here it is very important to notice that exact representations of highly entangled n-body quantum states need at-least one of the bonds to have this highest dimension of $2^{\lfloor n/2 \rfloor}$. In that case, using matrix product states are inefficient, and in fact, more expensive than normal vector representation which only uses one $2^n \times 1$ vector. This is because the highest sized tensor itself need $2 \times \left(2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}\right)$ parameters in addition to the other tensors.

However, the bond-dimension d_i is the number of Schmidt coefficients obtained from a bipartition between qubit i and $i+1$, and it is these Schmidt coefficients that characterise the amount of entanglement between the two partitions. Hence, using insights derived from section 2.5, one of the most straightforward ways to approximate a given quantum state $|\psi\rangle$ is by limiting the number of Schmidt coefficients that exist in any bi-partition pairs in that state. Matrix product states provide an extremely straightforward way to do this, as in an MPS, one could easily do this by setting upper bound $D_{\max}$ on all the bond-dimensions. Moreover, matrix product states are ideal ansatz candidates to represent quantum states where we know that the entanglement, and hence the bond-dimension, does not scale exponentially with system size. Ground states for local and gapped Hamiltonians, of 1D-systems especially, fall under this category. There even exist analytical proofs which show the efficacy of MPS in these scenarios [28].

Currently, there exist no analytical proofs on how good a matrix product state ansatz is in capturing the essence of all to all connected systems, such as QAOA circuits described in chapter 3. This is the primary question we are trying to answer with this thesis, and our findings described in chapter 8 and chapter 9 show promising prospects on the applicability of MPS in these scenarios too.

2.6.3 Product States

In this section, we talk about the extreme case obtained when the maximum bond-dimension $D_{\max}$ within the full matrix product state register is set to 1. Do note that, although $2^{\lfloor n/2 \rfloor}$ is the highest possible bond-dimension in an n-qubit matrix product register, it does not mean that all one needs such high bond-dimensions to describe all quantum states. For instance, a pure quantum state that is already a product state can be represented exactly by using an MPS with $D_{\max} = 1$. This is because product states are composed of un-correlated individual systems and hence have all bipartitions Schmidt rank = 1. Also, the i^{th} tensor within the matrix product state has a dimension of $d_{i-1} \times 2 \times d_i$. Since i product states all $d_i = 1$, all the tensors within a product states are $1 \times 2 \times 1$ tensors, requiring only a total of $2n$ parameters. This proves the intuition that was introduced at the beginning of this section.

Here, it is also worth noting that bonds with a dimension of 1 are considered as trivial bonds because they represent a tensor product between two un-correlated parts of the system. It is also in this regard that dummy indices of dimension 1 were

introduced to the matrices $M^{\sigma_1}_{1,a_1}$ and $M^{\sigma_n}_{a_{n-1},1}$ in Equation 2.8. Finally, Figure 2.6 shows the pictorial depiction of a matrix product state where σ_i denotes the i^{th} physical index, and d_i, the i^{th} bond-dimension. The trivially introduced bond of dimension 1 is shown in red. However, do note that it is only here in our system where this bond is only introduced for consistency reasons. Matrix product states used to represent periodic systems have a bond-dimension > 1, for this bond.

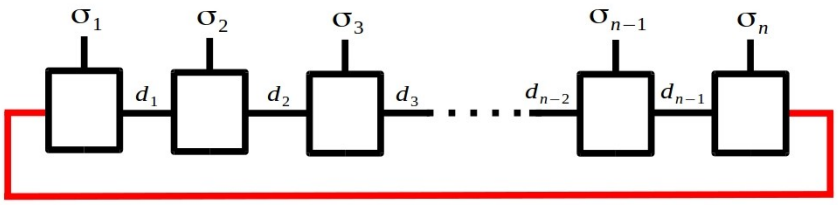


Figure 2.6: Final representation of a matrix product state. Trivial bond introduced for consistency shown in red.

2.7 Matrix Product Operators

Figure 2.1b presents a rank $2n$ tensor that is an n-qubit gate. It takes a rank n tensor as input, by which we mean that it performs contractions over the corresponding edges, and gives another rank n tensor as output. One can perform exactly the same set of operations that were performed onto a rank n tensor (Figure 2.1a) to convert it into an n register matrix product state (Figure 2.6), also onto this rank $2n$ gate to obtain an n-qubit matrix product operator, or MPO for short. The benefit of using an MPO is that it preserves the structure of an MPS even after gate operation. That is, an MPO takes an MPS as input, and provides another MPS as output.

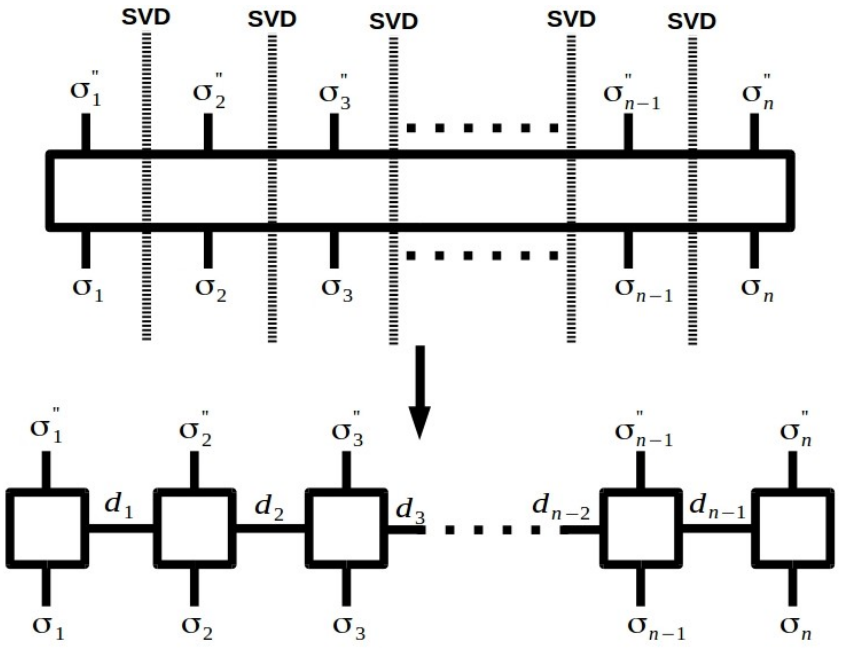


Figure 2.7: Conversion of a rank $2n$ gate into an n-qubit MPO

Figure 2.7 depicts the formation of an n-qubit MPO through multiple SVD operations performed on the original rank $2n$ tensor. Here too, the bond-dimension d_i gives insights into the entangling power of the gate. If $d_i = 1$, then that particular gate will create no added entanglement between qubits i and $i+1$. Note that in this thesis, we have only employed single-qubit gates and nearest neighbour two-qubit gates. See chapter 5, section 5.2 for further details on the gates used.

2.8 Quantum Circuits as Tensor Networks

With this, we have all the required tools to represent Quantum circuits as tensor networks. We use matrix product states to represent our quantum register, and since we are only employing nearest neighbour gates in this thesis, we have not resorted to using MPOs. Operating a gate onto a particular set of qubits would be performed through a contraction between the physical indices of those qubits, and the corresponding input indices of that gate.

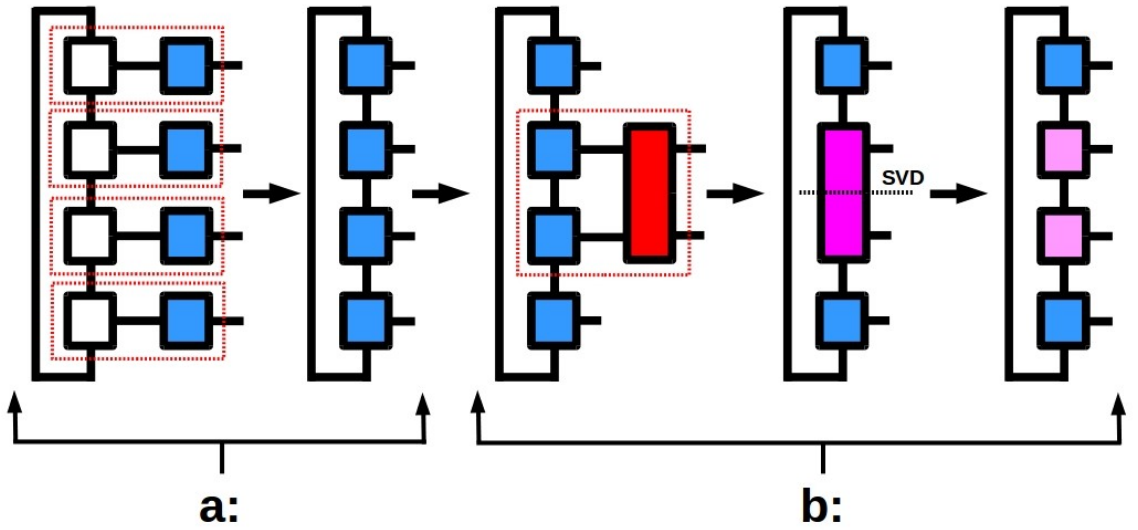


Figure 2.8: An example of Tensor Network implementation of single and two qubit gates on a 4 qubit matrix product state. a: single-qubit operations. part b: two-qubit operation. Tensor contractions are performed on indices encapsulated by red-dashed lines.

In Figure 2.8, part a: depicts the action of 4 single-qubit gates, (shown in blue), acting on all the qubits in a 4 qubit MPS register. section 2.8 also presents a full tensor network diagram of the QAOA algorithm, for circuit depth $p = 1$, without representing any of the contraction and truncation operations. Do note that single-qubit gates are incapable of changing the amount of entanglement in the quantum state, and hence do not alter the bond-dimension of the MPS.

Next, part b: shows the action of a two-qubit gate on qubits 2 and 3. In this thesis, since we are using only nearest neighbour gates, (non-local gates are implemented

via SWAP networks as described in subsubsection 5.2.2.2) we do not represent the gates as MPOs. All the bond-indices within the red-dashed lines are contracted to obtain a combined rank 4 tensor shown in magenta. Next, we perform an SVD along the dotted lines to return the register back to being a matrix product state. Note that two-qubit gates have the capability of creating entanglement, and hence, change the bond-dimension across the bonds they act on. If left unchecked, this could lead to an exponential rise in the bond-dimension, and hence, we can set a cap on the bond-dimensions by setting a limit $D_{\max}$ and keeping only the largest $D_{\max}$ singular values after performing each SVD. This is explained more in detail in chapter 5, section 5.3.

2.9 Reduced Density Matrix Calculations

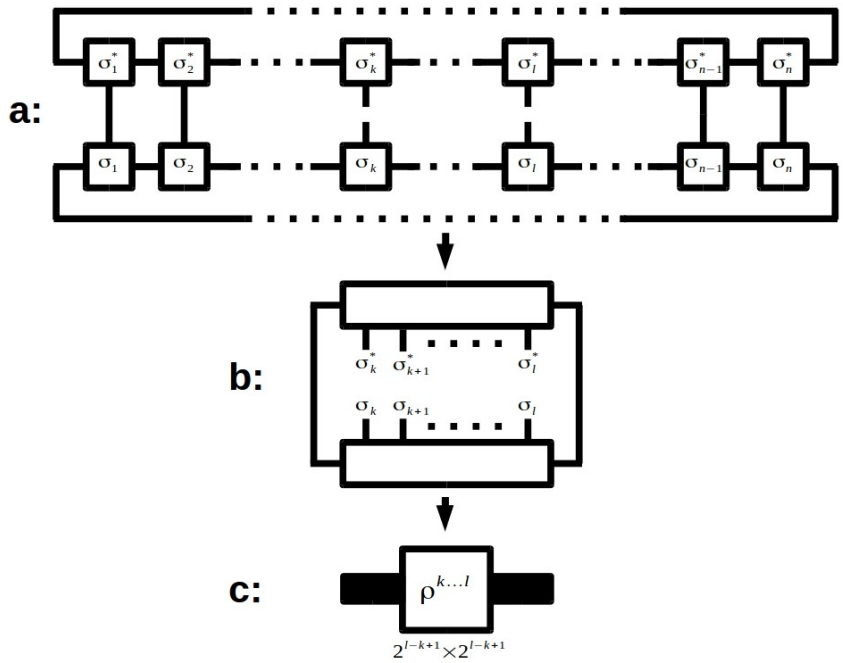


Figure 2.9: Generalised Tensor Network representation of Calculating the Reduced Density Matrix of qubits from k to l including k and l. a: The Tensor Network Diagram. b: All contractable edges contracted to give a rank $2(l-k+1)$ tensor with each index of dimension 2. c: All dangling edges compressed to give the $2^{(l-k+1)} \times 2^{(l-k+1)}$ density matrix $\rho^{k\cdots l}$.

We end this chapter by providing a way to calculate the reduced density matrix $\rho^{k\cdots l}$ of qubits from and including k to l from an n qubit matrix product state register in Figure 2.9. Reduced density matrix calculations are imperative in this thesis, as the QIOA method described in this thesis depend heavily on calculating them. Kindly turn to chapter 7 for more details on this. Further tensor network operations will be introduced in the following chapters upon requirement. For instance, expectation value calculations will be discussed in chapter 5, Figure 5.5.

3
QAOA

The Quantum Approximate Optimization algorithm, more commonly referred to as QAOA, is a hybrid quantum-classical algorithm proposed by Farhi et al. in late 2014 [2]. Although for a short while, QAOA was the best performing algorithm that solved constraint satisfaction problems [11], by showing that QAOA can beat random guessing [29], this was soon outperformed by a new classical algorithm proposed by Barak et al. [30]. Next, in 2016, Farhi et al. proved that it is classically impossible to sample from the output of a QAOA circuit even for the smallest circuit depth unless the polynomial hierarchy collapses, and P = NP [31]. This point will be discussed more in detail towards the end of this chapter, as we want to establish that although QIOA is a classical algorithm based on QAOA, it is not performing exact classical sampling of QAOA circuit outputs. Hence, our results do not violate any of the already existing literature. Furthermore, the quantum supremacy result by Farhi et al. could easily be intuitively understood using the tensor network framework that was explained in the previous chapter (chapter 2). Before revisiting these points, it could be instructive to briefly go over the original QAOA method through the next section. Also, please note that after the introduction of the original QAOA by Farhi et al., many other flavours of QAOA were introduced, such as Tree-QAOA [32], re-interpreted QAOA [33], Grover-QAOA [34] and many more, all of which are still quantum-classical hybrid algorithms based upon the original. In this light, the consensus is to call the original flavour of QAOA as vanilla-QAOA, which we too follow.

3.1 Quantum Adiabatic Algorithm

QAOA is a heuristic algorithm that has undoubtedly drawn considerable attention to the field of quantum-classical hybrid algorithms, especially because of the potential of these algorithms to be NISQ [1] accessible. The main idea behind QAOA was however inspired from a preexisting quantum algorithm called the Quantum Adiabatic Algorithm (QAA), also proposed by Farhi et al. [35]. Hence, before going over the original QAOA method, it could be beneficial to review the QAA algorithm. The QAA makes use of the adiabatic theorem in quantum mechanics. This theorem states that a quantum system which is initially in the ground state of a starting Hamiltonian, will remain in the lowest possible energy state even if the system Hamiltonian varies to another, given that the time T over which this variation happens is slow enough. In the infinite limit, when $T \longrightarrow \infty$, the system always exactly remains in the ground the state. That is, if we have a time varying

Hamiltonian,

$$H_{QAA}(t) = H_C\left(\frac{t}{T}\right) + H_B\left(1 - \frac{t}{T}\right).$$

Here, H_B is the Hamiltonian at time $t = 0$, whose ground state is easy to prepare and hence is the state used to initialise the system with. H_C is the final Hamiltonian which encodes the problem we would like to solve and whose ground state is what we are interested in finding. However, an issue here is that this method calls for large evolution time T to work in practice, which would correspondingly require long coherence times for the quantum computers. Hence it is not NISQ accessible.

3.2 QAOA Method

QAOA was proposed as a solution to this problem of needing qubits with long coherence times. It is essentially a trotterised version of QAA. From the Schrödinger equation, we know that the time evolution of a Hamiltonian $\hat{H}(t)$ is given by the operator $e^{-i\hat{H}(t)}$. Also, the "Suzuki-Trotter expansion" states that

$$e^{A_1+A_2+\ldots A_p} = \lim_{n\to\infty}\left(e^{\frac{A_1}{n}} \cdot e^{\frac{A_2}{n}} \cdot e^{\frac{A_3}{n}} \cdot \cdots \cdot e^{\frac{A_p}{n}}\right)^n \tag{3.1}$$

On applying Equation 3.1 to the time evolution of the QAA Hamiltonian, we obtain,

$$\begin{aligned} e^{-i\hat{H}_{QAA}(t)} &= e^{-i\left(\hat{H}_C\left(\frac{t}{T}\right)+\hat{H}_B\left(1-\frac{t}{T}\right)\right)} \\ &= \lim_{p\to\infty}\left(e^{\frac{-i\hat{H}_C\left(\frac{t}{T}\right)}{p}} \cdot e^{\frac{-i\hat{H}_B\left(1-\frac{t}{T}\right)}{p}}\right)^p \end{aligned} \tag{3.2}$$

The genius of Farhi et al. was in replacing the factor $\frac{1}{p}$ occurring in both the exponents of $e^{\frac{-i\hat{H}_C\left(\frac{t}{T}\right)}{p}}$ and $e^{\frac{-i\hat{H}_B\left(1-\frac{t}{T}\right)}{p}}$ with two mutually independent sets of parameters $\{\gamma_i\}$ and $\{\beta_i\}$ respectively, where i stands for the i^{th} repetition of the $\left(e^{\frac{-i\hat{H}_C\left(\frac{t}{T}\right)}{p}} \cdot e^{\frac{-i\hat{H}_B\left(1-\frac{t}{T}\right)}{p}}\right)$ block. This is also the step that makes QAOA a heuristic algorithm, unlike QAA, as there are no standardised proofs yet on how well QAOA works. With this step, they replaced the original QAA evolution with this modified evolution unitary U_{QAOA} for a finite p such that,

$$\begin{aligned} U_{\text{QAOA}} &= \left(e^{-i\cdot H_C\cdot\gamma_1} \times e^{-i\cdot H_B\cdot\beta_1}\right) \times \left(e^{-i\cdot H_C\cdot\gamma_2} \times e^{-i\cdot H_B\cdot\beta_2}\right), \\ &\times \left(e^{-i\cdot H_C\cdot\gamma_3} \times e^{-i\cdot H_B\cdot\beta_3}\right) \times \cdots \times \left(e^{-i\cdot H_C\cdot\gamma_p} \times e^{-i\cdot H_B\cdot\beta_p}\right). \end{aligned} \tag{3.3}$$

Furthermore, in the infinite limit, when $p \to \infty$, both QAA evolution, and QAOA evolution converge. We now divide each block into two commuting unitaries $U_C(\gamma)$ and $U_B(\beta)$ such that,

$$\begin{aligned} U_C(\gamma_i) &= e^{-i \cdot H_C \cdot \gamma_i} \\ U_B(\beta_i) &= e^{-i \cdot H_B \cdot \beta_i}. \end{aligned} \tag{3.4}$$

Please turn to chapter 5, section 5.2 for more details on $U_C(\gamma_i)$ and $U_B(\beta_i)$. Substituting Equation 3.4 into Equation 3.3, we get,

$$U_{\text{QAOA}} = [\, U_B(\beta_p) \cdot U_C(\gamma_p) \,] \times \cdots \times [\, U_B(\beta_2) \cdot U_C(\gamma_2) \,] \times [\, U_B(\beta_1) \cdot U_C(\gamma_1) \,]. \tag{3.5}$$

The order is reversed when represented as a Unitary, because $[\, U_B(\beta_1) \cdot U_C(\gamma_1) \,]$ gets applied first. Now, the full QAOA evolution unitary U_{QAOA} presented in Equation 3.3 is applied onto the n-qubit uniform superposition state $|+\rangle^{\otimes n}$, where,

$$|+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \left(\sum_{z \in \{0,1\}^n} |z\rangle \right),$$

to obtain a parametrized state $|\gamma, \beta\rangle$. Here, $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_p\}$ and β stands for $\{\beta_1, \beta_2, \ldots, \beta_p\}$. Finally,

$$|\gamma, \beta\rangle = [\, U_B(\beta_p) \cdot U_C(\gamma_p) \,] \times \cdots \times [\, U_B(\beta_1) \cdot U_C(\gamma_1) \,] \cdot |+\rangle^{\otimes n}. \tag{3.6}$$

The number of $[\, U_B(\beta_i) \cdot U_C(\gamma_i) \,]$ blocks applied is defined as the circuit depth p. Now, the energy expectation value of this $|\gamma, \beta\rangle$ with respect to the cost Hamiltonian H_C, is defined as the cost function $C(\gamma, \beta, H_C)$. That is,

$$C(\gamma, \beta, H_C) = \langle\, \beta, \gamma \mid H_C \mid \gamma, \beta \,\rangle. \tag{3.7}$$

Next, we use the parameters $\gamma_{opt}, \beta_{opt}$, corresponding to the global minima of the cost function $C(\gamma, \beta, H_C)$, to create a $|\gamma_{opt}, \beta_{opt}\rangle$ state. The idea is that after having enough circuit depth p, the population of the solution state $|gs\rangle$ we are interested in, increases in the full $|\gamma_{opt}, \beta_{opt}\rangle$ state, and hence can be obtained by sampling this final state repeatedly.

Throughout this discussion and in this thesis, we have not focused much on how the problem instances have been mapped onto the cost Hamiltonian H_C. The key point to note here is that we map the optimisation problems we are interested in solving onto the Ising Hamiltonian described below,

$$H_C = \sum_{i=1}^{n} h_i \hat{\sigma}_i^z + \sum_{j<i}^{n} \sum_{i}^{n} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z, \tag{3.8}$$

where $\hat{\sigma}_i^z$ is the Pauli Z operator. This is made possible because of the work of Lucas et al. [36] which describes ways to map NP hard problems onto this Ising Hamiltonian. For more discussions on the nature of the Hamiltonian, and gates used, please turn to chapter 5, section 5.2. We also discuss how the different instances used in this were generated, in chapter 4.

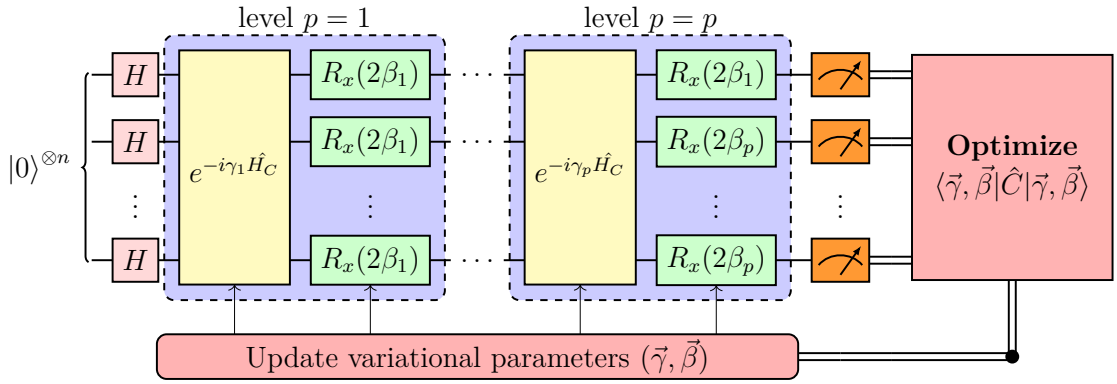


Figure 3.1: Full Schematic of QAOA with circuit depth p. $R_x(2\beta_i)$ here stems from a decomposed $U_B(\beta_i)$ (see subsection 5.2.1).

3.3 QAOA and Tensor Networks

From Equation 3.8, we can see that the term representing two-body interactions, J_{ij}, is not restricted to nearest neighbour entities, and assumes an all to all connectivity among the qubits. Such an architecture can be implemented on a linear chain of qubit register, such as a matrix product state using SWAP gates, and this is explained in great detail in subsubsection 5.2.2.2. Hence, in this section, we focus on the implications of this all to all connectivity, from a tensor network perspective.

As already stated in the introductory paragraph of this chapter there exists proofs on how difficult it is to classically sample from the output of a QAOA circuit, that is from a $|\gamma,\beta\rangle$ state even for the smallest circuit depth $p = 1$. From the discussions on matrix product states and bond-dimensions, in chapter 2, subsection 2.6.2, and section 2.8, we already know that the highest possible bond-dimension grows exponentially with system size n (Equation 2.10), as a result of un-approximated applications of all to all coupled entangling gates. This means that for exactly representing such highly-entangled states, one would require an exponentially increasing number of parameters, which quickly makes it impossible for classical hardware to keep up. Thus, the supremacy result of QAOA [31] could be understood straightforwardly from a tensor network perspective.

However, in QIOA, to keep the parametrised quantum states $|\gamma,\beta\rangle$ classically manageable, we introduce approximations by introducing a maximum limit of $D_{\max}$ on the bond dimensions throughout the evolution, by only keeping the largest $D_{\max}$ singular values. The rest of the Schmidt coefficients are discarded. This is a highly non-unitary and irreversible operation, after which one foregoes all hopes of exactly sampling from the original full $|\gamma,\beta\rangle$ state. All we are doing with QIOA is extracting the basis state $|s\rangle$ that holds the highest weight in a truncated QAOA state $|\gamma,\beta\rangle^{D_{\max}}$. We then compare to see if this state $|s\rangle$ is indeed the ground state $|gs\rangle$ of the problem we are interested in solving. However, this implementation gives no assurances on exactly representing what this population is in the original $|\gamma,\beta\rangle$ state, and hence does not violate the supremacy result from Farhi et al.

Part II

Methods

4

Instance Generation

From chapter 3, we know that the type of problems QAOA tries to handle are optimization problems that are hard for today's classical hardware even using the best classical algorithms. The field of computer science groups different problems into complexity classes based on how hard they are to tackle. Problems that are easy for classical computers to solve are all grouped under the class polynomial time, also called P, as the time it takes to solve this problem only scales polynomially with the problem size. Those problems that are expected to be beyond the capabilities of classical hardware now and in the future, are grouped under class NP, or non-deterministic polynomial time algorithms. Although it is still an open question if P is the same as NP, that is, if the problems in NP are hard only because we are yet to come up with an efficient algorithm to solve that problem, the most popular opinion is that these classes are different. More details on complexity classes and their properties can be found at this great textbook on the topic by Dr Boaz Barak and Dr Sanjeev Arora [11].

Furthermore, in complexity theory, decision problems are those problems that can be expressed as a "yes" or "no" question based on the inputs provided. For example, the question if a natural number provided as input is prime or not is a decision problem. Another class of problems that are closely related to decision problems, and are quite relevant in the field are optimization problems. Unlike decision problems where the output answer is a binary yes or no, these problems are committed to finding the best possible solution to a given problem. The fabled travelling salesperson problem [11] is an optimization problem example: given a set of cities, and their pairwise inter-city distances, what is the shortest routes covering all cities avoiding any repetition. In this thesis, we look at two different types of optimization problems, namely the Exact-Cover problem, and the Max-Cut problem.

4.1 Exact Cover

Mathematically, a cover of a given set X is a collection of sets S whose union contains X as a subset. An Exact Cover is a variation of a cover, where the elements of the total collection S are subsets of X, and there exists a sub-collection $S^{\dagger}$ of S such that there are no common elements among the members of $S^{\dagger}$, and the union of elements in $S^{\dagger}$ covers the full set X. The Exact Cover decision problem is one which queries given the set X and the collection of subsets of X, $\equiv S$, if there exists an

exact cover for X. For instance, given the set $X = \{2, 3, 5, 6\}$, and the collection $S = \{A, B, C, D\}$ such that $A = \{\}$, $B = \{2, 5\}$, $C = \{3, 5\}$ and $D = \{3, 6\}$. The sub-collection $S_1^\dagger = \{B, D\}$ is an exact cover of X because they have no elements in common, and their union covers the entire set X. Sub-collection $S_2^\dagger = \{A, B, D\}$ is also an exact cover of X as adding the null set A, does not change the above picture. However, the sub-collection $S_3^\dagger = \{B, C\}$ is not an exact cover as the sets B and C have a non-null intersection of $\{5\}$. Furthermore, elements of $S_3^\dagger$ also does not span the full set X. The optimization version of the exact cover problem is where given the set X and the collection S, one has to find the smallest sub-collection $S^\dagger$ that exactly covers X.

When we started working on this thesis, colleagues at the Applied Quantum Physics lab were already working on using QAOA to help solve the airline routing problem for Jeppesen [37], an airline route optimization company involved in route optimization. They are interested in solving the tail assignment problem [38] where the markers on the tail of each aeroplane, and hence that aeroplane is assigned to multiple routes. This was done by mapping the tail assignment problem onto exact cover instances. To make this problem accessible for today's NISQ era [1], they had taken real-world problems and truncated them to fit small enough quantum computers of sizes $8, 15$, and 25 qubits. While details on how they performed this mapping can be found here [39], for the purpose of this thesis, we treat Jeppesen as a black box providing real-world optimization problems already mapped onto Ising Spin Glass Hamiltonians.

4.2 Max-Cut

While tackling real-world problems using QIOA, the main algorithm proposed by this thesis, is exciting and relevant as is, to compare between QIOA and QAOA we wanted to also look at QIOA's performance on more standardised problems such as the Max-Cut instances. We choose Max Cut for this because works on benchmarking QAOA using max cut [40] and studies on the performance of QAOA on max cut [41] already exists in the literature. In this thesis, we explore using QIOA to solve randomly generated Erdős–Rényi graphs [42]. With the next two subsections, we shall go over the Max-Cut problem definition, and the family of Erdős–Rényi graphs.

4.2.1 Max-Cut Definition

In the field of graph theory, a cut is any partition of the full set of vertices of a graph into two subsets which fully covers the set of vertices, that is, have no elements in common, and fully spans the original set. A maximum cut of any graph is then any cut of that graph where each of the partition is at least the size of all other cuts in the partition. Simply put, this means partitioning the full set of vertices V, into two complementary subsets S and T such that the number of edges between the two sub-sets is as large as possible. The problem of finding the maximum cut of a given graph is hence rightly named, the Max-Cut problem. A more general version of this problem, where each edge is associated with a weight, is known as

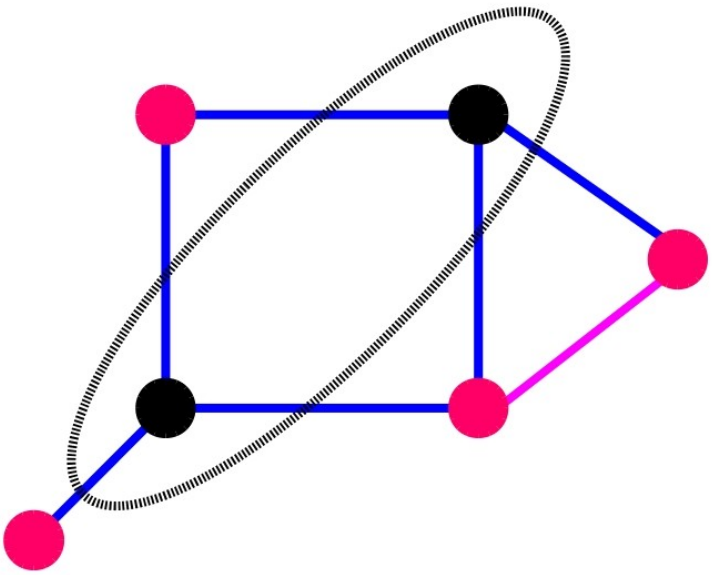


Figure 4.1: Example of a maximum cut (the dashed elliptic line) on a graph with 6 nodes and 7 edges. The split edges are highlighted in blue.

the weighted max-cut problem. Here, the objective is to maximise the weight of the edges between the two bipartitions. In this thesis, however, we are only concerning ourselves with the regular max-cut problem where all edge weights = 1.

Figure 4.1 depicts a max-cut instance solved pictorially. While creating a bipartition of a graph S and T, each of the nodes in the graph can be assigned to either S or T. Hence, we can map any partition of the graph of n nodes onto an n bit string accounting for each of the 2^{n-1} bipartitions, by assigning either $'0'$ or $'1'$ for a node in set S and the complementary bit for a node in set T. However, the total number of partitions = 2^{n-1} and not 2^n because of the bit symmetry in the system as any bit-string and its bit-flipped version points to the same partition. The maximum cut, which is also a partition is hence a member of the total 2^{n-1} possible partitions. Here, it is interesting and important to re-iterate that the assignment of $'0'$ or $'1'$ to either of the sets is completely arbitrary, and should not affect if a cut is maximum or not. Hence, any bit-string s and its bit-flipped string s^{-1}, both point to the same partition. Now while mapping the max-cut problem onto a Hamiltonian, if we encode how good a cut is, by assigning it an energy, then all states, including the ground state will not only be doubly degenerate but also entangled. Degenerate because both s and s^{-1} point to the same partition, and entangled because any superposition state $s + s^{-1}$ is a maximally entangled state.

4.2.2 Erdős - Rényi Graphs

Erdős–Rényi graphs are one of the most popular class of random graphs where there exists a fixed probability for the existence or non-existence of an edge between any two nodes in the graph. In this thesis, we have been dealing with the $G(n, p)$ model

of Erdős–Rényi graphs [42] where n stands for the number of nodes and p is the fixed probability on if an edge exists or not. For our work, we have chosen $p = 0.5$, where it is equally likely for an edge to either exist or not exist.

Mathematically, one of the most straightforward ways of representing a graph is using an Adjacency Matrix [43]. The adjacency matrix A of a graph G with n nodes, is an $n \times n$ matrix where each element a_{ij} of the i^{th} row and j^{th} column of A represents the weight of the edge between nodes i and j. If $a_{ij} = 0$, then there is no edge between the nodes i and j. In this framework, it is easy to see that all the diagonal elements $a_{ii} = 0$, because an edge needs at least two nodes to exist. Furthermore, since we are interested in non-weighted graphs, all non-zero elements $a_{ij} = 1$. So one easy way to create random Erdős–Rényi graphs is to create random adjacency matrices. For this, we first create a random matrix R where each element r_{ij} is a uniformly sampled value between 0 and 1. Next, when p is the probability of an edge existing or not, we set all elements $r_{ij} < p$ to 0, and all elements $r_{ij} \geq p$ to 1. Also, since we have a non-directed graph where it does not matter if node i is connected to node j or vice versa, the adjacency matrix needs to be symmetric, that is, $a_{ij} = a_{ji}$. Hence, we only choose the upper triangular part of our random matrix R and set the lower half to zero. Finally, we set all diagonal elements also to zero, and we have a randomly generated Erdős-Rényi Graph. Figure 4.2 is an example of an Erdős–Rényi graph with 10 nodes created using the method described here.

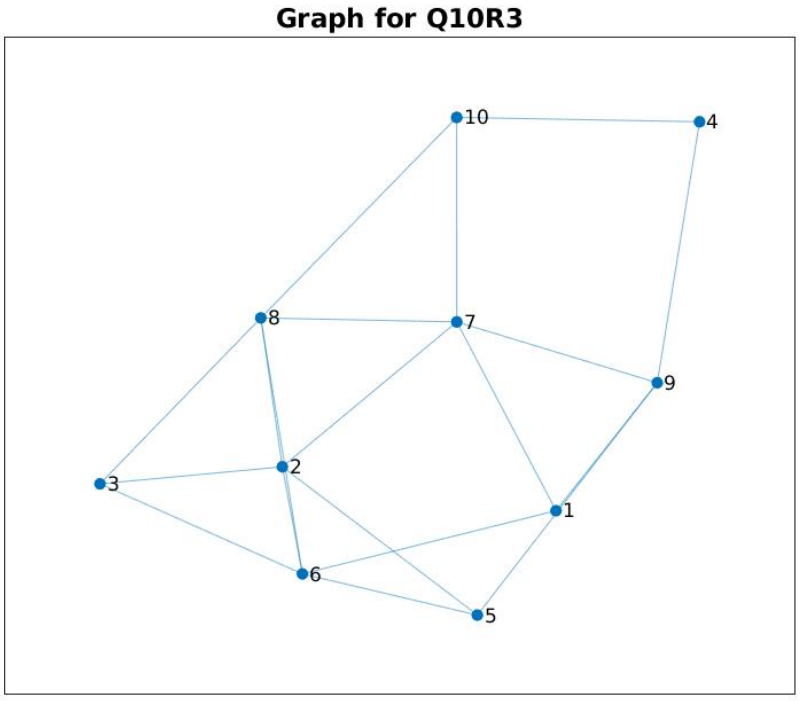


Figure 4.2: An example of an Erdős–Rényi graph with 10 nodes

4.3 Nomenclature

Before moving onto describing the state-preparation, training of QAOA/QIOA and the actual QIOA algorithm methods, it may be beneficial to briefly go over the nomenclature used to refer to problem instances in this thesis. We started our work by modelling the original QAOA algorithm using Tensor Networks to try and solve the Exact Cover problem described above. For this purpose, we looked at ten 8-qubit instances, 9 15-qubit instances, ten 25 qubit instances and one 50-qubit instance. Furthermore, we also looked at ten each of 8, 9, 10, 11, and 12-qubit instances of the max cut problems on randomly generated Erdős–Rényi graphs. In the remainder of this text, we will be referring to these instances using the code name **QnPi** where Qn stands for n number of qubits, $'P'$ is to signify that these instances are obtained from Jeppesen, and i, refers to the index ranging from 0 to 9. Similarly, instances of the max cut problems will be referred to as **QnRi** where $'R'$ is used to point that these are randomly generated instances, and rest remains the same. Hence, for example, $Q10R3$, which is, in fact, the instance shown in Figure 4.2, is the $10-$qubit max cut instance number 3, whereas $Q15P8$ is the $15-$qubit exact cover instance number 8 obtained from Jeppesen.

5

State Preparation

This chapter talks about the methodology implemented in preparing a parametrized state $|\gamma, \beta\rangle$. We first describe the creation of the initial state as an MPS, describe the single and two-qubit gates used as MPOs, and finally put them all together to implement the QAOA algorithm.

5.1 Initialization

We start off with the n qubit register all in the uniform superposition state, where all the 2^n basis states of an n qubit Hilbert space is in a uniform superposition each with weight equal to $1/2^n$. This state is also called the plus state and it is an eigenvector of mixer Hamiltonian U_B used in QAOA. The Plus state is also a product state with no entanglement as it is just a single tensor product state. While there exist multiple implementations of modified QAOA where different mixers and starting points are considered [33], we do not consider them in this study. As described in chapter 2, a product state with no entanglement are trivially connected by bonds of dimension of 1 throughout, and all the n qubit tensors are of dimension $1 \times 2 \times 1$. Figure 5.1 below represents schematically what the MPS looks like.

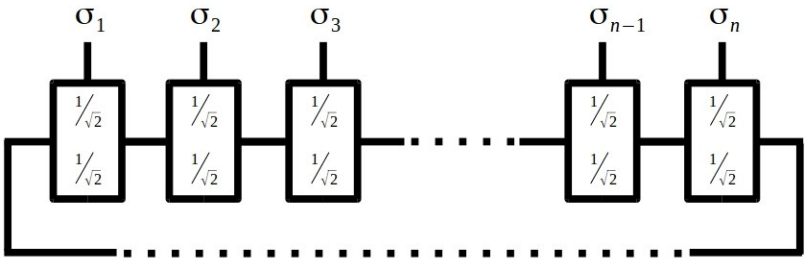


Figure 5.1: n Qubit MPS register in the uniform superposition state

5.2 Gates Used

This section introduces the different single and two-qubit parametrized and non-parametrized gates that are used for simulating QAOA. As described in chapter 2, the different gates are treated as Matrix Product Operators in the Tensor Network framework.

5.2.1 Single Qubit Gates

This section describes all the single-qubit gates used in the circuit simulation.

5.2.1.1 R_x Gates

Again, in this implementation of QAOA, we use a mixer Hamiltonian which is just the single-qubit Pauli X gate ($\hat{\sigma}^x$) applied to all the n qubits. Hence, as detailed in chapter 3,

$$H_B = \sum_{i=1}^{n} \hat{\sigma}_i^x \tag{5.1}$$

$$\begin{aligned} U_B(\beta) &= e^{-i\beta H_B} \\ &= e^{-i\beta \sum_{i=1}^{n} \hat{\sigma}_i^x} \\ &= \prod_{i=1}^{n} e^{-i\beta\hat{\sigma}_i^x}. \end{aligned} \tag{5.2}$$

The unitary $e^{-i\beta\hat{\sigma}_i^x}$ is the single-qubit rotation ($R_x(2\beta)$) applied to qubit i around the X-axis by an angle 2β. Hence,

$$U_B(\beta) = (R_x(2\beta))^{\otimes n}. \tag{5.3}$$

In the Tensor Network notation, $R_x(2\beta)$ is a rank 2 tensor with two indices, one for input and one for output of dimensions 2 each

$$R_x(2\beta) = \begin{bmatrix} \cos(\beta) & -i\sin(\beta) \\ -i\sin(\beta) & \cos(\beta) \end{bmatrix} \equiv \quad -\boxed{R_x(2\beta)}-. \tag{5.4}$$

5.2.1.2 R_z Gates

Since the aim of this thesis is to solve optimisation problems, the Cost Hamiltonians we use is the Ising Spin Glass model. As a reminder, this is because finding the ground state of a general Ising Spin Glass is an NP-hard problem, and all problems within the class NP-hard is reducible to all other problem with only a polynomial overhead. What this means that if we knew the solution to a problem A that we know is in the class NP-hard, then we would be able to find the solution of another problem B in the same class by using the algorithm to solve A as a sub-routine. Further, reference [36] gives detailed information on how to find the Ising formulations of many Np-hard problems. Thus the cost function has the form as given by Equation 5.5,

$$H_C = \sum_{i=1}^{n} h_i\hat{\sigma}_i^z + \sum_{i=1}^{n}\sum_{i\neq j\ j=1}^{n} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z. \tag{5.5}$$

$$
\begin{aligned}
U_C(\gamma) &= e^{-i\gamma H_C} \\
&= e^{-i\gamma \sum_{i=1}^{n} h_i \hat{\sigma}_i^z \; + \; \sum^n \sum^n{}_{i \neq j} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z} \\
&= e^{-i\gamma \sum_{i=1}^{n} h_i \hat{\sigma}_i^z} \;\times\; e^{-i\gamma \sum^n \sum_{i \neq j}^{n} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z} \\
&= \prod_{i=1}^{n} e^{-i\gamma h_i \hat{\sigma}_i^z} \times \prod_{i \neq j}^{n} e^{-i\gamma J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z} \\
&= \prod_{i=1}^{n} U_{C1}^{i}(\gamma, h_i) \;\times\; \prod_{i \neq j}^{n} U_{C2}^{ij}(\gamma, J_{ij}) \\
&= U_{C1}(\gamma) \;\times\; U_{C2}(\gamma)
\end{aligned}
\tag{5.6}
$$

Hence we see that the unitary $U_C(\gamma)$ (defined in Equation 3.3) is composed of two separate unitary operations $U_{C1}(\gamma)$ and $U_{C2}(\gamma)$. The two respective unitaries can be again taken to be tensor products of single and two-qubit gates $U_{C1}^{i}(\gamma, h_i)$ and $U_{C2}^{i,j}(\gamma, J_{ij})$ respectively

$$U_{C1}^{i}(\gamma) = e^{-i\gamma h_i \hat{\sigma}_i^z}, \tag{5.7}$$

$$U_{C2}^{i,j}(\gamma, J_{ij}) = e^{-i\gamma J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z}. \tag{5.8}$$

Here, $U_{C1}^{i}(\gamma, h_i)$ is the single-qubit Z rotation ($R_z(2\gamma h_i)$) applied to qubit i around the z-axis by an angle $2\gamma h_i$. On the other hand, $U_{C2}^{i,j}(\gamma, J_{ij})$ is the two-qubit gate that is responsible for creating entanglement within the system, and is described in the next section. Hence,

$$U_{C1}(\gamma) \;=\; R_z(2\gamma h_1) \otimes R_z(2\gamma h_2) \otimes .... \otimes R_z(2\gamma h_n). \tag{5.9}$$

Again, in the Tensor Network notation, $R_z(2\gamma h_i)$ is a rank 2 tensor with input and out index each of dimension 2

$$R_z(2\gamma h_i) \;=\; \begin{bmatrix} e^{-i\gamma h_i} & 0 \\ 0 & e^{+i\gamma h_i} \end{bmatrix} \equiv \quad -\boxed{R_z(2\gamma h_i)}- \tag{5.10}$$

5.2.2 Two Qubit Gates

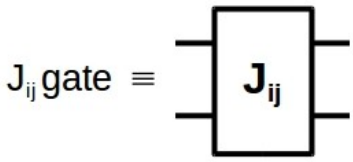
We now move on to describing the two-qubit gates used in the circuit simulation. See chapter 2, section 2.8, for a detailed description of how two-qubit gates can be applied to an MPS while keeping the MPS-form.

5.2.2.1 J_{ij} Gates

Continuing from subsubsection 5.2.1.2, it was noted that gates $U_{C2}^{i,j}(\gamma, J_{ij})$ are the two-qubit gates that are responsible for creating entanglement, or correlations in the circuit system. After a bit of algebra, it can be shown that:

$$
\begin{aligned}
U_{C2}^{i,j}(\gamma, J_{ij}) &= e^{-i\gamma J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z} \\
&= \cos(\gamma, J_{ij}) \times (I \otimes I) \; - \; i\sin(\gamma, J_{ij}) \times (\hat{\sigma}_i^z \otimes \hat{\sigma}_j^z) \\
&= \begin{bmatrix} e^{-i\gamma J_{ij}} & 0 & 0 & 0 \\ 0 & e^{+i\gamma J_{ij}} & 0 & 0 \\ 0 & 0 & e^{+i\gamma J_{ij}} & 0 \\ 0 & 0 & 0 & e^{-i\gamma J_{ij}} \end{bmatrix}.
\end{aligned} \tag{5.11}
$$

From now on, we refer to the two-qubit gate $U_{C2}^{i,j}(\gamma, J_{ij})$ as simply the J_{ij} gate for ease of reference. Equation 5.11 however is in the vector-matrix notation where an n qubit register is taken to be a single $2^n \times 1$ vector, a single-qubit gate is a 2×2 matrix, and a two-qubit gate is a 4×4 matrix. To convert our J_{ij} gate into tensor network notation, we reshape the 4×4 matrix into a $2 \times 2 \times 2 \times 2$ tensor of rank 4; with 2 input and output indices each.



5.2.2.2 SWAP Gates

Recalling from chapter 2, the Matrix Product State is also aptly called as a Tensor Train, as the different tensors representing the qubit registers are connected only linearly like the compartments in a train. This means that one qubit is only connected to its nearest neighbours in the MPS architecture, while the cost function of QAOA calls for an all to all connected framework in the most general case. Thus to implement non-local two-qubit gates in a linear architecture, we make use of the swap gates. While the no-cloning theorem prohibits making copies of qubits, we can still use swap gates to transfer quantum information from one qubit to another. In the matrix notation, the SWAP gate is given as:

$$
SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.12}
$$

As before, to convert this into a Matrix Product Operator, we reshape this 4×4 matrix into a $2 \times 2 \times 2 \times 2$ tensor, with 2 input and output indices each of dimension 2. We have tried two different SWAP networks that result in a net all-to-all connectivity.

SWAP Network I: In this network, to apply a two-qubit gate between qubits i and j, where we can take i < j without loss of generality, we first apply multiple

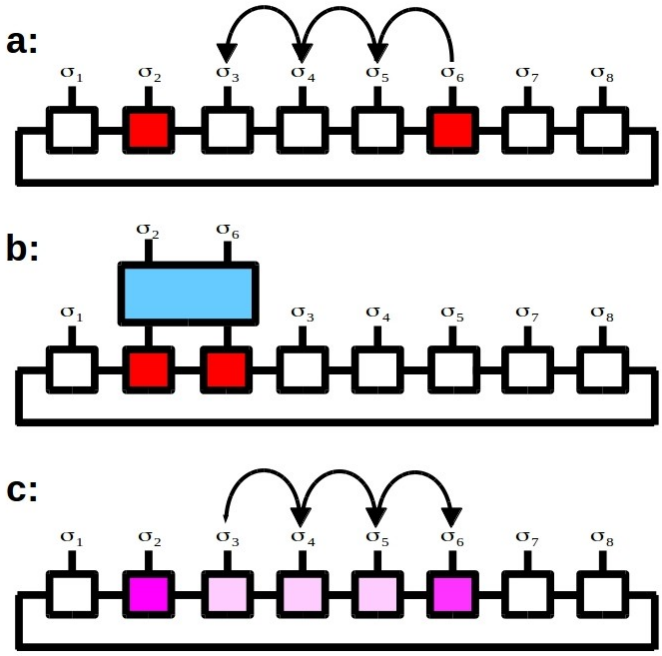


Figure 5.2: Schematic showing an example implementation of one non-local J_{26} gate using SWAP Network I in an 8 qubit register. a: represents applying 3 SWAP gates to bring qubit 6 to position 3. b: depicts the application of gate J_{26} and c: represents swapping qubit 6 back to position 6.

SWAP gates ($j-i-1$ SWAP gates to be exact), to bring qubit j to position i+1. Now the information previously contained in qubits i and j are now contained in qubits i and i+1 making any subsequent interaction between them local in this new configuration. Now we can apply our two-qubit gate J_{ij} and SWAP the qubits back to their original state by applying the ($j-i-1$) SWAP gates in reverse order. This entire procedure is depicted schematically in Figure 5.2. Calculating the number of SWAP gates in this implementation, we see that we need $\hat{O}(n^3)$ SWAP gates where n is the number of qubits. This is proved below. The basic idea is that there are $(n-1)$ nearest neighbour two-qubit gates, $(n-2)$ next-nearest neighbour gates needing 2 SWAP operations and so on. This mathematically translates to:

$$\begin{aligned}
\#\text{ of SWAP gates } \equiv\ S\ &=\ 2\times\sum_{i=1}^{n}(n-i)(i-1)\\
&=\ 2\times\left[(n+1)\times\sum_{i=1}^{n}i-n\times\sum_{i=1}^{n}1-\sum_{i=1}^{n}i^2\right]\\
&=\ n(n+1)^2\ -\ 2n^2\ -\ \frac{n(n+1)(2n+1)}{3}\\
&=\ \frac{n^3-3n^2+3n-n}{3}\\
&\equiv\ \hat{O}(n^3).
\end{aligned}$$

Figure 5.3: Schematic showing an example implementation of SWAP Network II in a 5 qubit register. Each Layer facilitates interaction between different pairs of qubits. Figure from reference [5]

SWAP Network II: In the previous implementation, it is maybe possible to intuitively understand that swapping each state back and forth is an overkill. The question now is if one can make use of the intermediate SWAPS to apply two-qubit gates, and if so, how? One way to formalise and implement this intuition was found in a paper from the Aspuru-Guzik lab [5] where alternating SWAPS between odd and even pairs of qubits were employed multiple times to bring about different configurations where each qubit pair (i,j) existed as nearest neighbours in at least one of the configurations. This is pictorially depicted in Figure 5.3 in the case of a 5 qubit system. Close observation reveals that all pairs of interactions are satisfied in the set of all layers. For instance, layer 3 in the figure facilitates interaction between qubits 2 & 4 and 1 & 5. Furthermore, we see that the information in the qubit register gets reversed after the application of n layers. And seeing that we need approximately $n/2$ SWAP gates in each layer, we can conclude that we only need $\hat{O}(n^2)$ number of SWAP operations in this network.

It is worth pointing out that the need for swapping operation is not just a quirk of using matrix product states in our circuit simulation, but is physically the methods used to apply non-local gates in qubit circuit architectures with only nearest neighbour coupling among qubits. While finding efficient mappings of the given problem on to what is physically feasible on the circuit connectivity at hand, is an entire research field in itself, the main idea is to limit the number of SWAP operations as it not only adds to the time complexity of the algorithm leading to needing qubits with higher coherence times but also leads to loss of information resulting from the lower fidelities of SWAP gates currently. Furthermore, since one of the main objec-

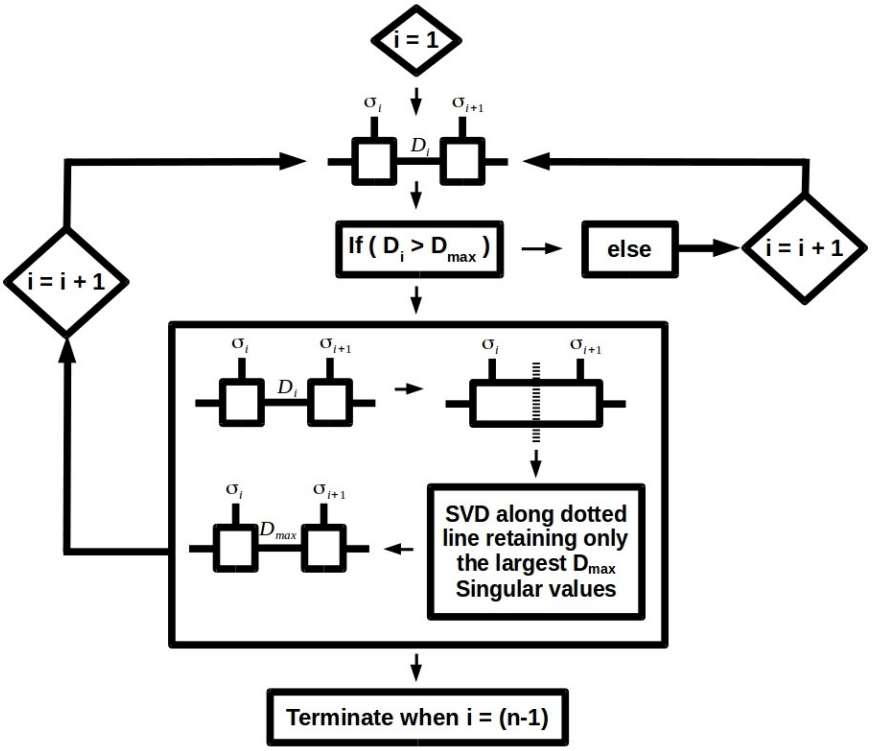


Figure 5.4: Flowchart describing the truncation operation

tives of this thesis is to find an efficient simulation of Variational Circuits, we shall proceed with SWAP Network II in this thesis. We also tried out both methods, and method II was significantly faster for larger systems as expected.

5.3 Truncation

While we can run exact simulations of Quantum Circuits using the toolsets we have already developed, this soon becomes impractical due to an exponential increase of quantum entanglement with the system size as a consequence of the all-to-all connectivity paradigm we are working on. As detailed in chapter 2, subsection 2.6.2, this increases the bond dimension of the matrix product state exponentially, leading to an exponential rise in the number of parameters needed to describe the state, making it intractable.

We can control the growing entanglement by setting an upper bound = $D_{\max}$ to the bond dimension, thus limiting the number of Schmidt coefficients and the size of the tensors representing the qubits. While doing this helps make the system tractable, the question now is if we can still extract useful information from this approximate simulation of the Quantum Circuits. Before going to answer that in the results in chapter 8 and chapter 7, we go over the truncation procedure here.

Single-qubit gates do not influence the amount of entanglement in the quantum register. Hence we do not need to apply truncation after their operation. It is the two-qubit gates that introduce entanglement in the system. Hence, one good way to limit entanglement would be to apply Schmidt decomposition partitioning the two affected qubits into separate bipartitions and limit the bond dimension to an upper bound $D_{\max}$ after the application of any two-qubit gate. This, however, doesn't look to be a correct strategy as this would call for a truncation even after application of SWAP gates. SWAP gates are not part of the actual problem definition, and hence truncations after SWAP would not only increase the time complexity of the algorithm significantly but also result in spurious loss of information. Thus the method employed in this thesis is to sweep through the $n-1$ bond dimensions in the matrix product state after the application of all J_{ij} gates associated with a particular layer as described in Figure 5.3. This entire procedure is diagrammatically depicted using a flowchart in Figure 5.4.

5.4 Cost Estimation

With section 5.3, we have all the required tools to prepare an arbitrary $|\gamma,\beta\rangle$ state. Calculating the expectation value of the cost Hamiltonian $\equiv C(\gamma,\beta) = \langle\beta,\gamma|\hat{H}_C|\gamma,\beta\rangle$ (Equation 3.7) is the next step

$$\begin{aligned} C(\gamma,\beta) &= \langle\beta,\gamma|\ \hat{H}_C\ |\gamma,\beta\rangle \\ &= \langle\beta,\gamma|\ \sum_{i=1}^{n} h_i\hat{\sigma}_i^z\ +\ \sum_{i=1}^{n}\sum_{i\neq j\ j=1}^{n} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z\ |\gamma,\beta\rangle \\ &= \sum_{i=1}^{n} h_i\ \langle\beta,\gamma|\ \hat{\sigma}_i^z\ |\gamma,\beta\rangle\ +\ \sum_{i=1}^{n}\sum_{i\neq j\ j=1}^{n} J_{ij}\ \langle\beta,\gamma|\ \hat{\sigma}_i^z\hat{\sigma}_j^z\ |\gamma,\beta\rangle \\ &= \sum_{i=1}^{n} h_i \times \text{Expt}(\ \hat{\sigma}_i^z\ )\ +\ \sum_{i=1}^{n}\sum_{i\neq j\ j=1}^{n} J_{ij} \times \text{Expt}(\ \hat{\sigma}_i^z\hat{\sigma}_j^z\ ). \end{aligned} \tag{5.13}$$

The $\text{Expt}(\hat{O})$ operation in Equation 5.13 refers to the expectation function which is defined as $\text{Expt}(\hat{O}) = \langle s|\ \hat{O}\ |s\rangle$ where $\hat{O}$ is the operator and $|s\rangle$ is a quantum state. Hence, calculating the expectation of the entire cost Hamiltonian has been broken down into calculating the expectation values of single and two-qubit operators $\hat{\sigma}_i^z$ and $\hat{\sigma}_i^z\hat{\sigma}_j^z$ respectively. While this may seem trivial at first glance, this is of great significance from a tensor network perspective. This is because the entire Hamiltonian of an n qubit system is a rank $2n$ matrix product operator of dimension $2\times2\times2\times.....\times2$, $2n$ times, needing 2^{2n} parameters. The size of the full Hamiltonian hence scales exponentially, and thus simulating more than 15 qubits would be a hassle on most laptops. On the other hand, as iterated in subsection 5.2.1 and subsection 5.2.2, the tensor representation of these single and two-qubit gates only need 4 and 16 parameters respectively. The number of these operations also only scales polynomially with system size n making the entire operation of cost calculation highly efficient on even most modern laptops. Figure 5.5 pictorially depicts

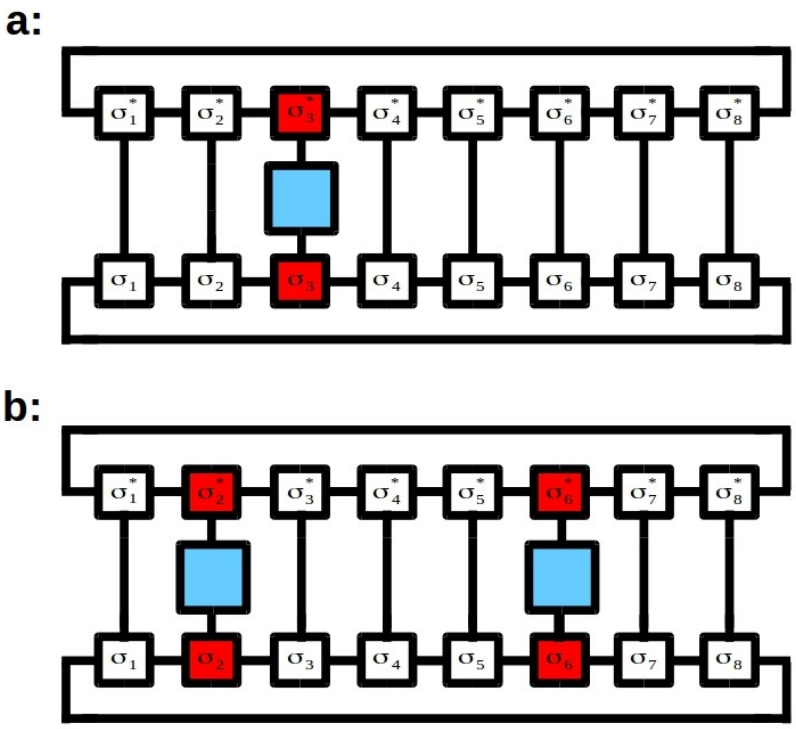


Figure 5.5: Pictorial depiction of calculating single and two-qubit expectations using matrix product states on an 8 qubit register. a: Expectation of single-qubit $\hat{\sigma}_3^z$. b: Expectation of two-qubit $\hat{\sigma}_2^z\hat{\sigma}_6^z$.

how one could calculate the single and two-qubit expectation values mentioned in Equation 5.13 using tensor networks.

5.5 Summary

With this, we now are equipped with efficient ways to

- Initialise the qubit register: **section 5.1**
- Apply single-qubit gates: **subsection 5.2.1**
- Apply local and non-local two-qubit gates effectively: **subsection 5.2.2**
- Truncate the $|\gamma, \beta\rangle$ state while retaining maximum information: **section 5.3**
- Calculate the expectation value of the full Cost Hamiltonian: **section 5.4**

and hence have set the stage for all further explorations. Figure 5.6 depicts how the entire QAOA state preparation circuit looks like, minus the truncation and contraction, for a 4 qubit system, for $p = 1$. We see again that after the application of all the gates, the qubit information order is reversed as noted in subsubsection 5.2.2.2. This particular block can be repeatedly applied p number of times to increase the circuit depth to p.

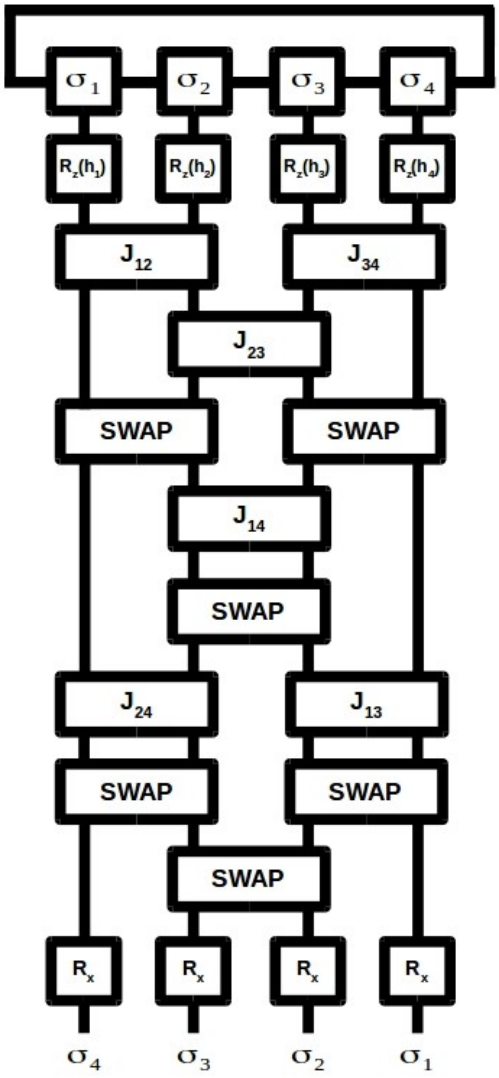


Figure 5.6: The actual Tensor Network circuit for preparing a generic 4 qubit QAOA state for $p = 1$ without the truncation and contraction steps being depicted.

The next chapter (chapter 6) now talks about using all these tools to train QAOA, and the chapter after that (chapter 7) would be about sampling from the optimum $|\gamma, \beta\rangle$ state to obtain the correct solution to the optimisation problem at hand.

6

Training QAOA

QAOA is a variational algorithm where a classically-parametrized quantum state ansatz is moulded to fit the problem specifications by varying the parameters. This has a lot of parallels with Neural Network architectures, where the weights of a neural network are varied to fit a high dimensional objective function that, say, classifies between different entities. In both cases, this process of finding the right parameters for the ansatz is referred to as the training phase. However, in the case of neural networks, one trains the parameters in the network by optimising them over different instances of training data and minimising a defined loss function. Whereas, for variational quantum algorithms, we find the optimum parameters by directly minimising the energy of the quantum state with respect to the Cost Hamiltonian H_C without involving any "training" data. This energy minimization method is based on the variational principle in quantum mechanics which states that the ground state of a system has the least energy. Hence, minimising the energy of a system eventually results in the system occupying the ground state. This is also from where these variational algorithms get the first part of their name. The key difference between classical neural networks and variational quantum algorithms is that while both paradigms work by optimising a cost function, neural networks make use of a generic cost function that most often does not encode any information about the problem by itself (except in cases where additional constraints are used in the loss function) [44]. On the other hand, the cost Hamiltonian H_C used in variational algorithms completely captures all essential details of the problem of interest.

The integral component for any training procedure in QAOA is a cost function that accepts the cost Hamiltonian H_C and $2p$ angles $\gamma_1, \gamma_2, \gamma_3, ..., \gamma_p, \beta_1, \beta_2, \beta_3, ..., \beta_p \equiv \{\gamma, \beta\}$, to output a scalar that is the energy of the $|\gamma, \beta\rangle$ state with respect to the cost Hamiltonian. From now on, we refer to this function as $C(\gamma, H_C, \beta)$. Also, while referring to the cost landscape, we choose to omit H_C as its role is implicitly understood

$$\text{QAOA Cost Function} \equiv C(\gamma, H_C, \beta). \tag{6.1}$$

In this thesis, we create such a function using the tools developed in chapter 5. The goal of the training procedure is to calculate the global minima of this function. Finding the global minimum of a function while staying away from local minimas is in itself a non-trivial task. Nevertheless, since the value of $C(\gamma, \beta)$ corresponds to the energy of the system, local minimas still correspond to angles which produce a state

with lesser energy. Hence, one could still extract the solution, that is the ground state if the minimas are deep enough. In this thesis, we have explored primarily two methods of finding the minima of this function. The grid search method, and off-the-shelf functional optimization tools such as Nelder-Mead [45].

6.1 Grid Search

This is one of the most straightforward search algorithms to find the global minima of a smooth function. Grid searches work best for smooth functions because, in this method, we essentially discretize our function to get insights about its landscape. Hence, the method wouldn't work if the landscape we obtain is highly jagged and dependent on the grid sizes we use. In QAOA, we have some decent assurance that the function is smooth, at least for a fixed circuit depth p that is independent of system size n [2]. While there can be more sophisticated grid search implementations which integrate coarse-graining into the search, we are not going to be discussing that here. Those methods work by progressively choosing smaller and smaller search domains around their minimas, while retaining the same number of points, effectively zooming in on the minima. The original QAOA article [2] argues that for QAOA functions handling combinatorial optimization problems of m clauses and n bits, all the partial derivatives of $C(\gamma, \beta)$ are bounded by $\hat{O}(m^2 + mn)$. This implies that the steepness of the function extremas will be bounded and hence by choosing a threshold grid size, and lower sizes, one can efficiently find the minima. This is, of course, assuming that the circuit depth p does not scale with the system size n. As otherwise, one would have to search for a minima within an exponentially scaling space $(\gamma, \beta) \in [0, 2\pi]^p \times [0, \pi]^p$. Note that the range $\gamma_i \in [0, 2\pi]$ and $\beta_i \in [0, \pi]$ is valid when the Hamiltonian H_C has integer eigenvalues. Figure 6.1 illustrates an example of finding the minima using grid search when circuit depth $p = 1$.

6.2 Optimisation Protocols

While having figures as colourful as Figure 6.1 is a great way to visually study the behaviour of the QAOA function $C(\gamma, \beta)$, the grid search method becomes quickly cumbersome for $p > 1$. This is because for $p > 1$ one has to search in a high dimensional $2p$ space. Here, we also lose the advantage of visual inspection as there is no straightforward method to visually represent a $2p + 1$ dimensional figure. Furthermore, since we are only interested in a single point within this entire $[0, 2\pi]^{\otimes p} \times [0, \pi]^{\otimes p}$ space, it is extremely wasteful to evaluate the value of $C(\gamma, \beta)$ at all the other points also. Hence, for a larger p, we opt to make use of numerical-functional minimization methods such as Nelder-Mead [45]. In this study, we avoid using methods such as gradient descent, which is the workhorse of the machine learning community [44], as such optimization methods require one to calculate the gradients of the function $C(\gamma, \beta)$. This would possibly make the procedure more cumbersome and error-prone, as we now could have added errors from the numerical approximations of the gradient as well.

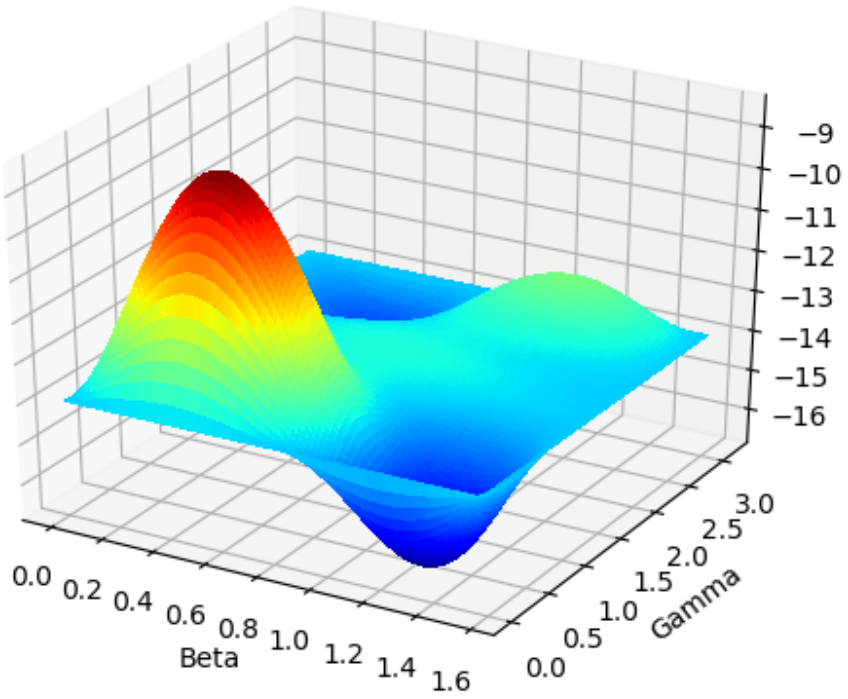


Figure 6.1: Example of the Energy landscape $C(\gamma, H_C, \beta)$ for a 11 qubit MaxCut instance for circuit depth $p = 1$. Number of grid points $= 100 \times 100$.

The chance of success, that is the probability of such optimisation protocols correctly identifying the global minima, is highly dependent on the starting point we initialise the protocol with. The further away this initial guess is from the actual global minima, the higher are the chances of getting stuck in a local minimum. However, although at first glance it may look like these methods have a very low success probability, in practice, they work remarkably well. Numerical studies by Lukin et. al. have also shown that the parameter angles γ_{p+1} and β_{p+1} are dependent on all the preceding angles $\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_p$ and $\beta_1, \beta_2, \beta_3, \ldots, \beta_p$ respectively [46]. Hence it is possible to extrapolate good initial guesses for finding the angles of circuit depth $p + 1$ from the parameters of circuit depth p.

In this thesis, we are using the optimal angle parameters calculated using a matrix-vector implementation of the same problems by a colleague on Matlab. As a result, do note that these parameters have been calculated using non-truncated $|\gamma, \beta\rangle$ states having the highest possible bond-dimension. That is, $D_{\max} = 2^{\lfloor n/2 \rfloor}$. For higher p, in both the tensor network and vector-matrix implementations, we treat the optimization protocols used as a black-box which outputs the angles when provided with the QAOA function $C(\gamma, \beta)$, and initial guesses from the extrapolation algorithm, obtained using [46]. In the next chapter, chapter 7, we go over the methodology used to extract the ground state from a truncated matrix product state, which is the essence of QIOA.

7
QIOA

This chapter talks about implementational details behind the Quantum Inspired Optimisation Algorithm, QIOA, that we are proposing with this thesis. Maybe, you the reader, have already noticed that the name QIOA is a play on the original QAOA [2] (please refer chapter 3 for more details). This is because QIOA is a classical quantum inspired algorithm that is inspired by QAOA. In vanilla QAOA, the cost function is mapped onto a highly entangling all-to-all connected circuit. From chapter 2, section 2.8, we know that such un-approximated applications of entangling gates between all the qubits leads to an exponential increase in the amount of entanglement among the qubits. It is this exponential increase that makes it infeasible for classical computers to keep track of the created quantum states. This is because the number of parameters required to represent such a highly entangled state increases exponentially with system size, making it infeasible for classical computers to keep up. The following lines describe the essence of QIOA.

It is already known that the original non-approximated vanilla-QAOA, (in which entanglement scales exponentially [31], section 3.3), facilitates one to obtain solutions to the problem of interest. The question that we are trying to answer with QIOA is as follows. Would it still be possible to extract useful information from a truncated implementation of QAOA, where the entanglement only scales polynomially with system size?

It turns out that for many problems, we can still extract the solution even from such truncated state implementations. A more detailed description of these results on training and on extracting the solution can be found at chapter 8 and chapter 9 respectively. While in all the tests we have done so far we have not been able to identify an example where QIOA fails, further studies are definitely required before making conclusive remarks on its applicability. After describing the general principle of QIOA in the next section, we move on to describe the implementation, first for product states, and then for states with non-zero but possibly limited entanglement.

7.1 General Principle

Superposition and entanglement are two quantum mechanical phenomena that we know are essential for having a quantum advantage. While it is still unclear if these are the only two necessary requirements, we know for sure that just with quantum

superposition, we cannot have any advantage. This can easily be proved by recalling from chapter 2, subsection 2.6.3 where we have shown that representing a non-entangled product state of an n qubit register requires only $2n$ complex parameters. One each for representing the coefficients of the $|0\rangle$ state and the $|1\rangle$ state. This representation only scales linearly with system size, making it easily tractable for any classical hardware. The absence of any entangling gates also ensures that the bond-dimension does not grow and stays at 1 during the state evolution, irrespective of the other gates applied. The same idea can also be generalised to account for states in which the entanglement does not scale exponentially with the system. Such systems can be described using a matrix product state whose component tensor-dimensions only grow polynomially [28]. Hence, we only need a polynomially scaling number of parameters to describe these systems. Thus, any algorithm making use of these ideas are classically tractable as they are polynomial scaling. However, since these algorithms may still be making use of quantum mechanical ideas such as the variational principle, entanglement, superposition etc., they would be classified under the category of quantum inspired classical algorithms.

QIOA is one such quantum inspired algorithm that takes a lion's share of its inspiration from vanilla-QAOA. The main idea behind increasing the circuit depth p in vanilla-QAOA is to increase the population of the solution ground state, $|gs\rangle$, in the full $|\gamma_{opt}, \beta_{opt}\rangle$. Here, γ_{opt} and β_{opt} are the $2p$ optimal parameters for the problem at hand. This is explicitly stated in the original QAOA paper [2]. This means that, given sufficient circuit depth p, the coefficient C_{gs} of the solution $|gs\rangle$ has the highest magnitude in the full $|\gamma_{opt}, \beta_{opt}\rangle$ state. Conversely, this implies that if we are able to extract the basis state $|s\rangle$ that has the maximum coefficient within the full $|\gamma_{opt}, \beta_{opt}\rangle$ state, then that has to be the ground state in vanilla QAOA. That is,

$$\begin{aligned} |\gamma, \beta\rangle \;&=\; C_{gs}|gs\rangle + \sum_{i \neq gs} C_i \; |i\rangle \\ (\gamma, \beta) \;=\; (\gamma_{opt}, \beta_{opt}) \;&\iff\; \|C_{gs}\| \;\geq\; \|C_i\| \;\; \forall \; |i\rangle. \end{aligned} \tag{7.1}$$

QIOA explores if this idea of the solution having the maximum magnitude still holds for a truncated state $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ also. Here, $D_{\max}$ refers to the defined upper bound on the maximum bond dimension in the matrix product state representation of the $|\gamma_{opt}, \beta_{opt}\rangle$ state. QIOA then extracts the basis state having the maximum magnitude from $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$, and verifies if that is indeed the ground state $|gs\rangle$. In all the problem instances we have simulated till now, this idea is valid and the extracted state $|s\rangle = |gs\rangle$.

The next section goes over how to implement the tools developed in chapter 5 to put together the state $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$. The section after that describes how to extract the ground state from a maximally truncated $|\gamma_{opt}, \beta_{opt}\rangle$ state $\equiv |\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}=1}$, with all entanglement removed. That is, from a product state. After that, this methodology used to extract the ground state from truncated states will be generalised to handle entangled states also with $D_{\max} > 1$.

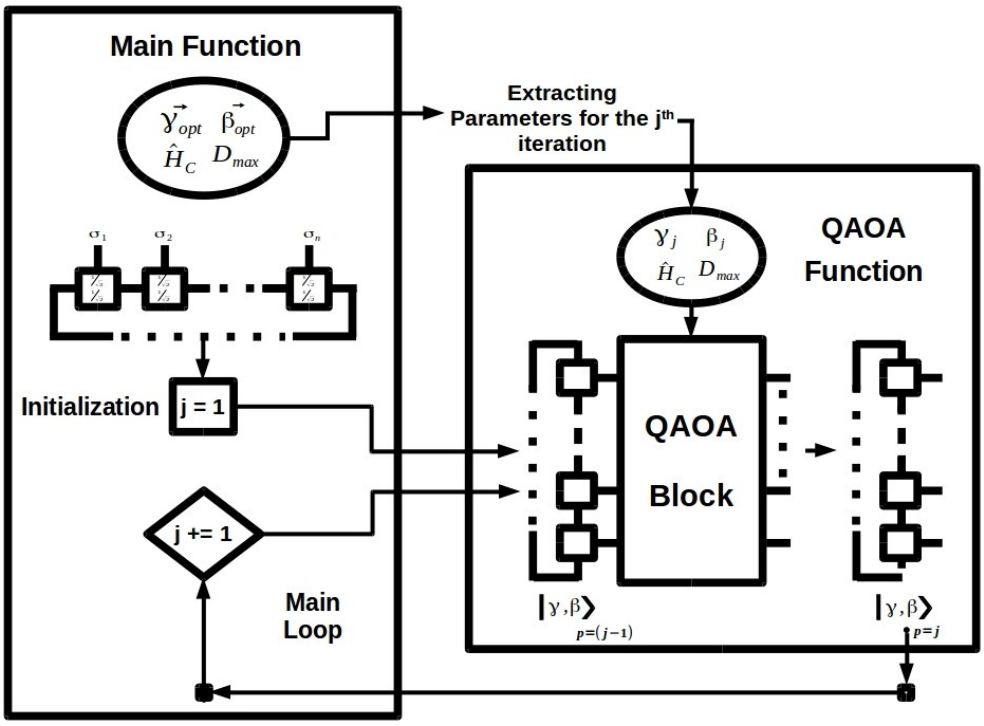


Figure 7.1: Illustration of the $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state preparation pseudo-code.

7.2 State Preparation

Before diving into the actual method of solution extraction, it may be instructive to go over the pseudo-algorithm we employed to create the state $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$. For this, we have coded for two functions. First is the main function that accepts the Hamiltonian H_C, the $2p$ optimum angles $\{\gamma_{opt}\}$ and $\{\beta_{opt}\}$, and the maximum bond dimension $D_{\max}$. Second is a QAOA-block function that accepts an initial matrix product state, and the relevant parameters as input, applies one QAOA block to the input state, and returns that as the output state. Hence, the idea is that the main function starts off with an initial $|+\rangle^{\otimes n}$ state, and repeatedly calls the QAOA block function p times, each time feeding the output of the preceding call as the input to the next one. Figure 7.1 is a pictorial illustration of the applied algorithm.

7.3 Extraction from Product States: $(D_{\max} = 1)$

Recalling again from chapter 2, subsection 2.6.3, each tensor in an n qubit matrix product state register, for $D_{\max} = 1$, is a $1\times2\times1$ matrix. Also, the two parameters in these tensors correspond to the coefficients of the $|0\rangle$ state and the $|1\rangle$ state, hereby referred to as α_k and ζ_k respectively, for the k^{th} qubit. At face value, one could consider finding the maximum coefficient among all the 2^n coefficients C_i for each state $|i\rangle$, to be an exponentially hard problem. This is because although finding the maximum in an array is a problem that scales linearly with the array size, the array size itself grows exponentially, making the entire problem exponentially scaling.

What is interesting, however, is to note that any coefficient C_i of any state $|i\rangle$ of the entire qubit register, is a product of these $\{\alpha_k\}$ and $\{\zeta_k\}$ for $k = 1, 2, \ldots, n$. That is,

$$\begin{aligned} \forall\ |i\rangle &= |k_1 k_2 \ldots k_j \ldots k_n\rangle \\ C_i &= \prod_{j=1}^{n} x_{k_j} \\ x_{k_j} &= \begin{cases} \alpha_{k_j} & if\ \ |k_j\rangle = |0\rangle \\ \zeta_{k_j} & if\ \ |k_j\rangle = |1\rangle. \end{cases} \end{aligned} \tag{7.2}$$

The matrix product state framework allows us to isolate and inspect the state of each qubit through its physical index (refer chapter 2, section 2.6). Hence, to find which coefficient is the maximum, we do not need to search through the full Hilbert space with 2^n basis states. Instead, we only have to search in n independent Hilbert spaces of dimension 2^1, as there is no entanglement and these are un-correlated spaces. Mathematically, this is also because the product operation and maximum operations commute, given the operands are independent. That is, the maximum of products is the same as the product of maximums for independent scalars.

$$\max\ (\ \prod_{k=1}^{n} x_k\ )\ =\ \prod_{k=1}^{n} \max(\ x_k\ )\quad \forall\ x_k\ \in\ \mathbb{C}. \tag{7.3}$$

Physically, this means that since there is no entanglement within the system, one can independently sample from each n qubits to get information about the entire register. Thus, to obtain the basis state with the highest population within the $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}=1}$ state, one only needs to find which is the highest coefficient among the $|0\rangle$ and $|1\rangle$ state, and choose that to be included in the product. This is again because of Equation 7.3 and since the coefficient of that state is a product of individual qubits' $|0\rangle$ and $|1\rangle$ state probabilities.

Thus, the strategy now is to build the solution string of '0s' and '1s' bit by bit. Assign $'0'$ to the k^{th} bit if $\alpha_k\ \geq\ \zeta_k$ or $'1'$, otherwise.

$$\begin{aligned} \text{Solution} &= \text{"}k_1 k_2 k_3 \ldots k_n\text{"} \\ k_j &= \begin{cases} '0' & \text{if}\ \ \alpha_j\ \geq\ \zeta_j \\ '1' & \text{otherwise.} \end{cases} \end{aligned} \tag{7.4}$$

7.4 Extraction from Entangled States: ($D_{\max} > 1$)

A matrix product state representation where the maximum bond-dimension limit $D_{\max}$ is set to 1 is the highest an entangled quantum state can be approximated to.

In the framework of QIOA explained in section 7.3 where $D_{\max} = 1$, there is zero entanglement within the register. This may be a needlessly stringent approximation because for an algorithm to be classically tractable, it is still okay for the amount of entanglement, and hence the bond-dimension, to scale polynomially with system size [28]. Also, the entanglement within a register corresponds to correlations among the qubits in the register. Hence, we may be able to extract better performance from QIOA if we somehow manage to make use of these correlations. Heuristically, we observed that product state QIOA can extract the solution of a problem after sufficient circuit depth p really well, in the cases where there is a unique solution to the problem at hand. Such as in exact-cover problems. That is, for problems where the ground state of the cost Hamiltonian H_C is non-degenerate and not entangled.

However, in the case of problems such as MaxCut [47] where a graph has to be bi-partitioned into to two maximally disjoint sets, the two sets can arbitrarily be assigned to be set $'0'$ or set $'1'$. Hence, for any solution string, its bit flipped version would also be a solution (refer chapter 4, subsection 4.2.1 for more details). This makes all the energy states of max-cut Hamiltonians, including the ground state, to be not only degenerate but also maximally entangled. Thus, intuitively we could hypothesise that removing absolutely all entanglement would prove to be detrimental for capturing the essence MaxCut problems. Moreover, we show in chapter 8, section 8.2 that for circuit depth $p = 1$, a product state with $D_{\max} = 1$ is unable to capture any information about the system for max-cut problems, confirming this intuition. However, the essence of exact-cover instances with non-degenerate ground states is still captured by states with $D_{\max} = 1$.

Hence, to also handle problems such as max-cut with maximally entangled energy states, the QIOA method of extracting the state $|s\rangle$ with the highest population from $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ need to be generalised for handling $D_{\max} > 1$ as well. But applying QIOA on entangled states is not as straightforward as it was in Equation 7.4. This is because, while for product states, the $|0\rangle$ and $|1\rangle$ states were represented by scalars α_k and ζ_k, for higher bond-dimension, α_k and ζ_k are matrices. When $D_{\max} > 1$ and all $D_i \leq D_{\max}$ for $i \in \{1, 2, \ldots, n-1\}$ where D_i is the i^{th} bond-dimension, α_k and ζ_k are $D_{k-1} \times D_k$ matrices (except when $k = 1$ or $k = n$ where they are $1 \times D_1$ and $D_{n-1} \times 1$ matrices respectively). As a result, to compare between the $|0\rangle$ and $|1\rangle$ states, we need to find a way to compare between matrices. To this end, we tried employing many mappings that convert matrices to scalars so as to compare between them. We tried methods inspired from machine learning such as trace distance, and determinants of the square matrix $M \times M^\dagger$ where $M^\dagger$ is the Hermitian conjugate of matrix M, but none showed promising results. In the end, we came to the conclusion that to also account for the correlations among qubits, we would have to use quantum mechanical methods such as calculating reduced density matrices of the qubits to extract the solution, and this worked. We have employed two such reduced density matrix based algorithms to extract the solution. In the first one, we use the reduced density matrices themselves directly. The second method, which we are referring to as the projected reduced density matrix method, is specifically tailored to account

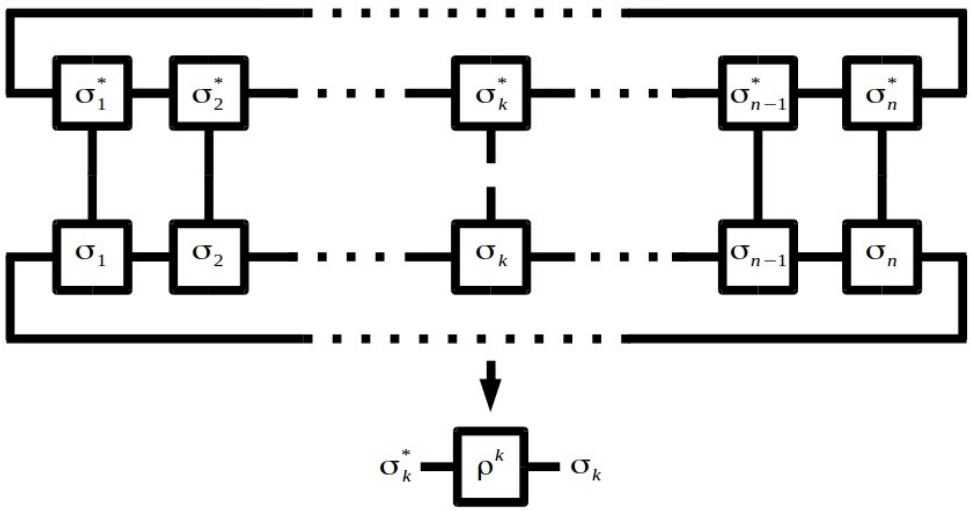


Figure 7.2: Single qubit reduced density matrix calculation of the k^{th} qubit using matrix product states

for the qubit correlations through projection operations. Henceforth, these methods would be called the DM, and PRDM methods respectively, for ease of reference.

7.4.1 Reduced Density Matrix Method

$$\begin{aligned} \text{Solution} &= \text{"}q_1\ q_2\ q_3\ \ldots\ q_k\ \ldots\ q_n\text{"} \\ q_k &= \begin{cases} '0' & \text{if } \rho^k_{00} \geq \rho^k_{11} \\ '1' & \text{otherwise} \end{cases} \\ \forall\ k &\in \{1, 2, \ldots, n\}. \end{aligned} \tag{7.5}$$

The section 2.9 of chapter 2 talks about how to calculate a generalised density matrix of multiple qubits using matrix product states. For this implementation, we use that principle to generate the reduced density matrix of the k^{th} qubit $\equiv \rho^k$ by tracing out all the other qubits. Hence, modifying Figure 2.9, we get Figure 7.2. Density matrices are more generalised representations of quantum states that help us identify the populations of states by directly reading off its elements. Furthermore, from the single qubit reduced density matrix (Equation 7.6) of the k^{th} qubit, we know that ρ^k_{00} gives the probability of measuring qubit k to be in the state $|0\rangle$ while ρ^k_{11} is the probability of measuring qubit k to be in the state $|1\rangle$. Note that since both ρ^k_{00} and ρ^k_{11} are probabilities, they are both $\in [0, 1]$ and $\rho^k_{00} + \rho^k_{11} = 1$.

$$\rho^k = \begin{bmatrix} \rho^k_{00} & \rho^k_{01} \\ \rho^k_{10} & \rho^k_{11} \end{bmatrix}. \tag{7.6}$$

Hence, a simple strategy to build the solution string of the problem encoded by the

optimum parameters $\gamma_{opt}, \beta_{opt}$, is by calculating ρ^k of the k^{th} qubit, and assigning character $'0'$ to the k^{th} bit of the solution if $\rho^k_{00} \geq \rho^k_{11}$. Otherwise, assign $'1'$ to the k^{th} solution bit. Refer Equation 7.5 for a mathematical illustration of this discussion.

7.4.2 Reduced Density Matrix Method with Projections

$$\begin{aligned} \text{Solution} &= \text{"}q_1\ q_2\ q_3\ \ldots\ q_k\ \ldots\ q_n\text{"} \\ q_k &= \begin{cases} '0' & \text{if} \quad \rho^k_{proj}[0,0] \geq \rho^k_{proj}[1,1] \\ '1' & \text{otherwise} \end{cases} \\ \forall\ k &\in \{1, 2, \ldots, n\}. \end{aligned} \tag{7.7}$$

Although the reduced density matrix method described above works well for $D_{\max} > 1$ as well, because we are tracing out the other qubits while considering the state of a single qubit, we are not making use of the correlations in the system. Moreover, in problems such as max-cut with maximally entangled solution states, both a bitstring and its bit-flipped version, are equally likely solutions. Hence, when considered individually, both $'0'$ and $'1'$ are equally correct possibilities for all qubits. Thus, we proposed a modified method of solution state extraction that is based on the reduced density matrix method.

The idea is that as we already know the states of qubits $1, 2, \ldots, k-1$ before we move on to calculate the state of qubit k, it makes sense to project the known qubits to their respective states. This makes sense because now we are no longer be tracing out the influence of the preceding qubits entirely. Instead, we would be retaining and bringing to the forefront, the effects of the correlations among the qubits as a result of $D_{\max} > 1$. Also, this is a simple fix to avoid the problem faced earlier by the reduced density matrix where each qubit was equally likely to be $'0'$ and $'1'$ when considered individually. This is because we are no longer deciding the state of qubit k by considering it individually. Furthermore, the results described in chapter 9 shows that this method of projected reduced density matrix calculations requires lesser circuit depth p to start working for a larger bond dimension D as expected. This is because as D increases, the number of correlations also increase.

For this method to work, first, the entire matrix product state has to be left-canonicalised, if we start building the solution string from the left, and right-canonicalised if starting from the reverse. Now, both the expectation values obtained after projecting the first qubit into the $|0\rangle$ state and the $|1\rangle$ state respectively, are calculated. Now, depending on which is value is higher, we assign either bit $'0'$ or bit $'1'$ to the first qubit. Next, since we already know the state of the first qubit, we project the first qubit of the matrix product state on to the calculated state. Then we move on to the next qubit, and so on using the same procedure. We shall call the projected reduced density matrix of qubit k, where the qubits $1, 2, \ldots, k-1$ are

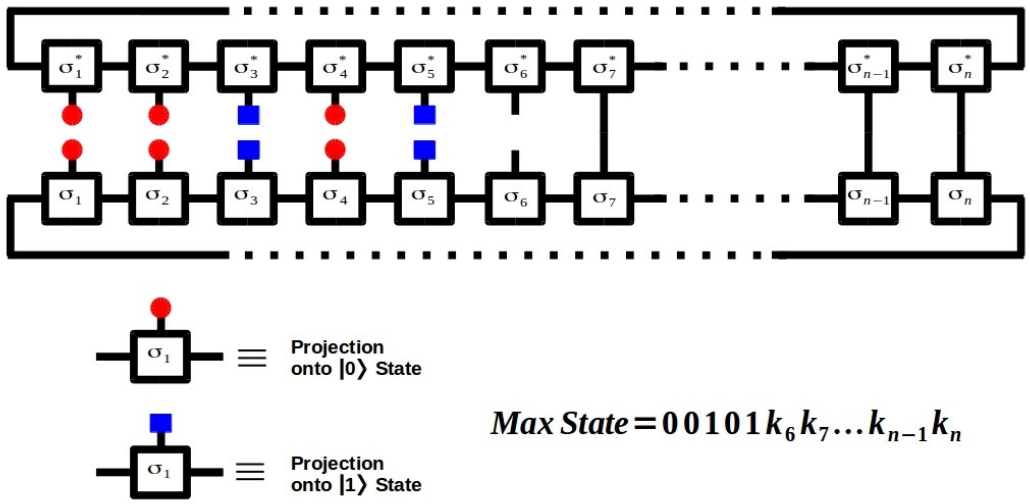


Figure 7.3: An example of calculating the projected reduced density matrix of qubit 6 in an n qubit register. Here, qubits 1 *to* 5 have already been mapped onto $'00101'$

all projected on to the calculated as ρ^k_{proj}. Now, we repeat the same procedure as in subsection 7.4.1 by comparing between $\rho^k_{proj}[00]$ and $\rho^k_{proj}[11]$ to assign either $'0'$ or $'1'$ depending on which is larger. Calculating the projected density matrix of qubit 6 when qubits 1 *to* 5 have already been mapped to $'00101'$ from an n qubit matrix product state is illustrated in Figure 7.3

This entire procedure is pictorially depicted in Figure 7.4 using a tree structure for all the 2^3 possible scenarios of a 3 qubit instance. In this figure, the left branch always assumes that $\rho^k_{proj}[00] > \rho^k_{proj}[11]$ while the right branch assumes the opposite. Do note that during an actual implementation, one would be able to make those comparisons and hence would be traversing along one single branch along this tree. Hence, although there are an exponential number of branches $= 2^n$, traversing along a single branch is a linear problem of $\hat{O}(n)$.

7.5 Summary

In this chapter, we have primarily gone through the implementation of QIOA, Quantum Inspired Optimisation Algorithm. QIOA is a quantum inspired classical algorithm derived from the original QAOA [2].

- **section 7.1** describes the main essence behind QIOA.
- After that, we briefly go over the $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state preparation in **section 7.2**.
- Now that we have the state $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ prepared, we go over the method of extracting the basis state with maximum probability from $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ with $D_{\max} = 1$. That is, a maximally approximated product state.
- Next, we extend this state extraction method for states having non-zero cor-

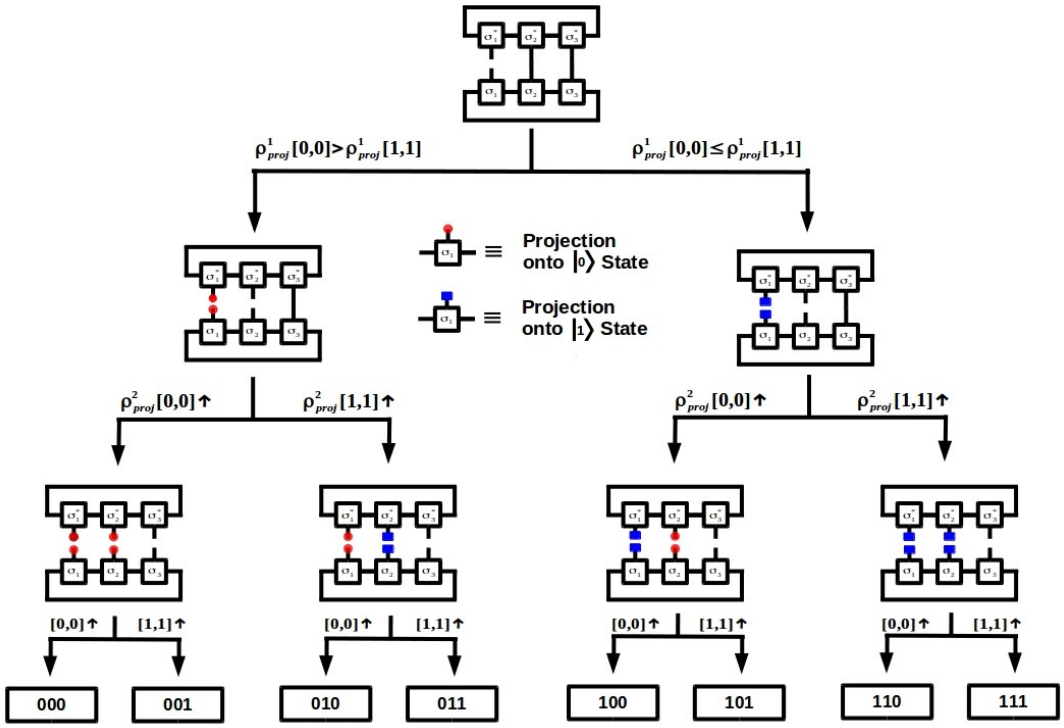


Figure 7.4: All the 2^3 possible Projected density matrix implementation scenarios depicted as a tree for the case of 3 qubits

relations using two different methods as described in **subsection 7.4.1** and **subsection 7.4.2**.

It is worth re-emphasising here that through QIOA, we are attempting to find out which basis state from the computational basis holds the highest probability, in the entire truncated $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$. The primary idea behind the original QAOA paper [2] is to increase the probability of measurement of the solution to a problem which is encoded as the ground state of the cost Hamiltonian H_C, by increasing the circuit depth p. With increasing circuit depth, the entanglement among the qubit registers increases exponentially, and this makes it intractable for classical computers to simulate QAOA for large systems and depths.

A big part of this thesis tries to answer the question if the same behaviour persists, that is, the probability of measurement of the ground state increases with increasing p, even for truncated systems where either there are no entanglement, or the entanglement only grows polynomially with system size.

Hence, we calculate the state holding the maximum probability of measurement in the truncated state, and cross-verify if that is the solution to the problem we set out to solve. Turns out, in many cases, (in all cases we simulated), the idea holds and we can extract the solution from classically tractable $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state.

Part III

Results & Discussion

8

Training Results and Discussions

In this chapter, we go over the results concerned with the training phase of QAOA. Please revert to chapter 3 if you'd like to have a refresher on QAOA and to chapter 6 for the training methods employed. For these results, we shall be sticking to the grid search method talked about in section 6.1 and a circuit of depth $p = 1$.

As a review, during the training phase, we first create a parametrized quantum state $|\gamma, \beta\rangle$ starting from the uniform superposition state $|+\rangle^{\otimes n}$. We do this by applying single and two qubit gates defined using the cost Hamiltonian H_C and the parameters $\gamma,\ \beta$ onto $|+\rangle^{\otimes n}$ (refer chapter 5 for details). We also have an option to limit the amount of entanglement in the system by setting a maximum bond-dimension $D_{\max} \leq 2^{\lfloor n/2 \rfloor}$ (Equation 2.10) on all the bonds within the matrix product state representation of the $|\gamma, \beta\rangle$. This results in a possibly truncated state $\equiv |\gamma, \beta\rangle^{D_{\max}}$ (section 5.3). Here $2^{\lfloor n/2 \rfloor}$ is the highest possible bond-dimension in an n qubit MPS register. Next, we calculate the expectation value of the cost Hamiltonian H_C with respect to this, possibly truncated, parametrized state $|\gamma, \beta\rangle^{D_{\max}}$. This expectation value is then defined as the cost $C(\gamma, \beta, D_{\max})$ associated with the angles $\gamma,\ \beta$ and bond-dimension $D_{\max}$. That is,

$$C(\ \gamma, \beta, D_{\max}\ )\ =\ {}^{D_{\max}}\langle\ \beta, \gamma\ |\ H_C\ |\ \gamma, \beta\ \rangle^{D_{\max}}. \tag{8.1}$$

We then find the optimum parameters $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ that correspond to the global minima of the cost function $C(\gamma, \beta, D_{\max})$ such that,

$$C(\ \gamma_{opt}, \beta_{opt}, D_{\max}\ )\ \leq\ C(\ \gamma, \beta, D_{\max}\ )\ \forall\ (\gamma, \beta)\ \in\ [0, 2\pi]. \tag{8.2}$$

It is this training phase that encompasses the variational aspect of QAOA. Here, the idea is that the population of the ground state $|gs\rangle$ within the $|\gamma, \beta\rangle^{D_{\max}}$ state is maximum at the global minima of the cost function $C(\gamma, \beta, D_{\max})$ This is assured in the original QAOA when $D_{\max} = 2^{\lfloor n/2 \rfloor}$, and in this chapter, we verify if this still holds true for smaller $D_{\max}$ as well. Further, we define the solution landscape, which is the magnitude square of inner product between the ground state $|gs\rangle$ and the $|\gamma, \beta\rangle^{D_{\max}}$ state represented in %, as $S(\gamma, \beta, D_{\max})$. This is also the same as the success probability of finding the solution state from the $|\gamma, \beta\rangle^{D_{\max}}$ upon sampling from it. In the case where there exists multiple equally likely solutions $|gs_1\rangle,\ |gs_2\rangle,\ \ldots,\ |gs_m\rangle$, as in max-cut problem instances, (section 4.2), the ground state $|gs\rangle$ is itself an equal

superposition of all the possible m solutions. In such cases, $S(\gamma, \beta, D_{\max})$ is the sum of squares of individual inner-products times 100

$$\begin{aligned} |gs\rangle &= \frac{1}{\sqrt{m}} \times (\ |gs_1\rangle \ + \ |gs_2\rangle \ + \ \ldots \ + \ |gs_m\rangle \ ) \\ S(\gamma, \beta, D_{\max}) &= || \ \langle \ gs \mid \gamma, \beta \ \rangle^{D_{\max}} \ ||^2 \ \times \ 100 \\ &= \frac{1}{m} \times (\ || \ \langle \ gs_1 \mid \gamma, \beta \ \rangle^{D_{\max}} \ ||^2 \ + \ || \ \langle \ gs_2 \mid \gamma, \beta \ \rangle^{D_{\max}} \ ||^2 \ + \\ &\ldots \ + \ || \ \langle \ gs_m \mid \gamma, \beta \ \rangle^{D_{\max}} \ ||^2 \ ) \ \times \ 100. \end{aligned} \tag{8.3}$$

In the framework of QAOA, due to the variational principle of quantum mechanics, and the ground state being the solution to the problem encoded in H_C, the point corresponding to the global maxima of the solution function $S(\gamma, \beta, D_{\max}) \equiv (\gamma_{max}, \beta_{max})$ corresponds with the global minima point of the cost function. The difference between these points ($\equiv \Delta\gamma$ and $\Delta\beta$) reduces with increasing circuit depth p. In this sense, it is safe to say that during the training phase, all one cares about, is finding the global minima of the cost function and if this minima corresponds to a maxima in the solution landscape $S(\gamma, \beta, D_{\max})$. Hence, any approximation made which preserves the position of the global minima of $C(\gamma, \beta, D_{\max})$ and its correspondence with $S(\gamma, \beta, D_{\max})$ can be considered as a valid approximation.

Recalling from subsection 2.6.2, in a non-truncated matrix product state the highest possible entanglement, and hence the maximum bond dimension $D_{\max}$, grows exponentially with the system size along the n qubit register. This quickly makes it intractable for classical hardware to keep track of the quantum state as n increases. Thus, the questions we are answering through our work are twofold:

- Firstly, if one can still get the desired parameters $\gamma_{opt}, \beta_{opt}$ used in the original QAOA, using a truncated state with an upper bound on the bond dimension that does not scale exponentially with system size. That is, if

$$\begin{aligned} || \ \gamma_{opt}(D_{\max}) \ - \ \gamma_{opt}(2^{\lfloor n/2 \rfloor}) \ || \ &< \ \epsilon \\ || \ \beta_{opt}(D_{\max}) \ - \ \beta_{opt}(2^{\lfloor n/2 \rfloor}) \ || \ &< \ \delta \\ \text{for } D_{\max} \ &\ll \ 2^{\lfloor n/2 \rfloor}, \end{aligned} \tag{8.4}$$

  where ϵ and δ are not too large. How large ϵ and δ can be depends on the difference between the success probability obtained using the original parameters $\gamma_{opt}(2^{\lfloor n/2 \rfloor}), \beta_{opt}(2^{\lfloor n/2 \rfloor})$ and the truncated parameters $\gamma_{opt}(D_{\max}), \beta_{opt}(D_{\max})$.

- Secondly, if one could train the angles used in QIOA (chapter 7, section 7.2) using the truncated $\mid \gamma, \beta \ \rangle^{D_{\max}}$ state and $C(\gamma, \beta, D_{\max} \leq 2^{\lfloor n/2 \rfloor})$, the cost landscape associated with it.

This chapter goes over results addressing the first question raised above, and in the next chapter we discuss results on the QIOA method while attending to the second question. The following pages present multiple calculations of the cost and solution landscapes $C(\gamma, \beta, D_{\max})$ and $S(\gamma, \beta, D_{\max})$ respectively, over varied maximum bond-dimension $D_{\max}$ of various Max-Cut (section 4.2) and Exact-Cover (section 4.1) instances.

8.1 Cost and Solution Landscapes

One of the most curious things about the cost landscape $C(\gamma, \beta)$, and hence the solution landscape $S(\gamma, \beta)$, is that it is quite independent of not only the particular problem instance it is generated from but also the number of variables in that instance. This essentially makes the whole landscape, instance independent and has been already well documented by Fernando et. al. [48], with the creators of the original QAOA. Here, they prove that if the parameters within the instance definition are sourced from a reasonable distribution, then the values of $C(\gamma, \beta)$ is nearly the same for all these instances. In fact, they claim that it is even possible to train QAOA on a smaller instance of a similar problem and use those angles for bigger instances as a way to conserve resources. This is exactly what was observed in all our calculations as all instances from the exact-cover problems had almost the same cost landscape, and all instances from max-cut had very similar but different from the exact-cover landscape. This is because as detailed in subsection 4.2.2, we create our problem instances using a uniform probability distribution, which is one of the well-behaved distributions Fernando talks about in their work.

We have observed that this same behaviour persists to a large extent even on truncating the bond-dimension too. Reiterating chapter 2, subsection 2.6.2, one could represent any quantum state exactly using matrix product states by letting the maximum possible bond dimension $D_{\max}$ to be $2^{\lfloor n/2 \rfloor}$, for an n qubit register. Hence we already know that MPS based calculations of QAOA with this highest possible bond dimension will follow the characteristics described in the above paragraph. In fact, this was how we benchmarked the code used in this thesis, and also how we verified its validity. However, it was still an open question if the same characteristics persist even after reducing the bond dimension by setting the maximum dimension $D_{\max} \ll 2^{\lfloor n/2 \rfloor}$. Being able to obtain comparable results by reducing $D_{\max}$, which originally scales exponentially with system size, to something that scales polynomially with n, would make the training phase classically tractable. This would then suggest the possibility of eliminating the need for a quantum processing unit (QPU) during the training phase. Previously this year, the Data Lab of Volkswagen Group had published a paper on the possibility of training QAOA without using a QPU [32]. While their method is also based on Tensor Networks, they employ tree tensor networks which are fundamentally different from the MPS methods we use. Their work also tries to handle the system exactly without making any approximations, making these two approaches inherently different. Nevertheless, having the same conclusion from a completely different perspective is highly encouraging.

8.2 Max-Cut Training Results

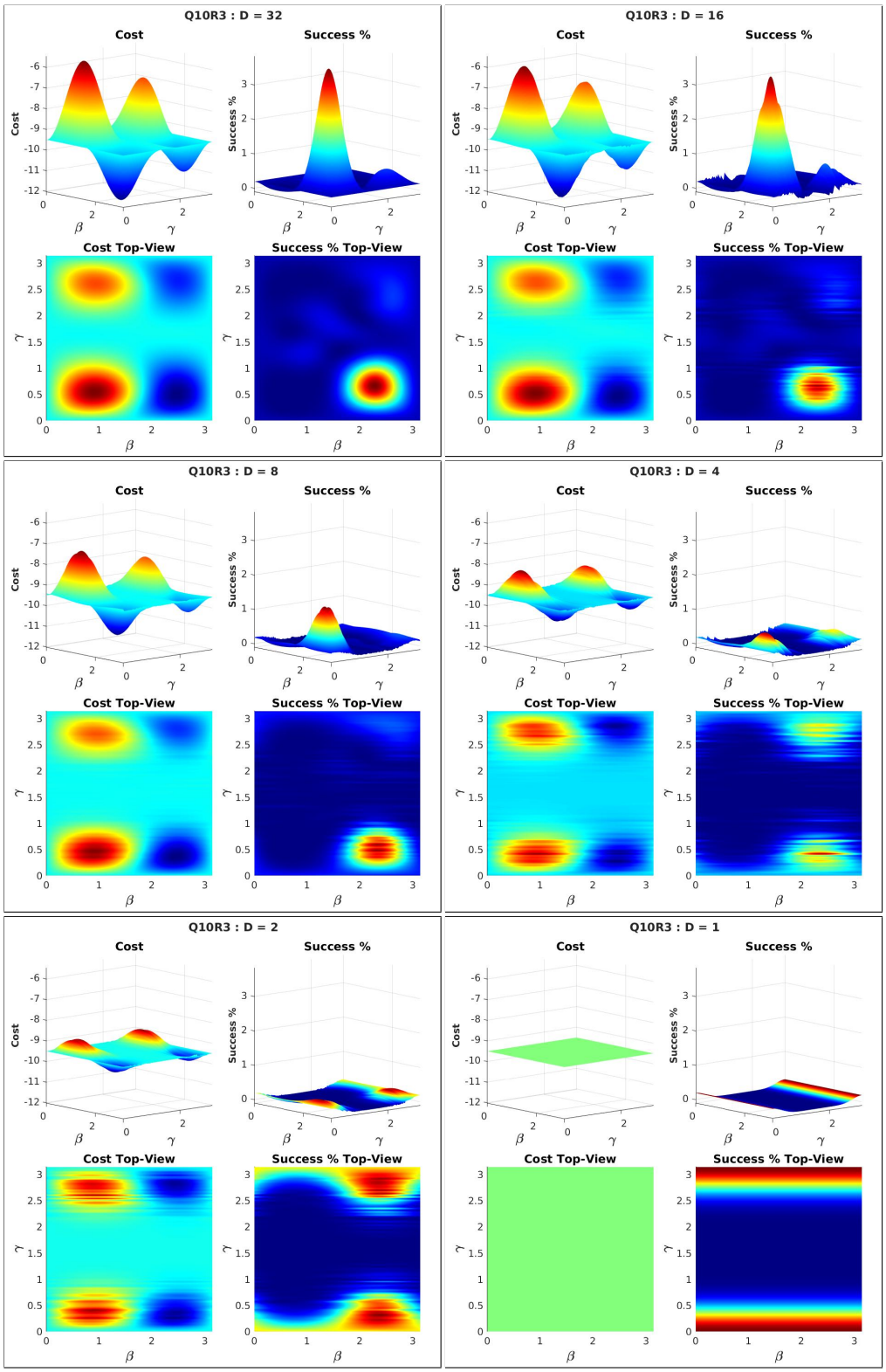


Figure 8.1: An example of the azimuthal and top views of Cost and Solution Landscape of a 10 qubit Max-Cut instance changing with representative Bond dimensions $D = 32,\ 16,\ 8,\ 4,\ 2,$ and 1. For a 10-qubit system, highest $D = 2^{\lfloor 10/2 \rfloor} = 32$.

Because all the calculated cost and solution landscapes look extremely similar for all the instances from within the same problem family, we will present representative landscape images and all calculated plots for one from each max-cut and exact-cover problems. Figure 8.1 presents the calculation results of a 10-qubit max-cut instance. Also, the graph of this instance whose maximum cut we are trying to find is depicted in Figure 4.2 as an example of a randomly generated Erdős–Rényi graph [42]. Each of the 6 panels in that image presents the azimuthal and top views of both the cost and solution landscapes for bond dimensions $D = 32,\ 16,\ 8,\ 4,\ 2$ and 1. Some of the key points of interest about these figures are listed below:

- The most striking feature here is that, although we are making approximations to the system by truncating the bond dimension and limiting the net entanglement in it, the cost and solution landscapes largely retain their form. That is to say, although the magnitude does decrease, the position of the minimas and maximas largely remain unchanged. Recalling from the last section discussing the cost and solution landscapes, since all one cares about is the position of the global minima in the cost landscape, this strongly suggests that one can still derive useful information from these approximated cost landscapes $C(\gamma, \beta, D_{\max})$ also, even when $D_{\max} \ll 2^{\lfloor n/2 \rfloor}$.

- As seen in Figure 8.1, most cost landscapes have two minimas, one global and one local, and many have two corresponding peaks in their solution landscapes too. One subtle, but important point to be raised here is that having a minima in the cost landscape need not always translate to having a peak in the solution landscape. This is because in some cases, those minimas are created by $|\gamma, \beta\rangle^{D_{\max}}$ states which are primarily populated by other low-energy states like the first and second excited states. This is probably what is happening in this particular instance too, as we can hardly observe a peak in $S(\gamma, \beta, D_{\max})$ over the second minima in $C(\gamma, \beta, D_{\max})$.

- We can also see that the magnitude of the landscape peaks decreases considerably by reducing bond dimensions. This is primarily because reducing the bond dimension to $D_{\max}$ is essentially throwing away all except the largest $D_{\max}$ Schmidt weights. As a result, the norm of the state $|\gamma, \beta\rangle^{D_{\max}}$ is no longer equal to one. This is not a bug in the method and is in fact by design. This is because normalising the state after each truncation introduces spurious effects which alter the shape of the landscapes, which we want to avoid. The introduction of these spurious effects is because the Schmidt weights are also functions of γ and hence the approximations made are not uniform for all γ.

- Combining insights from the above two points, we see that for smaller bond-dimensions, $D_{\max} = 4$ and $D_{\max} = 2$, although the overall magnitude of both $C(\gamma, \beta, D_{\max})$ and $S(\gamma, \beta, D_{\max})$ decrease, the presence of the second peak in $S(\gamma, \beta, D_{\max})$ becomes increasingly prominent. We have also observed that there have been shifts between local and global maximas both in the cost and solution landscapes too. While the exact reasons that govern these shifts are

yet to be studied, this could be because MPS based calculations preferentially try to preserve the low-lying energy states. Hence, the other higher energy states may have been subjected to larger reductions in magnitude via the approximations performed, making the ground state relatively more pronounced.

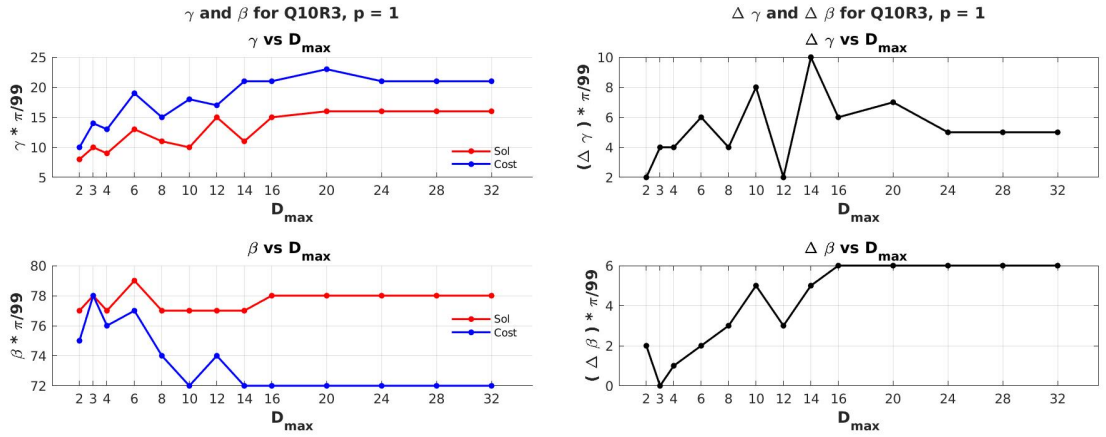
- The previous point sets the stage well for the next observation which is that the landscape plots for smaller bond dimensions look grainy along the γ axis. Especially visible in the top-views of panels for $D_{\max} = 8,\ 4,$ and 2 in Figure 8.1. We also note that there is no such graininess along the β axis. This is exactly because the Schmidt weights, which are representative of entanglement in the system, are only functions of γ and hence does not depend on β. Furthermore, the Schmidt weights are also observed to be periodic in γ, making some slices along γ more truncated than others. It is also because of this non-uniform truncation, that normalising the states introduces non-uniform un-warranted effects.

- Next, it has already been proven that it is classically impossible to sample from the output of a QAOA circuit of large system sizes n by Farhi et. al. [31]. These landscape plots also corroborate that result as we see that information about the actual magnitudes of the solution states is lost completely upon truncation. This is to say that one cannot just simply sample from this truncated state to obtain the solution state.

- Another surprising aspect of these landscape plots is the almost complete lack of retrievable information from the fully truncated state with $D_{\max} = 1$. We believe that this is because the state $|\gamma, \beta\rangle^{D_{\max}=1}$ is a product state, and since all the energy states associated with max-cut problem instances are at least doubly-degenerate as explained in section 4.2, a parametrized state of at least bond-dimension 2 is required to capture its essence. This is in stark contrast with the results seen for similar $D_{\max} = 1$ landscapes of the exact-cover instances where no such degeneracy exists, further supporting this claim.

- In QAOA, the main idea is to minimize the cost landscape, and not maximise the solution landscape. In the infinite limit, when circuit depth $p \to \infty$, a minima in the cost landscape will undoubtedly correspond to a maxima in the solution landscape, because in that limit, QAOA is the same as an infinite run-time Quantum Annealing [2]. However, in the intermediate limit, in some cases, a maxima in the solution landscape may correspond to a very shallow minima, in the cost landscape. This is because the state created using those particular parameters may have a significant population of a very high energy state along with the ground state, giving net high energy, and thus a shallow minima. This insight was primarily derived from [39].

Again, the most important aspect of a cost landscape $C(\gamma, \beta, D_{\max})$ is the position



(a) $\gamma_{opt}(D_{\max}), \beta_{opt}(D_{\max})$ in blue and $\gamma_{max}(D_{\max}), \beta_{max}(D_{\max})$ in red

(b) $\Delta\gamma(D_{\max})$ and $\Delta\beta(D_{\max})$ in top and bottom panels respectively.

Figure 8.2: $(\gamma_{opt}, \beta_{opt})$, $(\gamma_{max}, \beta_{max})$, and $(\Delta\gamma, \Delta\beta)$ for $D_{\max} = 2, 3, 4, 6, 8, 10, 12, 14, 16, 20, 24, 28,$ and 32

of the global minima and the corresponding point $\gamma_{opt}(D_{\max}), \beta_{opt}(D_{\max})$. As we are interested in studying the influence of truncating the bond-dimension on these points, we calculated them for each bond-dimension $D_{\max}$ and represented it as a line plot in Figure 8.2a. Because these instances are small and we already know their solutions, we also calculated the solution landscape $S(\gamma, \beta, D_{\max})$, and their respective $\gamma_{max}(D_{\max}), \beta_{max}(D_{\max})$ point corresponding to the global maxima. This too was plotted along with $\gamma_{opt}(D_{\max}), \beta_{opt}(D_{\max})$ to keep track of their co-evolution with reducing bond-dimensions as shown in Figure 8.2a. We have also calculated the difference between these two points, $\equiv \Delta\gamma(D_{\max}), \Delta\beta(D_{\max})$ to explicitly see how that evolves with $D_{\max}$ and presented it in Figure 8.2b. The relevance of Figure 8.2b will be discussed more in detail when talking about the trainability of QIOA in the next chapter.

From Figure 8.2a, we see that there is a gradual shift in the angles γ_{opt} and β_{opt} on reducing the bond dimension. Recalling Equation 8.4, ϵ here $= 11 \times \pi/99$ and $\delta = 6 \times \pi/99$ where $\pi/99$ is the grid discretization used. As will be shown later, this is one of the more radical changes, and the median change is around 6 and 2 grid points for ϵ and δ respectively. The question now is how useful these angles $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ for small values of $D_{\max}$. To answer this, one can substitute these angles into the original solution landscape $S(\gamma, \beta, D_{\max} = 2^{\lfloor 10/2 \rfloor})$ for the full bond dimension to calculate how well they perform. From Figure 8.2b, we can see that even in the case when $D_{\max} = 2^{\lfloor 10/2 \rfloor}$, that is, even when there is zero approximations made, $\Delta\gamma$ and $\Delta\beta$ are not equal to zero. This is still okay because the global maxima of $S(\gamma, \beta, D_{\max} = 2^{\lfloor 10/2 \rfloor})$ is not a delta function and thus has a gradual slope to it. Hence, the changing of angles $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ for small values of $D_{\max}$ could still be okay as long as we are still able to land along the slope of the maxima. Although landing closer to the base will mean a lower success rate. To make reasonable comparisons between instances, we normalise

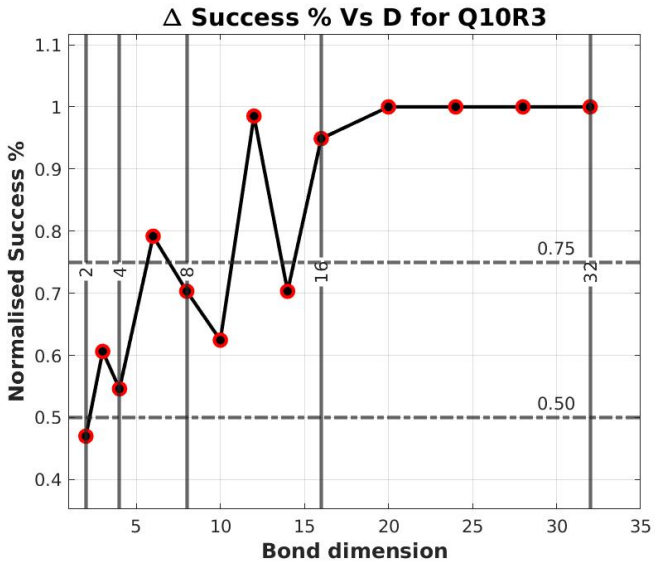


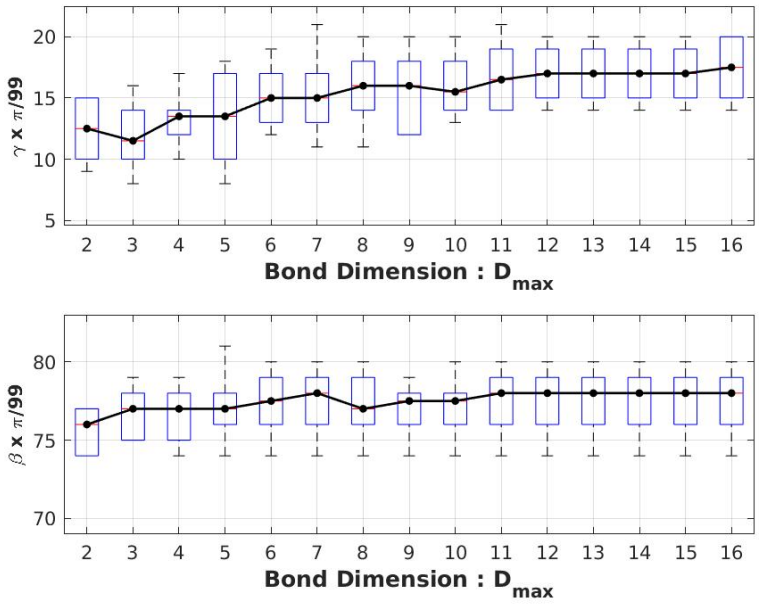
Figure 8.3: Normalised Success rate obtained using the angles $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ with respect to $D_{\max}$.

these probabilities by dividing it with the success rate obtained with no truncation.

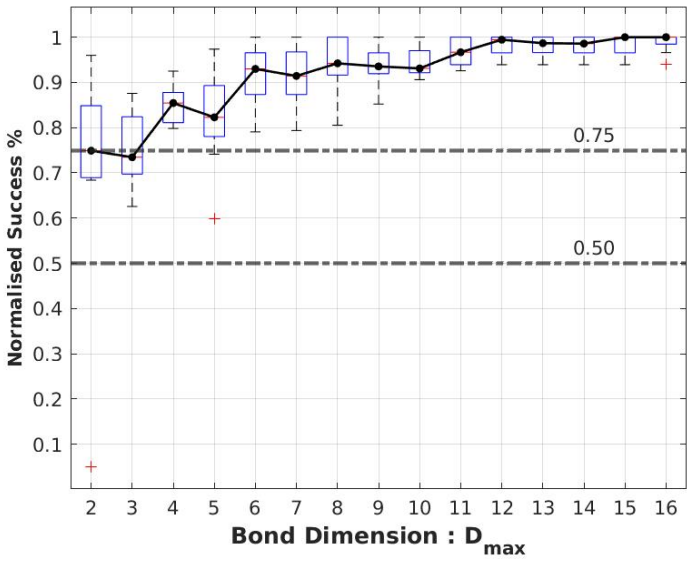
Figure 8.3 depicts the normalized performances of the angles $\gamma_{opt}(D_{\max})$, $\beta_{opt}(D_{\max})$ for different values of $D_{\max}$. The idea is that we create a full $|\gamma_{opt}(D_{\max}), \beta_{opt}(D_{\max})\rangle$ state using the original QAOA circuit, and then sample from this state to calculate the success probability. We now divide this obtained success probability with the success rate obtained on using the original γ_{opt}, β_{opt}, found by training vanilla-QAOA. A value of 1 signifies that the angles $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ perform as well as the the non-approximated angles $\gamma_{opt}(D_{\max} = 2^{\lfloor 10/2 \rfloor})$ and $\beta_{opt}(D_{\max} = 2^{\lfloor 10/2 \rfloor})$ derived from the original QAOA. The two dashed lines represent 75% and 50% of relative performance when compared with the original QAOA. Here, we see that even on making the biggest possible truncation, $D_{\max} = 2$, one is able to get a performance little shy of 50% when compared to the vanilla QAOA. At this point it is worth recollecting that the these results are for circuit-depth $p = 1$, and that for $p = 1$, the maximum success rate is a little more than 3% here (Figure 8.1). Thus a value of 1 in Figure 8.3 corresponds to having a success rate of 3%. The primary difference here is that vanilla QAOA calculates the non-approximated optimum angles by repeatedly calling a QPU, while the approximated angles can be easily calculated using classical hardware, as the truncated bond-dimension $D_{\max}$ does not scale exponentially.

Till now, we have been discussing the properties of a single max-cut instance Q10R3. In addition to this single max-cut instance we have discussed so far, we have run calculations for 49 other instances, all of which show similar behaviour. Hence, to summarize their average behaviour, we present box-plots [49] which is a great way to understand the overall statistical trend in the behaviour of the system at different bond dimensions. Every above-mentioned calculation have been performed for each

of the 10 specimens for all the 8, 9, 10, 11, and 12 qubit max-cut instances to make the statistical inferences composing of these box-plots.



(a) $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ in top and bottom panels respectively.

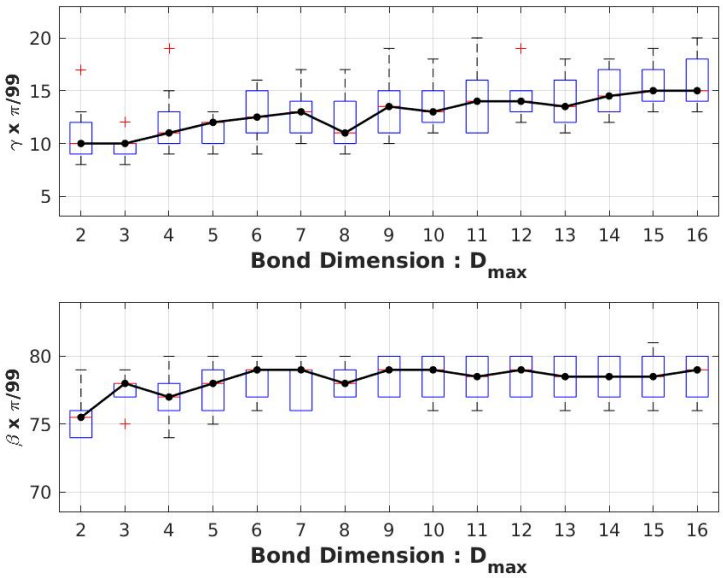


(b) Normalised Success rate of 8-qubit instances as a function of $D_{\max}$.

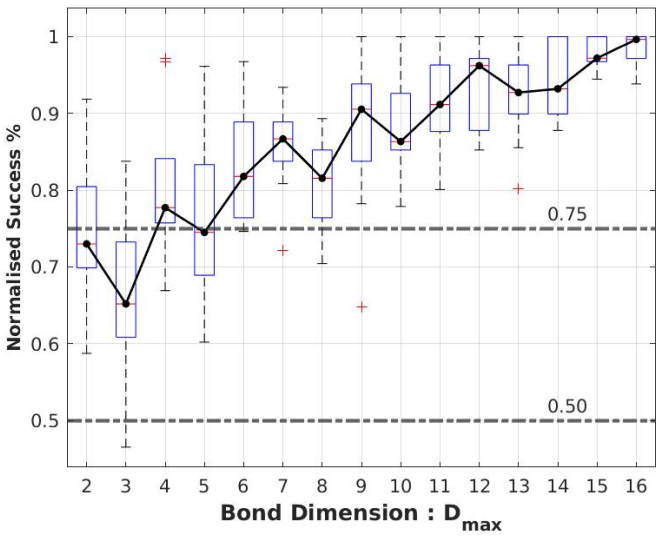
Figure 8.4: Calculated statistical behaviour of 8-qubit max-cut instances.

Figure 8.4 provides the average behaviour of all the 10 calculated 8-qubit instances. The X-axis in all plots correspond to the maximum bond-dimensions $D_{\max}$, and the black line in each plot connects the median behaviour of all the 10 instances at the respective bond-dimensions. Note that in Figure 8.4b for $D_{\max} = 2$, there

is an outlier point (represented by a red cross) caused by a shift in the minima angles. This is exactly what was mentioned previously where there is a relative switch between local and global minimas in some instances. Figure 8.4b presents the performance of the approximately calculated angles when used in the original QAOA for all the 10 8-qubit instances. We see that even in the worst case, we are still able to obtain a median performance of little less than 75% as good as a full quantum processor. The outlier case, however, provides a considerably poorer performance, but this can be resolved by choosing to sample over all minimas.



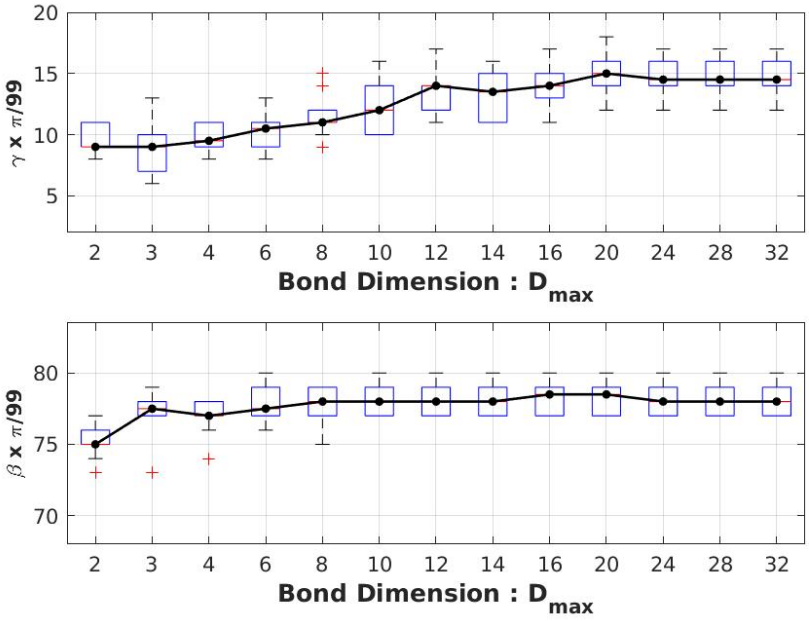
(a) $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ in top and bottom panels respectively.



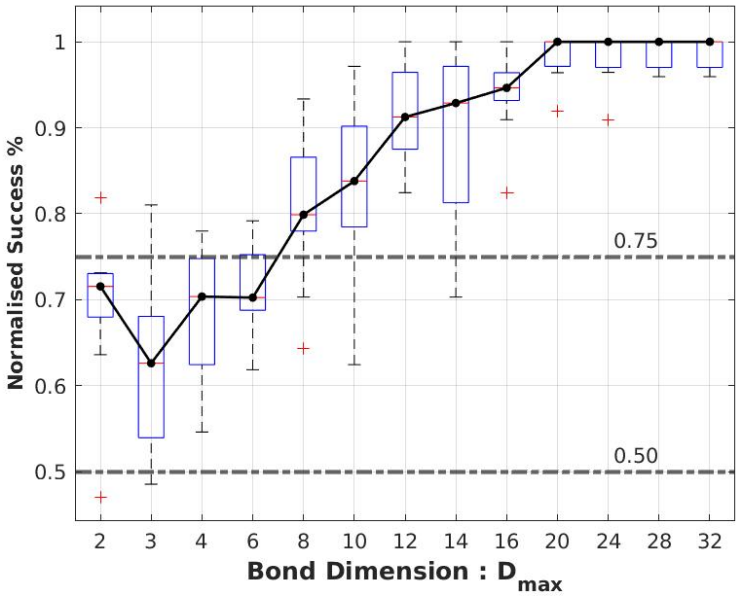
(b) Normalised Success rate of 9-qubit instances as a function of $D_{\max}$.

Figure 8.5: Calculated statistical behaviour of 9-qubit max-cut instances.

Figure 8.5 depicts the training behaviour of 9-qubit max-cut instances averaged over 10 examples. From Figure 8.5a, we can see that the maximum change in median angles is by 5 grid points for γ and 4 grid points for β. Also, Figure 8.5b shows that the worst median case behaviour is for $D_{\max} = 3$ with a worst-case relative performance little less then 50%. This could again be because of the doubly degenerate nature of max-cut as the performance improves for $D_{\max} = 2$.

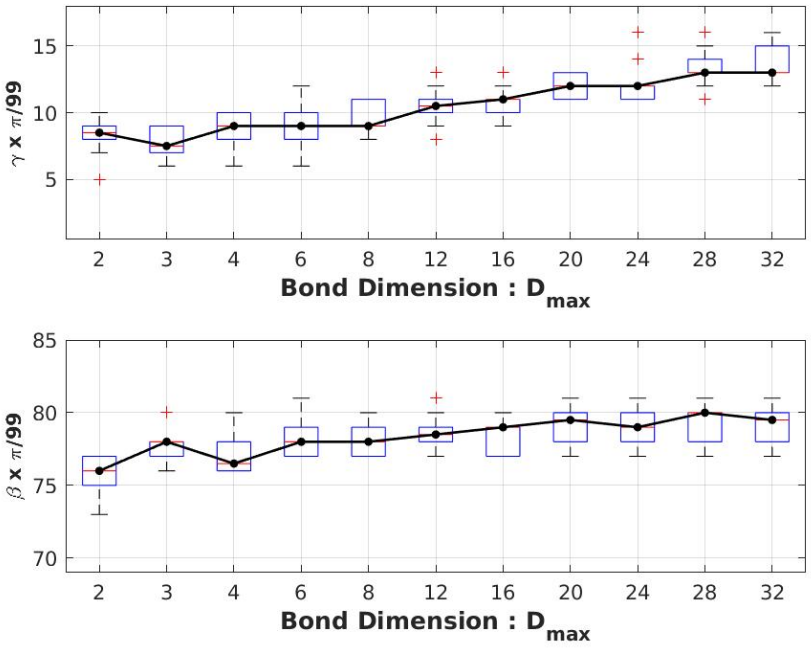


(a) $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ in top and bottom panels respectively.

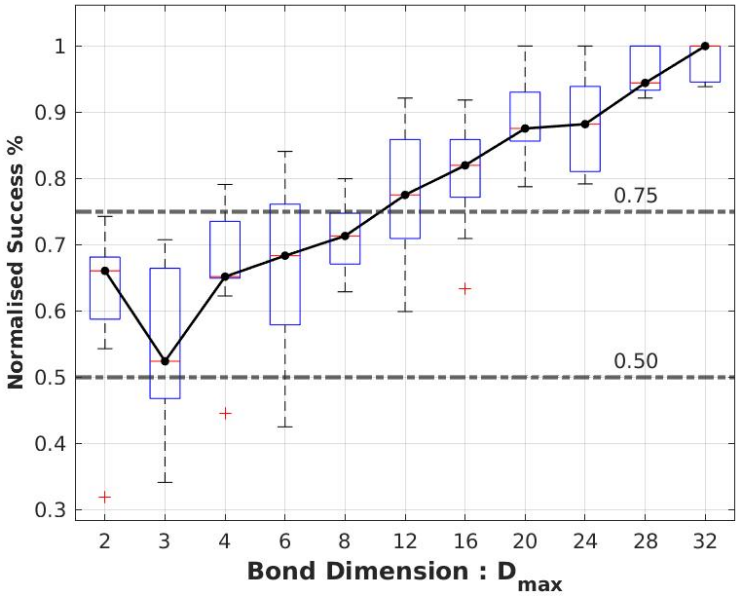


(b) Normalised Success rate of 10-qubit instances as a function of $D_{\max}$.

Figure 8.6: Calculated statistical behaviour of 10-qubit max-cut instances.



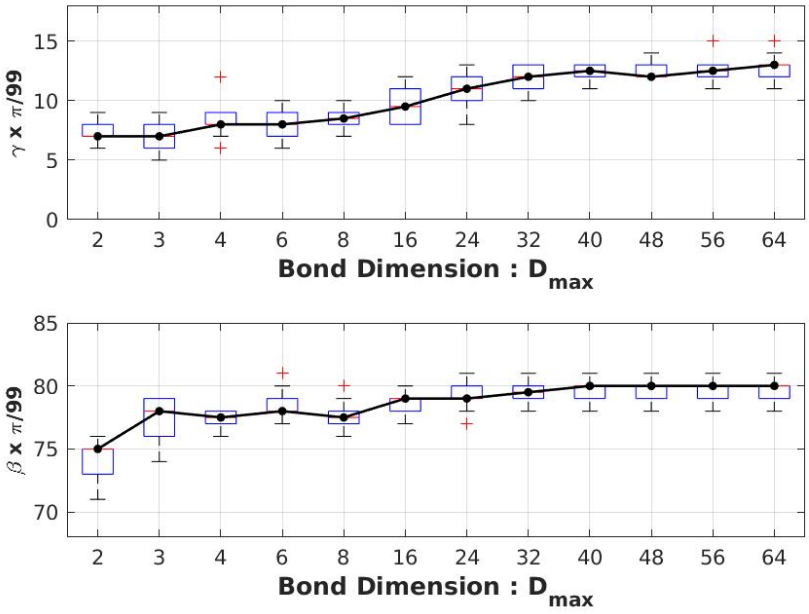
(a) $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ in top and bottom panels respectively.



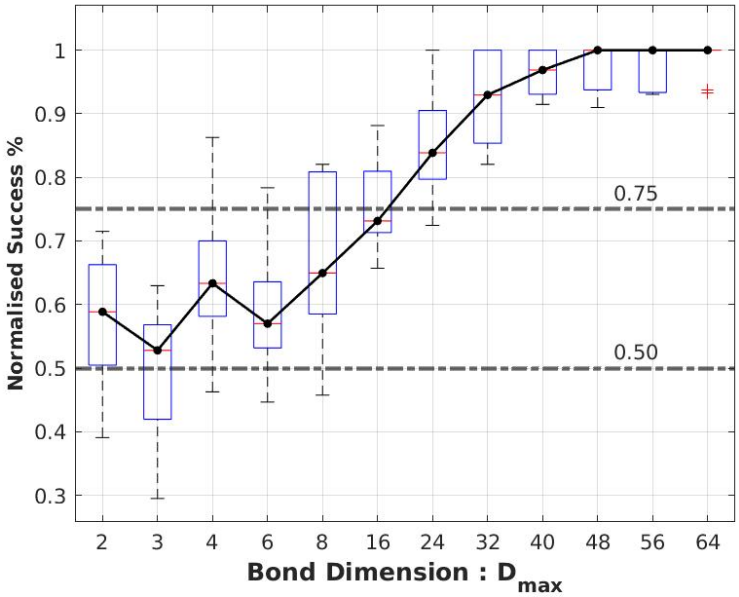
(b) Normalised Success rate of 11-qubit instances as a function of $D_{\max}$.

Figure 8.7: Calculated statistical behaviour of 11-qubit max-cut instances.

Here too in Figure 8.6b, we see a dip in performance when $D_{\max} = 3$. This point, that max-cut states seem to be subjected to higher approximations for odd bond-dimension will be raised also in chapter 9 while discussing some of the plots presented there.



(a) $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ in top and bottom panels respectively.



(b) Normalised Success rate of 12-qubit instances as a function of $D_{\max}$.

Figure 8.8: Calculated statistical behaviour of 12-qubit max-cut instances.

Figure 8.6, Figure 8.7 and Figure 8.8 present the average behaviour of 10, 11, and 12-qubit instances averaged over 10 examples, respectively. As explained in the previous cases, the (a) plot provides insight into how the global minima point γ_{opt}, β_{opt} evolved with the maximum bond-dimension $D_{\max}$ and plot (b) shows the normalised success rate of the angles $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ when used during the sampling phase of the original QAOA. Key points of insights from these plots are listed below:

- As mentioned above, in some calculations, changing the bond-dimension also introduces a switch between local and global minimas. This switch happens mostly in calculations of reduced bond-dimensions because higher-energy states are subject to larger approximations compared to the ground state. In such cases, substituting the changed global minima point into $S(\gamma, \beta, D_{\max} = 2^{\lfloor n/2 \rfloor})$ may not result in landing on a comparable peak. In this work, we have only kept track of the global minimas of all the landscapes, and we believe this is the reason why there are a few outlier and fringe cases with a poor relative performance for smaller bond-dimensions. This could easily be fixed by taking into account not only the global minima but also all local minimas for sampling and choosing the best relative performance for each bond-dimension. Since the number of local-minimas is finite, this only adds a constant factor to the problem scaling.

- Another aspect that could be leading to poor angle detection and possibly poor trainability over the cost-landscape $C(\gamma, \beta, D_{\max})$ when $D_{\max}) \ll 2^{\lfloor n/2 \rfloor}$ is the graininess over the γ axis as seen in Figure 8.1, and described in the following discussion. This could lead to multiple unintended small local minimas within the full landscape, decreasing the chances of finding the true minimas. This problem, however, could be overcome by using some smoothing function [50] that helps to capture the essence of the landscape by avoiding falling into such minimas. Such smoothing methods are routinely used in statistics to extract relevant patterns in the data while minimising noise.

- Admittedly, the instances calculated till now are extremely small compared to what is classically intractable today. Hence, currently, it is difficult to make meaningful comparisons between polynomially scaling bond-dimensions and exponentially scaling bond-dimensions. This is because the highest possible bond dimension in an n-qubit register grows as $2^{\lfloor n/2 \rfloor}$ which is 64 for the 12-qubit instance. For such small instances, $n^2 = 144$ and hence it is difficult to even look at if a bond-dimension that scales as $\hat{O}(n^2)$ with system size n gives a good performance. We can, however, make comparisons with a linearly scaling bond-dimension and it can be seen that for such cases, the median relative performance is around 0.65 in the 12-qubit instance.

- Nevertheless, even with these small instances, we can make constant order, $\hat{O}(1)$ inferences, that is inferences on how the problems behave for a constant bond-dimension, by looking at $D_{\max} = 2$. In all the cases, this is the maximum useful approximation possible, and we see that the performance is comparable with $\hat{O}(n)$ scaling of $D_{\max}$ where the bond-dimension limit scales linearly with system size.

- The big question is how these behaviours scale with both system size n and circuit depth p. If we see that similar behaviour persists even for larger instances and circuit depths, then it would call into question the need for a QPU during the training phase of the original QAOA.

8.3 Exact-Cover Training Results

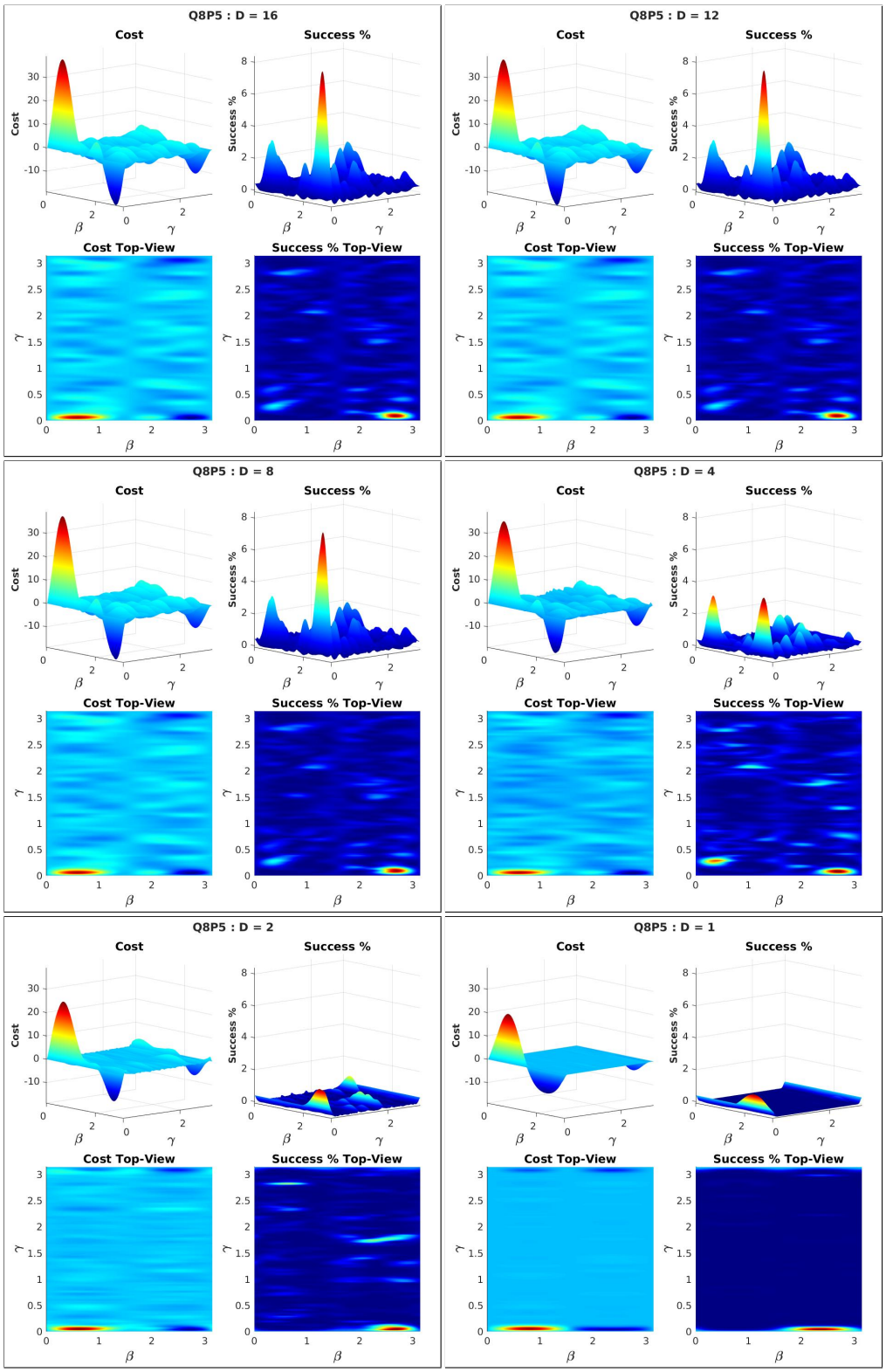


Figure 8.9: An example of the azimuthal and top views of Cost and Solution Landscape of an 8 qubit Exact-Cover calculations over representative Bond dimensions $D = 16,\ 12,\ 8,\ 4,\ 2,$ and 1. For an 8-qubit system, highest possible $D = 2^{\lfloor 8/2 \rfloor} = 16$.

We performed the same set of calculations for all the examples at hand of 8, 15, and 25 qubit instances of Exact-Cover problems sourced from Jeppesen. Figure 8.9 presents a representative calculation result of an 8-qubit instance which performed the worst among all the others with respect to normalised success probability calculations. Both the top and azimuth views of the cost and solution landscapes for $D = 16,\ 12,\ 8,\ 4,\ 2,$ and 1 are shown. Key insights derived from these calculations are listed below:

- As observed for all the max-cut instances, the cost and solution landscapes look eerily similar among all the exact-cover examples too, as it did for the max-cut instances. While in this thesis we use them directly as provided by Jeppesen without concerning ourselves with how they were exactly created (see [39] for details on instance creation), it is safe to assume that this too is a consequence of the behaviour noted by Fernando et al. in [48] and described in section 8.1.

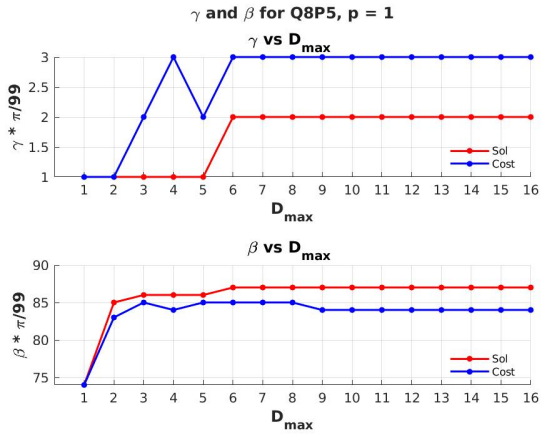
- A striking difference between the landscapes observed in exact-cover and max-cut is how steep the extremas (both maximas and minimas) are in the case of exact-cover instances. The actual cause of why such a difference in landscape structure exists is not known yet. This may mean that the exact-cover minimas are harder to detect and hence harder to train.

- Another surprising feature of the exact-cover instances is the existence of a peak in the solution landscape at the position that corresponds to a maxima in the cost landscape. This phenomenon is more pronounced in the higher qubit count instances, and the reason for this is still unclear. Nevertheless, although it is surprising, it does not violate any of the principles of QAOA, and this could simply be an artefact of the fact mentioned in the discussion below Figure 8.1. In QAOA, we focus on minimising the cost, and not maximising the solution. This is of course for obvious reasons, as one needs to already know the solution to operate over the solution landscape, and for practical applications, we do not already know the solution.

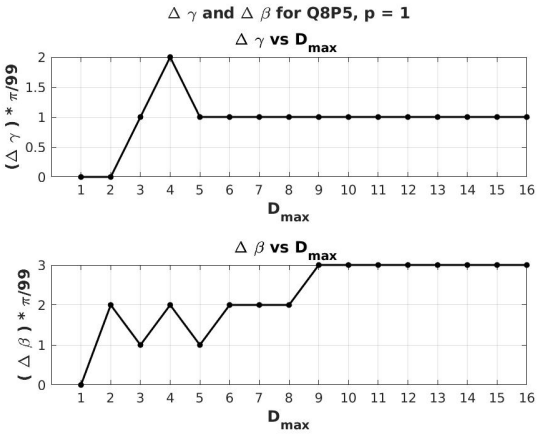
- Another important difference between the two cases is the nature of the landscape for $D_{\max} = 1$. We can observe here that both the cost and solution landscapes retain most of their structure even in the case of a product state with $D_{\max} = 1$. This means that even a product state is able to capture and retain some of the information about the problem at hand. As theorised before, this could be because all the states in the exact-cover instances are not doubly degenerate as they are in the case of max-cut and hence a product state could also hold some information.

- An additional interesting observation about exact-cover landscapes is that the optimal γ value is negligibly small, ranging between $1 \times \pi/99$ and $2 \times \pi/99$. This is also quite in contrast to the max-cut landscapes where γ_{opt} is roughly an order of magnitude higher. Since γ_{opt} is the parameter characterising the

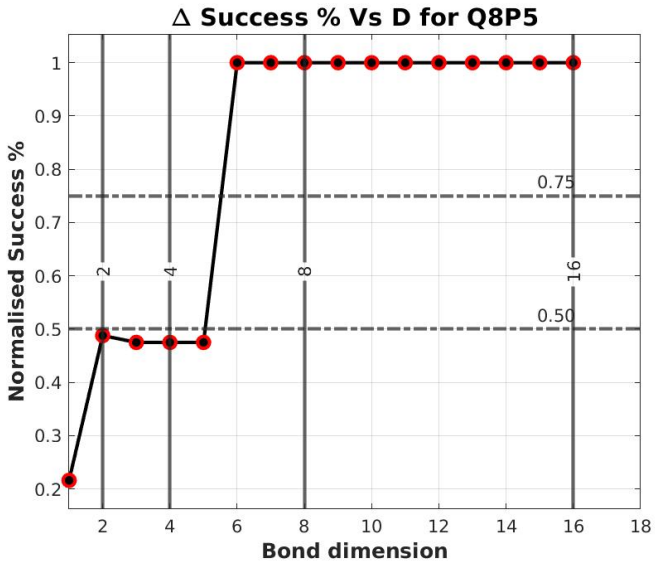
entanglement in the final $|\gamma_{opt}, \beta_{opt}\rangle$ state, this may imply that final state for max-cut holds more entanglement than exact-cover. Intuitively, this makes sense, because all exact-cover energy levels have maximally entangled states.



(a) $\gamma_{opt}(D_{\max}), \beta_{opt}(D_{\max})$ in blue and $\gamma_{max}(D_{\max}), \beta_{max}(D_{\max})$ in red.



(b) $\Delta\gamma(D_{\max})$ and $\Delta\beta(D_{\max})$ in top and bottom panels respectively.

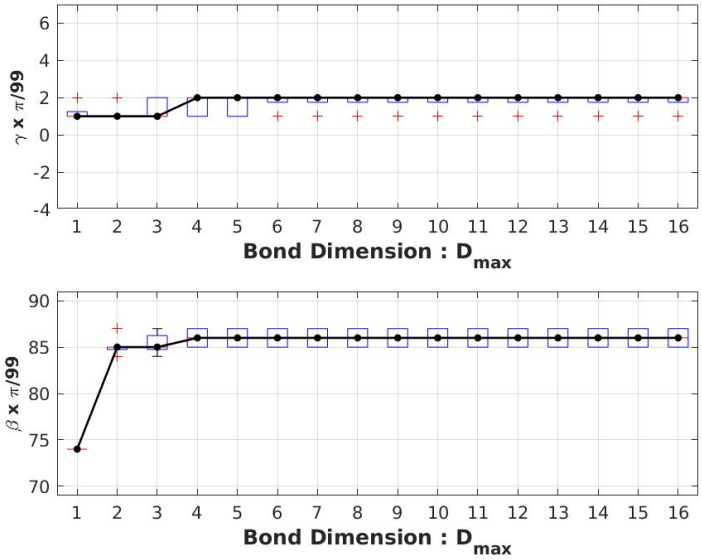


(c) Normalised Success rate obtained using the angles $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ w.r.t $D_{\max}$.

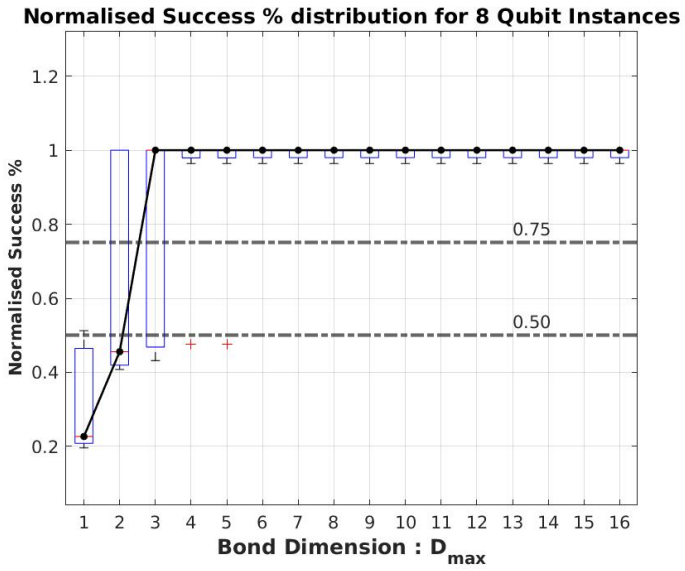
Figure 8.10: Calculated properties of instance Q8P5 for $D_{\max} = 1$ to 16.

All the same calculations mentioned under section 8.2 discussing the max-cut instances were also performed for the exact-cover examples. Figure 8.10 provide a snap-shot of this performed on the landscape depicted in Figure 8.9. Here again, $\Delta\gamma(D_{\max})$ and $\Delta\beta(D_{\max})$ is the difference between the global minima point of the cost landscape $= (\gamma_{opt}(D_{\max}), \beta_{opt}(D_{\max}))$ and the global maxima point in the solution landscape given by $\gamma_{max}(D_{\max}), \beta_{max}(D_{\max})$. We also note that although the

degree to which the angles $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ shift as a function of $D_{\max}$ is much less in the case of these exact-cover instances, the corresponding change is relative success rate is profound. For example, in Figure 8.10a, when γ changes by one grid point from $D_{\max} = 5$ onward, the corresponding change in Figure 8.10c is by 50%. This is due to how steep the peaks and minimas are in the original landscape. Also, an additional shift in β is observed commonly for $D_{\max} = 1$, which further takes the relative success rate as seen in Figure 8.10c. Below, we present the present box plots providing insights into the average behaviour of 8, and 15 qubit exact-cover instances.

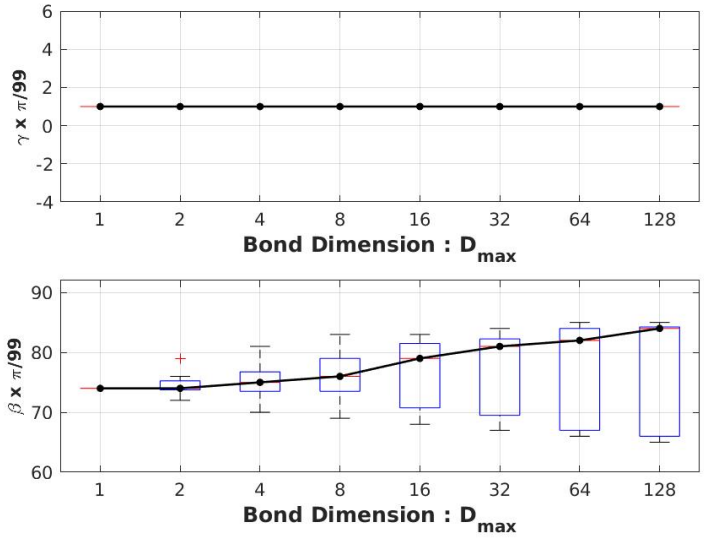


(a) $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ in top and bottom panels respectively.

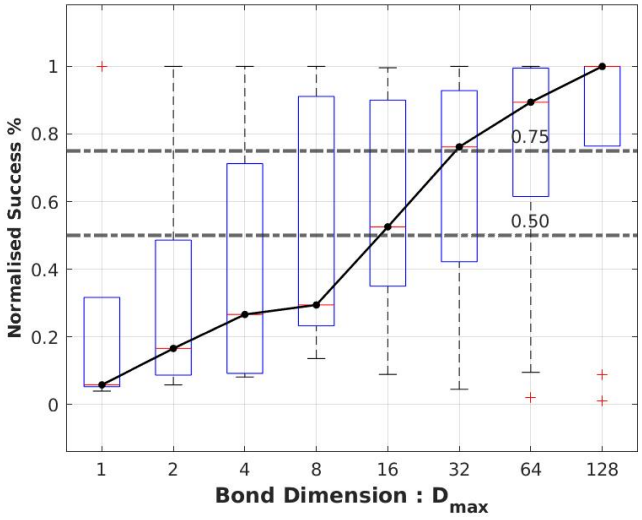


(b) Normalised Success rate of 8-qubit instances as a function of $D_{\max}$.

Figure 8.11: Calculated statistical behaviour of 8-qubit exact-cover instances.



(a) $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ in top and bottom panels respectively.



(b) Normalised Success rate of 15-qubit instances as a function of $D_{\max}$.

Figure 8.12: Calculated statistical behaviour of 15-qubit exact-cover instances.

Figures Figure 8.11a and Figure 8.12a represent statistical box-plots of the optimum angle evolution $\gamma_{opt}(D_{\max})$ and $\beta_{opt}(D_{\max})$ for 8 and 15 qubit instances respectively. As previously mentioned, we see a very tight bound in $\gamma_{opt}(D_{\max})$ in both cases. However, the data in Figure 8.12 was quite surprising in two ways.

1. Firstly we see an extremely tight bound in $\gamma_{opt}(D_{\max})$ where the angles show zero deviation with decreasing $D_{\max}$. This is surprising because γ_{opt} is the parameter controlling entanglement in the final $|\gamma_{opt}, \beta_{opt}\rangle$ state. So since we are reducing the amount of entanglement in the state by reducing $D_{\max}$, one would have intuitively expected γ to show changes.

2. Second surprising point is the more prominent shift in β_{opt} with decreasing $D_{\max}$ all converging onto the same tight bound point for $D_{\max} = 1$.

Hence, it is also this drift in β_{opt} that is leading to a decreasing normalised success rate for the 15-qubit instances as shown in Figure 8.12b. However, it is still promising to note that the median normalised success rate is around 50% when $D_{\max} = 16$, which is in the range of a bond-dimension linearly increasing with system size. Data on the optimum angle evolution with $D_{\max}$ for the 25 qubit instances are still being calculated, and hence not available for presentation yet. Nevertheless, from preliminary observations, we note a much tighter bound over both γ_{opt} and β_{opt} with decreasing $D_{\max}$ in them.

One big caveat we have to make here before concluding this chapter is that till now, we have only shown that the landscape largely retains its structure even on decreasing $D_{\max}$ for circuit depth $p = 1$. Further calculations are required to investigate if this behaviour persists for larger circuit depths, $p < 1$, as well. If we see that this behaviour persists even for larger p, then even a normalised success rate performance of 20% could be acceptable. This is because in that case, we only need to increase the required circuit depth from p to $5 \times p$, to make the algorithm work.

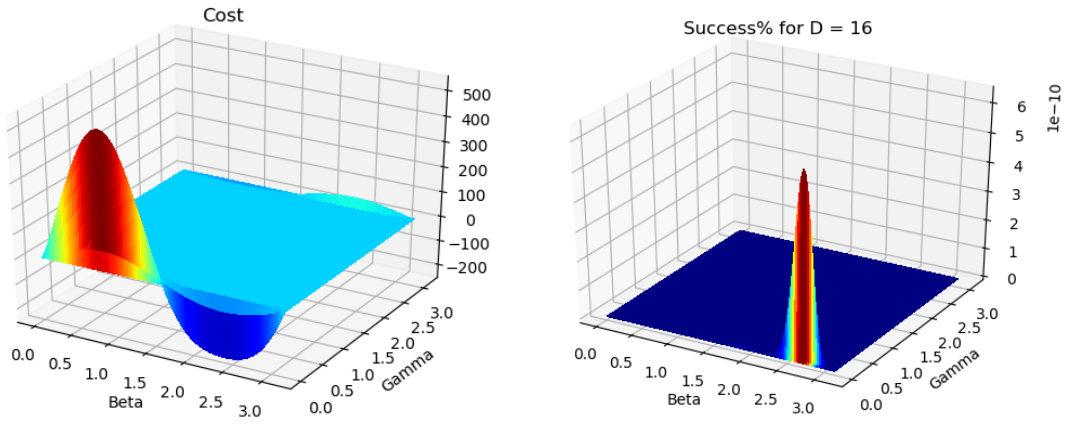
Another very interesting avenue of exploration would be to study if one could use the angles sourced from training phases using the truncated cost landscapes for larger p and smaller $D_{\max}$ in the QIOA algorithm described in chapter 7. The next chapter, chapter 9, discusses the extraction of the ground state $|gs\rangle$ even from truncated $|\gamma_{opt}, \beta_{opt}\rangle$ states, using the optimum parameters obtained from the original vanilla-QAOA training.

9

QIOA Results and Discussions

While the last chapter (chapter 8) presented results hinting at the possibility of classically calculating the optimum parameters required in the original QAOA, in this chapter, we look at the possibility of classically extracting the solutions from approximated QAOA states, $|\gamma_{opt}, \beta_{opt}\rangle$. Today, there exist multiple flavours of the originally proposed vanilla-QAOA [2], such as Tree-QAOA [32], proposed by the Data lab of Volkswagen group, where also the possibility of training vanilla-QAOA classically for max-cut on regular graphs was suggested, and QAOA variations such as Grover-QAOA [34] etc. which modify vanilla-QAOA by proposing different mixing gates. To the best of our knowledge, these, and other currently existing flavours of QAOA, all agree that one needs access to a QPU for sampling the state $|\gamma_{opt}, \beta_{opt}\rangle$ to find the solution $|gs\rangle$. This chapter, which presents our results on QIOA, goes over if indeed there is a need for a QPU in all cases to extract the solution $|gs\rangle$. Towards the end, there will also be a discussion on the trainability of QIOA, and ways to potentially make it better.

9.1 Inspiration



(a) Cost landscape $C(\gamma, \beta, D_{\max} = 16)$.

(b) Solution landscape $S(\gamma, \beta, D_{\max} = 16)$.

Figure 9.1: Calculated data of a 50-qubit Exact-Cover instance for $D_{\max} = 16$

Figure 9.1 depicts the cost and solution landscape calculated for a 50-qubit exact-cover instance for a maximum bond-dimension capped at $D_{\max} = 16$. This is the largest system size we had looked at in this project, and it was interesting to see that the cost landscape $C(\gamma, \beta, D_{\max})$ looks very similar to the other Exact-cover calculations presented in section 8.3. The maximum bond-dimension for a non-truncated 50-qubit state is $2^{\lfloor 50/2 \rfloor} = 33554432$, which is more than 33.55 million. Exact simulations of 50-qubit systems teeth at what is feasible for even most advanced high-performance computing systems, as the matrices involved in such simulations would be 33.55 million $\times$ 33.55 million. Hence, it was highly encouraging to see that comparable results could be obtained using matrix product states which only involved 16×16 matrices trivially handled by today's laptops. However, do keep in mind that we have only explored this for circuit depth $p = 1$, and need further studies to test if the results described in chapter 8 hold for $p > 1$ as well. We also found the calculated solution landscape to be equally intriguing. A careful study of Figure 9.1b reveals that the scale on the Z-axis reads 10^{-10}. Nevertheless, it was surprising to note that even after making approximations worth 6 orders of magnitude to the bond-dimensions, the relative form of the solution landscape remained intact. This was one of the first signs that although reducing the bond-dimension leads to the $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state being non-normalised and thus a reduction of the weights of individual basis states within this full state, it still looked like their relative magnitudes were preserved. To us this was a strong hint that it could be possible to extract useful information about the ground state $|gs\rangle$ even from a highly truncated $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state with $D_{\max} \lll 2^{\lfloor n/2 \rfloor}$.

9.2 Hamming Distance

The QIOA method described in chapter 7, takes the cost Hamiltonian H_C, maximum bond-dimension $D_{\max}$, and the $2p$ optimum parameters $\vec{\gamma_{opt}}, \vec{\beta_{opt}}$ as input, and returns an n-bit bitstring as output. Now, we need a way to compare this output string with the original solution string, to see how well QIOA performed. We chose to use the Hamming distance to compare between two strings as it is one of the simplest and most straight-forward protocols to implement [51]. The Hamming distance is the total number of mismatched characters between two strings of equal length. A distance of 0 would hence mean that both the strings are the same, which in our case implies that the predicted bitstring is indeed the real solution. For example, the Hamming distance between bitstrings $'000'$, and $'001'$, $'110'$, and $'111'$, are equal to 1, 2, and 3 respectively. However, a problem with the Hamming distance is that multiple pairs of bitstrings can have the same distance among them. For instance, the Hamming distances between $'000'$ and $'101'$, and $'000'$ and $'110'$, are both = 2. This makes it almost impossible to pinpoint state pairs just from knowing the Hamming distance between them.

In hindsight, an obvious way to get over this issue would have been to look at the approximation ratio, that is the ratio of the energies corresponding to the predicted state and the actual true ground state [52]. An approximation ratio of 1 would

mean that the predicted bitstring is indeed the solution. Moreover, further the ratio is from one, poorer the predicted bitstring is in solving the problem. At the moment though, we have only employed the Hamming distance to compare between bitstrings, and the approximation ratio implementation is currently listed an easily implementable future work.

9.3 Black-box Picture of QIOA

Equation 10 in the original vanilla-QAOA paper [2] explicitly state that in the infinite limit, when $p \to \infty$, the approximation ratio of the $|\vec{\gamma_{opt}}, \vec{\beta_{opt}}\rangle$ state would be equal to 1. Which is to say that at this limit, $|\vec{\gamma_{opt}}, \vec{\beta_{opt}}\rangle$ state would exactly be the ground state $|gs\rangle$ of the problem Hamiltonian H_C. For all practical purposes, however, infinities do not exist, and as long as the ground state $|gs\rangle$ holds the highest weight within the full $|\vec{\gamma_{opt}}, \vec{\beta_{opt}}\rangle$ state, one can easily sample form this to obtain the solution. Hence, one way to look at vanilla-QAOA is as a black box with a tunable parameter p that provides a particular state distribution when measured in the computational basis. Also, with increasing p the state distribution changes such that the weights C_{gs} of the ground state $|gs\rangle$ increases, and all the other weights C_i decreases. Refer Equation 7.1 for a refresher on this. Do note that in this black-box perspective of vanilla-QAOA where the focus is on tunable parameter p, the need for calculating the $2p$ optimum angles $\gamma_1, \gamma_2, \gamma_3, \ldots \gamma_p$ and $\beta_1, \beta_2, \beta_3, \ldots \beta_p$, and the role of the cost Hamiltonian H_C are implicitly defined. Which is to say that although their roles are not explicitly mentioned in this picture, they are still extremely vital.

The question we are trying to answer with QIOA is if the same nature persists even for smaller bond-dimensions $D_{\max}$. That is, if the weight C_{gs} of the ground state $|gs\rangle$ increases with circuit depth p even in a truncated state $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state even for $D_{\max} \ll 2^{\lfloor n/2 \rfloor}$. By doing so, we introduce another tunable parameter $D_{\max}$ which is the maximum allowed bond-dimension in the full matrix product state representation of all $|\gamma, \beta\rangle^{D_{\max}}$ states throughout their evolution. As a reminder, it is the exponential growth of an un-checked $D_{\max}$, that leads to classical intractability of any full $|\gamma, \beta\rangle$ state. A persistence of this behaviour of $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ states with increasing circuit depth p even for small $D_{\max}$ would imply that using the highest possible, exponentially scaling, bond dimension by setting $D_{\max} = 2^{\lfloor n/2 \rfloor}$ may be an unnecessary overkill.

Revert to chapter 7 for a refresher on how we extract the state with the highest weight from a $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state. There, we have defined two methods used for state extraction, the Reduced Density Matrix method (subsection 7.4.1), hereafter referred to as the DM method, and the Projected Reduced Density Matrix method (subsection 7.4.2), henceforth referred to as PRDM method. Hence, QIOA can now be treated as another black-box, similar to QAOA, but with two tunable parameters $(p, D_{\max})$, instead of the single tunable parameter p in QAOA. Here, each pair of entry would produce an n-bit bitstring as output for both DM and PRDM meth-

ods, which can be compared with the original solution using the ideas presented in section 9.2.

9.4 Plot Definitions

This section goes over the different tools used to not only study the performance of QIOA but also gain insights to its inner workings and test our hypothesises.

9.4.1 Hamming Maps

Combining insights from the above two sections, one way to see how well QIOA performs is by plotting a heat-map of Hamming distances for each $(p, D_{\max})$ pair, hereafter called as Hamming maps. Furthermore, since there may-be multiple equally likely solution states, we choose to represent the minimum Hamming distance between all possible (predicted state, solution- state) pairs in the Hamming maps. This is because the goal of solving the max-cut problem is to find at least one maximum cut of the graph, and not provide all possible maximum cuts. Such Hamming maps would not only help gauge the efficacy of QIOA but also provide insights on its underlying principles. They would also serve as an excellent tool to compare between DM and PRDM methods. Also, as stated in chapter 6, section 6.2, discussing the training phase of QAOA, the $2p$ angles used in this thesis for the Hamming maps have all been sourced by employing a combination of Nelder Mead [45], and the heuristic developed by Lukin et al. [46]. These calculations, were performed on non-truncated states using a vector matrix implementation of vanilla-QAOA and hence have $D_{\max} = 2^{\lfloor n/2 \rfloor}$. For the Max-cut instances, we performed our calculations up to $p = 100$, while for the exact-cover problems, we have angles up to $p = 20$ for the 8 and 15 qubit instances and up to $p = 11$ for the 25 qubit examples.

9.4.2 Confidence Plots

Next, in QIOA, to infer the state of the k^{th} bit in the full n-bit bitstring, we always compare between two numbers. In the case of the DM method, we compare ρ^k_{00} with ρ^k_{11} , the 00^{th} and 11^{th} elements in the reduced density matrix of qubit k in the full n-qubit register. We assign 0 to qubit k if $\rho^k_{00} \geq \rho^k_{11}$ and 1 otherwise. And in the case of the PRDM method, we compare between $\rho^k_{proj}[0,0]$ and $\rho^k_{proj}[1,1]$ using the same set of arguments as above. The only difference between DM and PRDM being that in DM, ρ^k is the full reduced density matrix with all the other qubits traced out, while in PRDM, ρ^k_{proj} is the reduced density matrix of qubit k with the previous $k-1$ qubits projected onto their respective calculated states, and the other $n-k$ qubits fully traced out. An interesting question here is to ask how far apart the two numbers corresponding to the 00^{th} and 11^{th} elements were in both cases, and so in a sense, how confidently did QIOA make this choice between 0 and 1 for the k^{th} qubit. We now define this quantity as $\text{Conf}(p, D_{\max}, k)^{\text{DM/PRDM}}$. Also, note that $\text{Conf}(p, D_{\max}, k)^{\text{DM/PRDM}}$ is something we have introduced in this thesis for ease of reference. That is,

$$\begin{aligned} \text{Conf}(p, D_{\max}, k)^{\text{PRDM}} &= \text{abs}(\ \rho^k_{proj}[0,0]\ -\ \rho^k_{proj}[1,1]\ ) \\ \text{Conf}(p, D_{\max}, k)^{\text{DM}} &= \text{abs}(\ \rho^k_{00}\ -\ \rho^k_{11}\ ). \end{aligned}$$

Finally, we also average over all the n qubits in the register to find a value that is only a function of p and $D_{\max}$ for both DM and PRDM methods.

$$\text{Conf}(p, D_{\max})^{\text{DM/PRDM}} = \frac{1}{n}\left(\sum_{k=1}^{n} \text{Conf}(p, D_{\max}, k)^{\text{DM/PRDM}}\right) \tag{9.1}$$

Both QAOA and QIOA use the uniform superposition state $|+\rangle^{\otimes n}$ as their initial state. At this stage, all of ρ^k_{00} , ρ^k_{11} , $\rho^k_{proj}[0,0]$, and $\rho^k_{proj}[1,1]$ are all equal to 0.5 for all the k qubits in the register. Hence, for circuit depth $p = 0$, nothing has yet been done to the initial state and we just have the superposition product state with $D_{\max} = 1$. In this case, $\text{Conf}(p = 0, D_{\max} = 1, k)^{\text{DM/PRDM}} = 0 \quad \forall\ k \in \{1, 2, \ldots, n\}$, both for DM and PRDM methods. As we keep increasing the circuit depth p, $\text{Conf}(p, D_{\max}, k)^{\text{DM/PRDM}}$ increases. In this regard, usefully extractable information from this confidence measure $\text{Conf}(p, D_{\max})^{\text{DM/PRDM}}$ are twofold:

1. Firstly, it acts as a measure of how much further the end state $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$, for a particular circuit depth p and bond-dimension caped at $D_{\max}$, has evolved away from the superposition state $|+\rangle^{\otimes n}$. For instance, for $p = 1$ and $D_{\max} = 1$, which is the most approximated state with the least amount of possible evolution, we observed that the $\text{Conf}(p, D_{\max})^{\text{DM/PRDM}}$ was on the order of 10^{-10} which meant that there was almost no change made to the $|+\rangle^{\otimes n}$ state. Nevertheless, it was still interesting to note that there were instances where even that was enough to make the correct predictions. Now with both increasing p, and $D_{\max}$, $\text{Conf}(p, D_{\max})^{\text{DM/PRDM}}$ would also increase.

2. Next, one can compare the confidence maps of both the DM and PRDM methods to make inferences on which one works better for what type of problems. For instance, we know that for max-cut problems, the ground state is an equal superposition of at least two maximally entangled states where one state is the bit-flipped version of the other (chapter 4, subsection 4.2.1). In such a case, as p increases, the weights of all possible ground states increases. This makes both 0 and 1 equally likely for the k^{th} qubit if we only calculate the reduced density matrix. This inference was made in retrospect, after seeing that the DM method performs poorly for max-cut instances, and where ever it performed poorly, it had a very small confidence value. The PRDM method avoids this problem by only looking at a smaller subset of 2^{n-k+1} states by projecting the previous $k-1$ states while deciding the states of the k^{th} qubit.

The following sections will now discuss these Hamming maps and Confidence plots for the same Max-Cut and Exact-Cover instances, whose training results for $p = 1$ was discussed in chapter 8. For all the 10 examples of 8, 9, 10, 11, and 12 qubit

Max-Cut instances, we have calculated over the entire bond-dimension range ($1 \leq D_{\max} \leq 2^{\lfloor n/2 \rfloor}$) and for circuit depth $1 \leq p \leq 100$. In the case of Exact-Cover instances, we have calculated over the full range of bond-dimensions for examples of 8 and 15 qubits with p ranging from 1 to 20, while for the 25 qubit examples, $1 \leq D_{\max} \leq 55$) and $1 \leq p \leq 11$.

9.5 Max-Cut QIOA Results

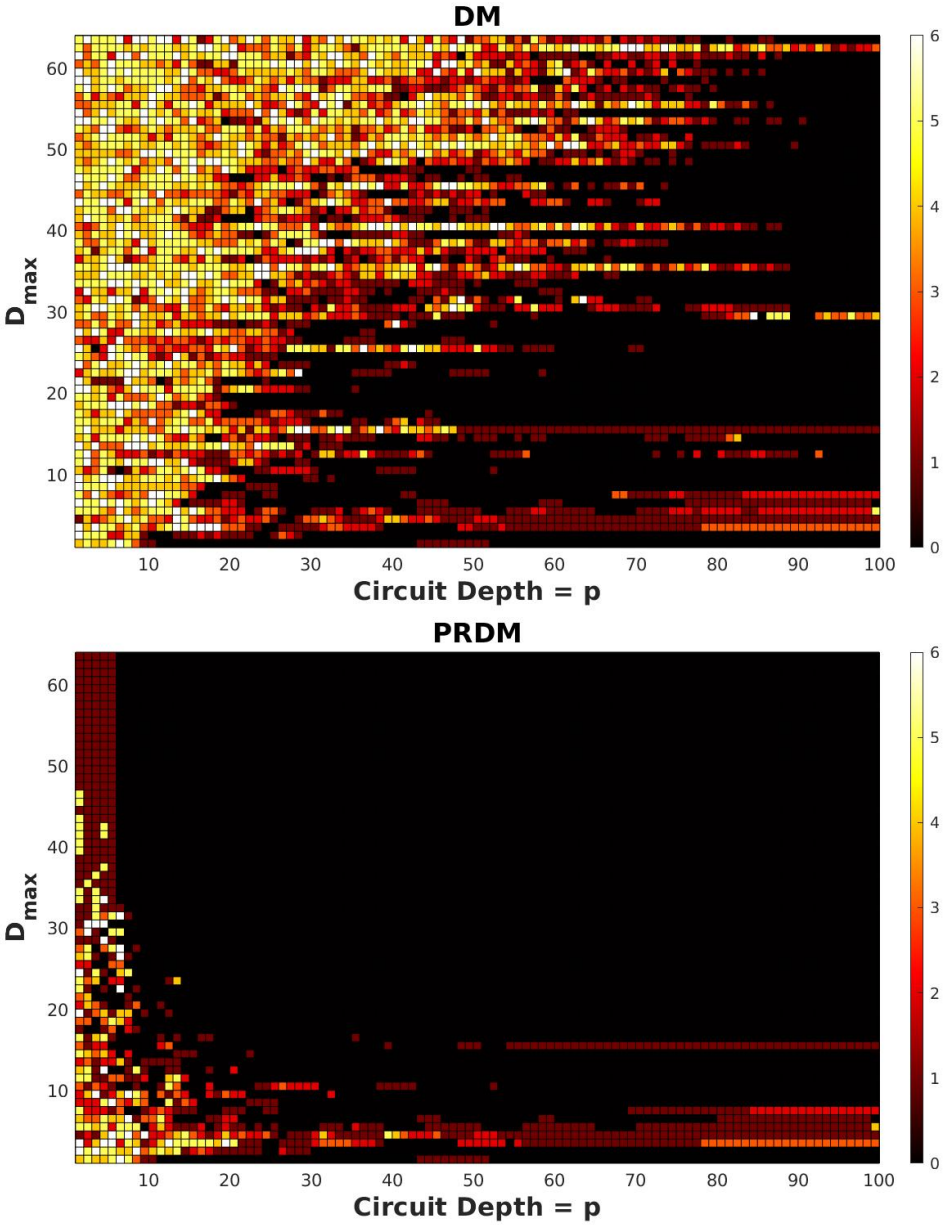


Figure 9.2: Hamming Maps of both DM and PRDM methods implemented on a 12 Qubit Max-Cut instance, Q12R5. Here, $1 \leq D_{\max} \leq 2^{\lfloor 12/2 \rfloor} = 64$ and $1 \leq p \leq 100$. A black pixel corresponds with QIOA predicting the correct solution for that ($p, D_{\max}$) pair.

Figure 9.2 depicts the Hamming map of a 12-qubit Max-Cut instance (Q12R5). The upper and lower panels present the performances of the DM and PRDM methods respectively. Each pixel in the plot depicts the Hamming distance between the actual ground state of the cost Hamiltonian H_C, and the predicted bitstring calculated using the corresponding $(p, D_{\max})$ pair as inputs. The colour-bar to the right of these images depict the scale used and assigns a Hamming distance of 0 with the colour black. Hence, in all the areas in the Hamming map where the pixels are black, we can conclude that QIOA had successfully predicted the correct solution for the problem. We also see that in the worst case, QIOA predicts a bitstring which has a Hamming distance of 6 with respect to the actual solution. Furthermore, the left-bottom corner of these Hamming maps depicts the region that is classically tractable and as one progresses away from the origin along either axes, the more classically impractical it gets, as both or either of p, $D_{\max}$ increases.

While Figure 9.2 is presented as a representative example of QIOA performance, it shares a lot of features common with all the other 49 different Max-Cut instances. Some of those common features are listed below:

1. Possibly the most interesting insight from Figure 9.2 is to see that using the PRDM method of QIOA, there are regions in the Hamming map near the leftmost-bottom slices, with black pixels. For instance, the entire range of $(D_{\max} = 1)$ and $(11 \leq p \leq 43)$ and then again for $(52 \leq p \leq 100)$. This was quite unexpected because in Figure 8.1, we saw that for $D_{\max} = 1$ and $p = 1$, the cost landscape was almost completely flat and devoid of any structure. We had then theorised this lack of topology to be because of the maximally entangled nature of max-cut states which could not be captured by a fully truncated $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state with $D_{\max} = 1$, as it is a product state. Following this logic, we expected this behaviour to persist even for larger p, but this result shows that it is possible even for non-entangled product states to capture the same solutions as vanilla-QAOA.

2. Moreover, in vanilla-QAOA, there exists no tunable entanglement. Hence, one would always be operating on the uppermost slice of the $D_{\max}$ axis on these Hamming maps, where $D_{\max} = 2^{\lfloor n/2 \rfloor}$. However, Figure 9.2, and all the 49 other such Hamming maps explicitly show that operating at this maximum entanglement regime is very likely a huge overkill.

   As it currently stands though, QIOA is still a heuristic algorithm. Which is to say that given a $(p, D_{\max})$ pair, it is currently not known a priory if the bitstring QIOA produces is the solution, or not. This, however, is not a major limitation, because, in the left-bottom regime, QIOA is classically tractable. Hence, one only need to repeatedly run this method for different classically simulable $(p, D_{\max})$ pairs, till we land on a solution. Such a strategy works because of the very nature of NP problems, where although it is very hard to find the solutions of an NP problem, given a prediction, it is possible to verify if this given prediction is indeed the solution in polynomial time.

3. As hinted in point 2 of from the list in subsection 9.4.2, we see that for Max-Cut, the performance of the DM method gets worse with increasing $D_{\max}$. Especially after crossing the $D_{\max} = \frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$.

   This sudden change of behaviour after crossing this threshold has been seen multiple times before. For instance, in all the normalised success probability plots such as Figure 8.4b to Figure 8.8b, we see a minimal change on going from $D_{\max} = 2^{\lfloor n/2 \rfloor}$ to $D_{\max} = \frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$. A same behaviour also persists in the plots depicting the change in optimum angles. Furthermore, although the peaks in cost and solution landscape decrease with decreasing, $D_{\max}$, we again see a minimal change here too when going from $D_{\max} = 2^{\lfloor n/2 \rfloor}$ to $D_{\max} = \frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$. All of these suggest that the amount of approximations made when changing the bond-dimension limit from $2^{\lfloor n/2 \rfloor}$ to $\frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$ is negligible.

   This implies that for $\frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right) \leq D_{\max} \leq 2^{\lfloor n/2 \rfloor}$, since there are not much approximations made onto the full $|\gamma_{opt}, \beta_{opt}\rangle$ state, and since max-cut solutions are maximally entangled, it makes perfect sense that both $'0'$ and $'1'$ are almost equally likely choices for each qubit (see subsection 4.2.1 for details).

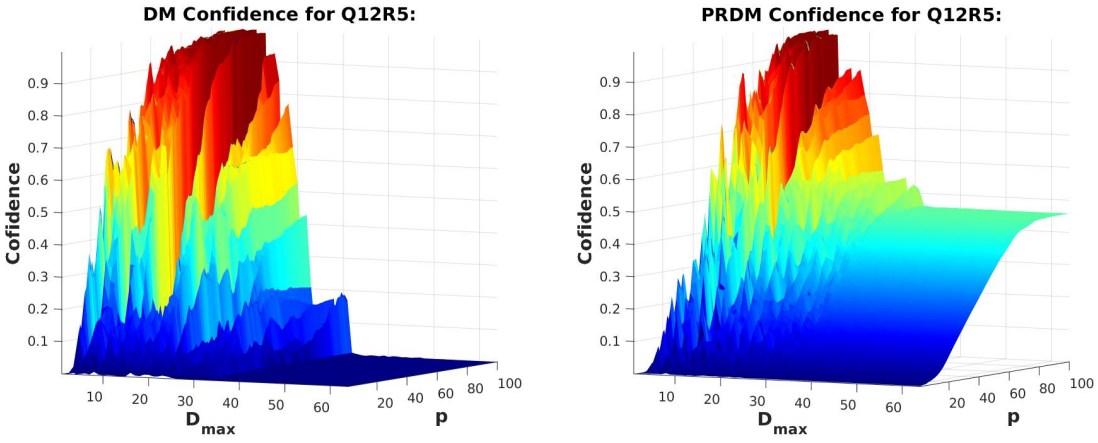
4. Another possible reason for the much poorer performance of the DM method could be that there could exist multiple maximally entangled pairs as equally likely solutions. This could cause a problem because there is nothing in the reduced density matrix calculation framework that helps the predicted bit-string to lock onto a particular state from this set of all possible solutions. The PRDM method overcomes this problem by projecting the previous $k-1$ qubits to their respective predicted states while making the prediction for the k^{th} qubit, hence (possibly) locking on to one of the feasible solutions.

5. These claims, (claims made in point 3 and point 4), are further supported by the observation that both DM method and PRDM method show a comparable performance for Exact-Cover problems where the problems are so designed that there are no degeneracies involved. This shall be made more clear in the discussions following the Exact-Cover QIOA results in section 9.6.

6. Next, the results described in chapter 8, on the evolution of the optimum parameters $\gamma_{opt}, \beta_{opt}$ for $p = 1$ with $D_{\max}$ catalog a gradual decrease in the angles with decreasing $D_{\max}$. We have also presented results there hinting that the global maxima of the solution landscape also shifts corresponding with this shift in γ_{opt}, and β_{opt}. As stated earlier in subsection 9.4.1, the $2p$ optimum angles however, were calculated for QAOA with non-truncated bond-dimension $D_{\max} = 2^{\lfloor n/2 \rfloor}$. Hence, we strongly suspect this to be one of the reasons of poor performance of QIOA for smaller bond-dimensions $D_{\max}$ and circuit depths p. This also explains why QIOA works really well for larger $D_{\max}$ as there is relatively negligible drift in the optimum angles in this range. This point will be further scrutinised in section 9.8 discussing the trainability of QIOA in more detail.

7. Another noteworthy point common to all Hamming maps is the existence of domains within the map with the same Hamming distance. For instance, $48 \leq D_{\max} \leq 64$) and $1 \leq p \leq 5$ in Figure 9.2, PRDM panel. As explained in section 9.2, it is difficult to know if all these predicted bitstrings are the same, although they have the same Hamming distances from just the Hamming maps. Nevertheless, while performing these calculations we had observed that in most cases the bitstrings belonging to a particular domain are the same. Knowing more about the nature of when QIOA fails could help to make the algorithm better. This was one place where a similar plot of approximation ratios could have helped.

Figure 9.3 presents representative confidence plots of both the DM (Figure 9.3a) and PRDM (Figure 9.3b) methods. These plots are of the same 12-qubit max-cut instance Q12R5, featured in Figure 9.2, and they further amplify the points raised above. Figure 9.3 helps us to picturise the contrast between the inner-workings of DM and PRDM methods. The following line of reasoning uses these confidence maps to connect all the dots discussed above.



(a) Conf$(p, D_{\max})^{\mathrm{DM}}$. **(b)** PRDM : Conf$(p, D_{\max})^{\mathrm{PRDM}}$.

Figure 9.3: Calculated Confidence plots for Q12R5.

1. From basic Quantum mechanics, we know that the trace of any reduced density matrix is equal to 1. That is, $\rho^k_{00} + \rho^k_{11} = 1$ for any single qubit reduced density matrix.

2. Also, Conf$(p, D_{\max})^{\mathrm{DM}}$ calculates the difference between ρ^k_{00} and ρ^k_{11} averaged over all qubits (Equation 9.1) for each $(p, D_{\max})$ pair.

3. Now, in Figure 9.3a we clearly see that the Conf$(p, D_{\max})^{\mathrm{DM}} \approx 0$ for all values of p and $D_{\max} > \frac{1}{2}\left(2^{\lfloor 12/2 \rfloor}\right) = 32$. Which means that $\rho^k_{00} + \rho^k_{11} = 1$, and $|\rho^k_{00} - \rho^k_{11}| \approx 0$. Implying that $\rho^k_{00} \approx \rho^k_{11} \approx 0.5$, which is what one would expect from the density matrix of a maximally entangled ground state.

4. Furthermore, we already know all the possible solutions of all the instances and hence know that Q12R5 only has a single pair of maximally entangled states as the ground state. This is also beautifully reflected in Figure 9.3b, where we see that with increasing p and $D_{\max} > \frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$, $\text{Conf}(p, D_{\max})^{\text{PRDM}}$ saturates to 0.5. Moreover, it was observed that where there are multiple pairs of solutions involved, the value $\text{Conf}(p, D_{\max})^{\text{PRDM}}$ saturates to also decreases accordingly. These observations support our hypothesis that the PRDM method locks onto one of the possible solutions hence avoiding confusions, exceptionally well.

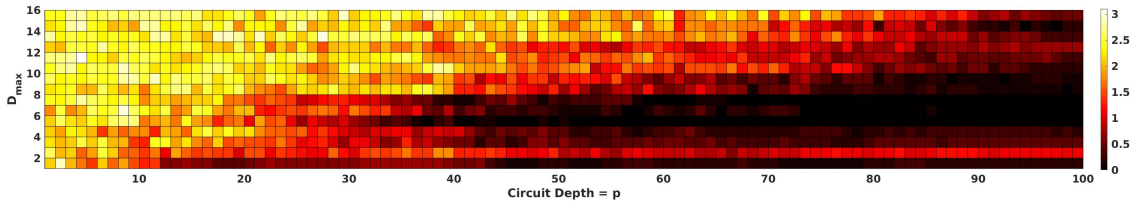
Thus, putting all these pieces together, we can reasonably conclude that almost little approximations are made in the range $\frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right) \leq D_{\max} \leq 2^{\lfloor n/2 \rfloor}$. However, reducing the highest possible $D_{\max}$ by half still doesn't remove the problem of exponential scaling, and is not enough. More interesting region is towards the beginning of $D_{\max}$ axis when $D_{\max} \ll \frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$. The exact reasoning of why QIOA works in this regime also is still unclear. Nevertheless, two interesting observations characteristic to all max-cut instances in about this region are:

1. The existence of alternating ridges of low and high confidence perpendicular to the $D_{\max}$ axis for $D_{\max} < \frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$ in both confidence plots.

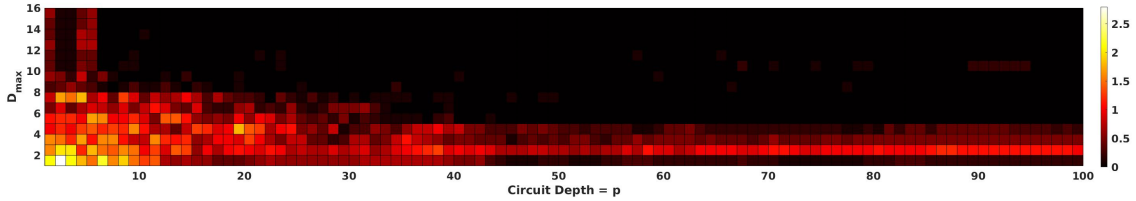
2. The DM method also works considerably better in this first half of the $D_{\max}$ axis compared to the second half.

From the above discussion, we could hypothesise whenever $\text{Conf}(p, D_{\max})^{\text{DM/PRDM}}$ values are high, it means that the algorithm had fewer choices to choose from, and hence, could make more confident predictions. The existence of alternating ridges perpendicular to $D_{\max}$ axis could suggest that certain truncations of the bond-dimension preferentially removes one of the maximally entangled pairs from the $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state thus making it easier for the algorithm to make more confident predictions. A similar trend was observed in the normalised success % plots (Figure 8.4b to Figure 8.8b) where we saw data hinting that bond-dimensions $D_{\max} = 3$ and $D_{\max} = 1$ were subject to more approximations than the rest. In both cases, we believe these be a feature of the doubly degenerate nature of max-cut states. As will be explained in section 9.6, these features are not present in the case of exact-cover instances where there is no degeneracy, further bolstering our claims. One way to verify this hypothesis would be to perform full state tomographies at each bond-dimension $D_{\max}$.

As in chapter 8, to get a better picture of the overall performance of QIOA on max-cut, we have also calculated Hamming maps and confidence plots of both DM and PRDM methods averaged over different 8, 9, 10, 11, and 12 max-cut instances.

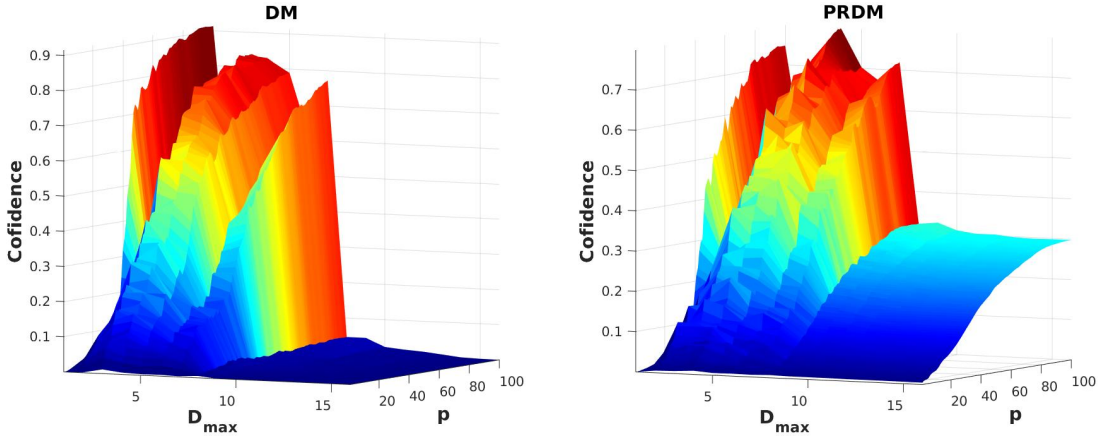


(a) DM method



(b) PRDM method

Figure 9.4: Average Hamming maps of 8-qubit Max-Cut instances



(a) Average $\text{Conf}(p, D_{\max})^{\text{DM}}$.

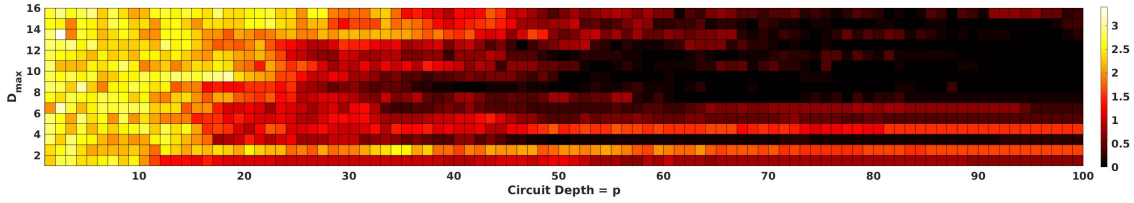
(b) Average $\text{Conf}(p, D_{\max})^{\text{PRDM}}$.

Figure 9.5: Average Confidence Plots for 8 qubit Max-Cut instances.

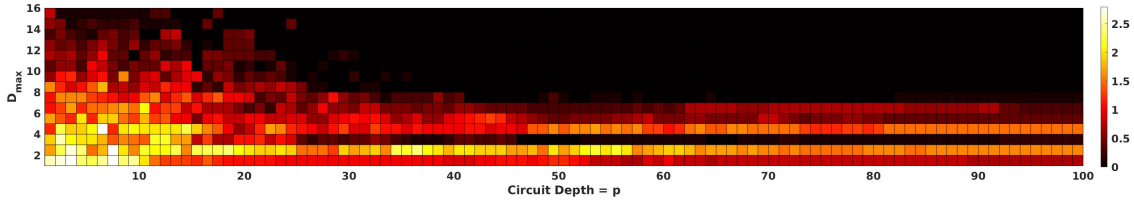
Figure 9.4 presents an average Hamming map where the hamming distance produces by a $(p, D_{\max})$ pair is the average Hamming distance averaged over all the 10 8-qubit Max-Cut instances. Here too we can observe the stark contrast between the performance of DM and PRDM methods that has been described above. Next, Figure 9.5 presents average $\text{Conf}(p, D_{\max})^{\text{DM}}$ (Figure 9.5a) and average $\text{Conf}(p, D_{\max})^{\text{PRDM}}$ (Figure 9.5b) averaged over all available 8-qubit instances. Here too we see the presence of ridges perpendicular to the $D_{\max}$ axis, and that $\text{Conf}(p, D_{\max})^{\text{DM}} \approx 0$ for $D_{\max} > \frac{1}{2}\left(2^{\lfloor 8/2 \rfloor}\right) = 8$, as explained above.

The inherent randomness associated with heuristic methods such as QIOA on if a particular $(p, D_{\max})$ pair can produce a solution or not, has blurred regions in the left-most bottom corner of the plot considerably, making it look like QIOA on an

average does not work for $D_{\max} < \frac{1}{2}\left(2^{\lfloor 8/2 \rfloor}\right) = 8$. However, this is not true and is a consequence of the fact that different $(p, D_{\max})$ pair work for different instances. One has to apply QIOA onto the respective problems to solve them.

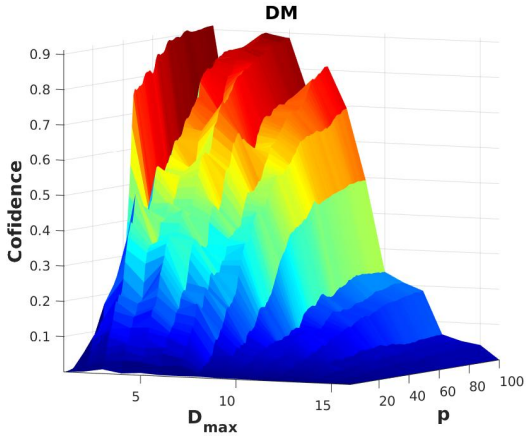


(a) DM method

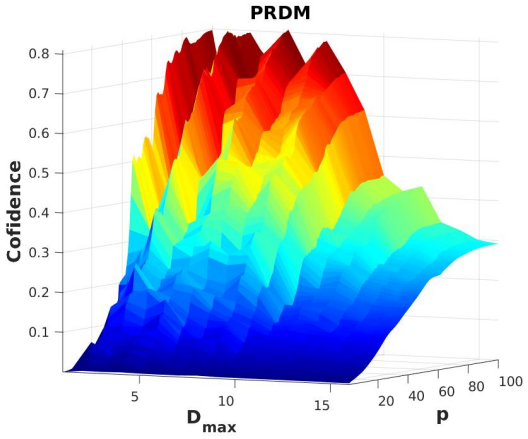


(b) PRDM method

Figure 9.6: Average Hamming maps of 9-qubit Max-Cut instances



(a) Average $\text{Conf}(p, D_{\max})^{\text{DM}}$.

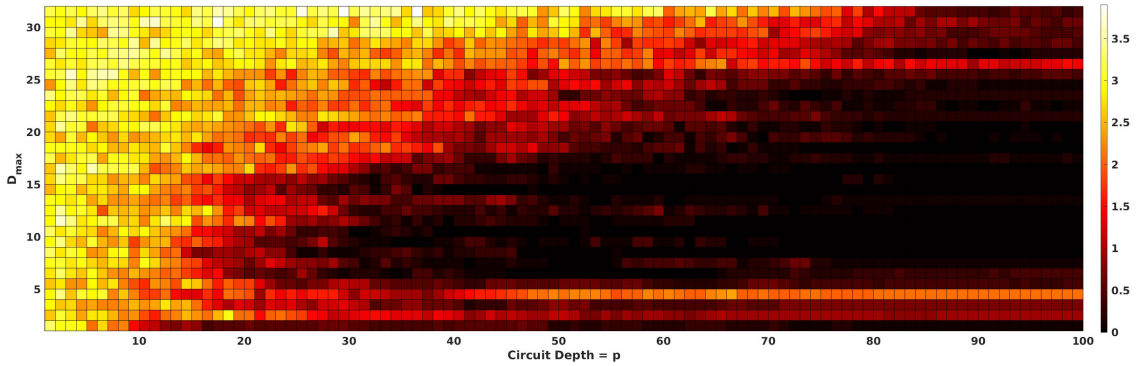


(b) Average $\text{Conf}(p, D_{\max})^{\text{PRDM}}$.

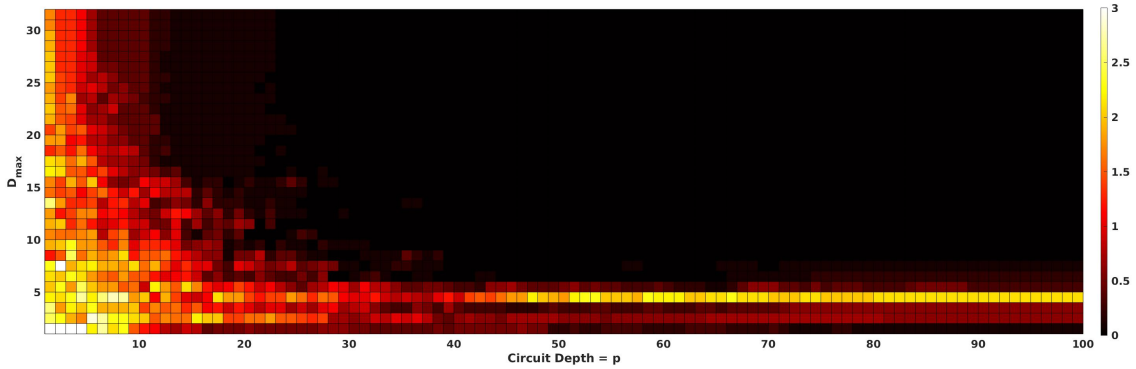
Figure 9.7: Average Confidence Plots for 9 qubit Max-Cut instances.

Figure 9.6 and Figure 9.7 presents the average behaviour of all the 10 9-qubit max-cut instances. Here too we observe all the previously explained features. One minor change would be in Figure 9.7a where we see that the average $\text{Conf}(p, D_{\max})^{\text{DM}}$ is not exactly zero for $D_{\max} > \frac{1}{2}\left(2^{\lfloor 9/2 \rfloor}\right) = 8$. We believe this is because matrix product states of odd numbered qubit registers have two bonds with the maximum bond-dimension as opposed to a single bond with highest bond-dimension in even numbered registers (refer subsection 2.6.2). Hence, registers with odd number of elements are subject to more approximations than those with even number of elements when having the same highest bond-dimension limit $2^{\lfloor n/2 \rfloor}$. This increased

approximations could result in fewer choices between solutions as explained during the discussion on the behaviour of QIOA for $D_{\max} < \frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$. Fewer available choices lead to making more confident predictions.

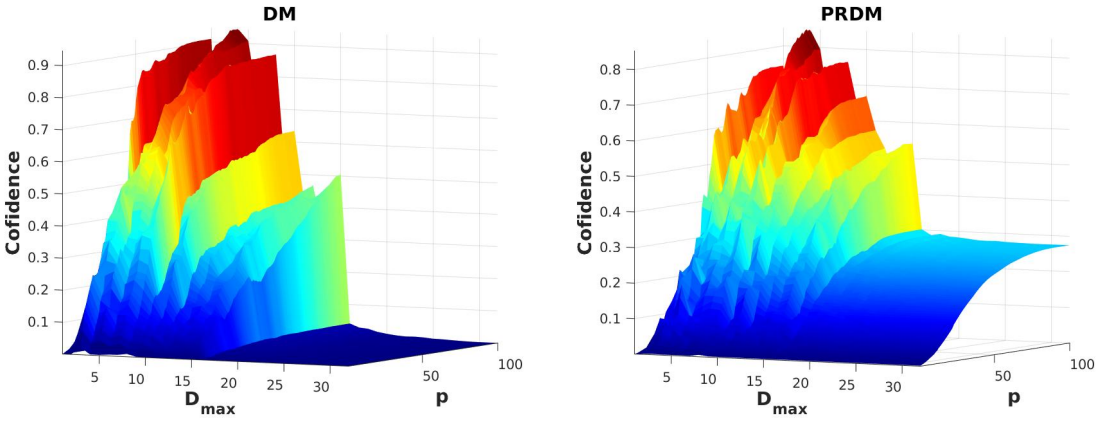


(a) DM method



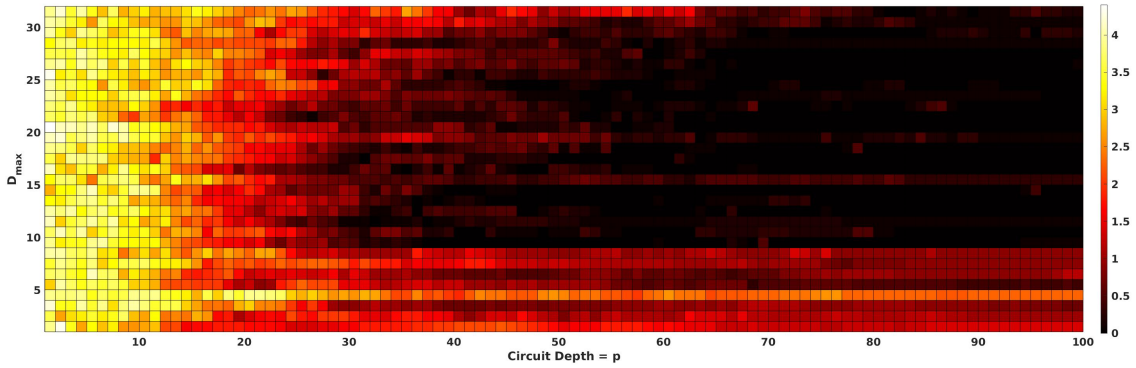
(b) PRDM method

Figure 9.8: Average Hamming maps of 10-qubit Max-Cut instances

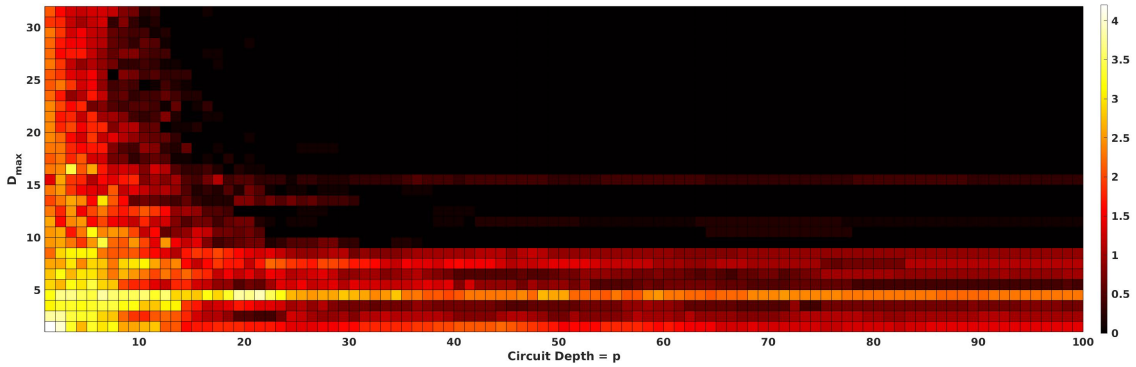


(a) Average $\text{Conf}(p, D_{\max})^{\text{DM}}$. **(b)** Average $\text{Conf}(p, D_{\max})^{\text{PRDM}}$.

Figure 9.9: Average Confidence Plots for 10 qubit Max-Cut instances.

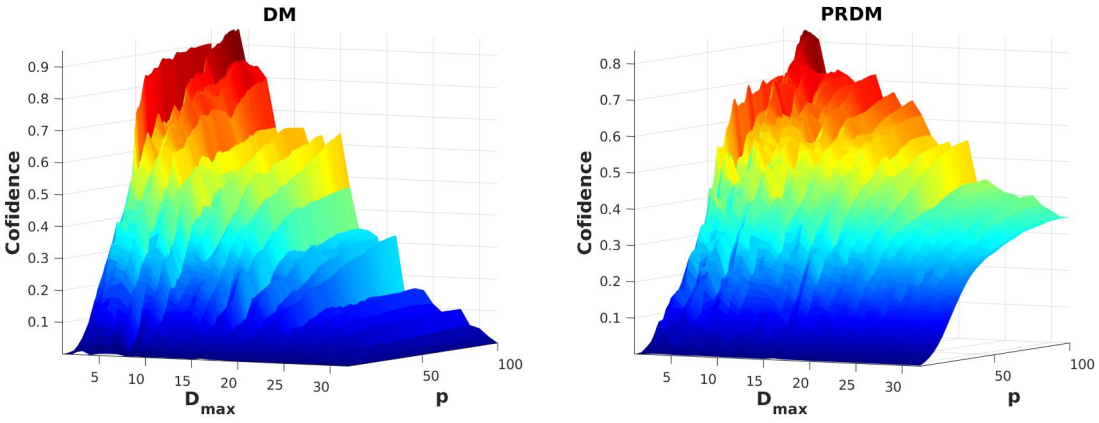


(a) DM method



(b) PRDM method

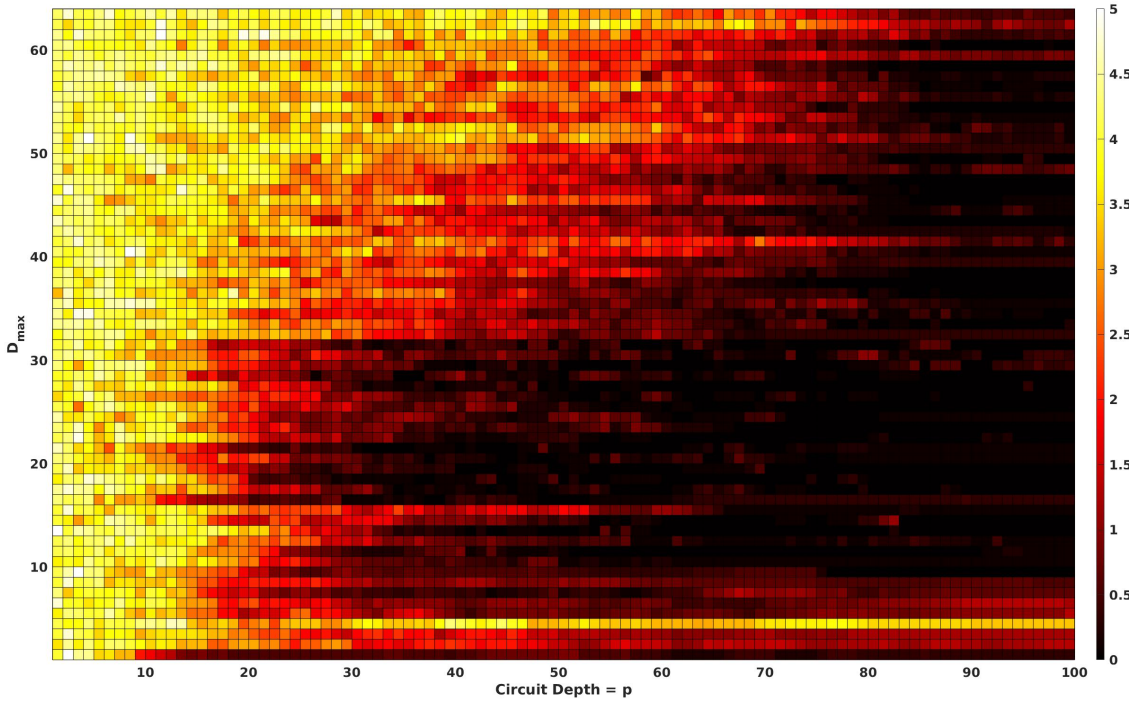
Figure 9.10: Average Hamming maps of 11-qubit Max-Cut instances



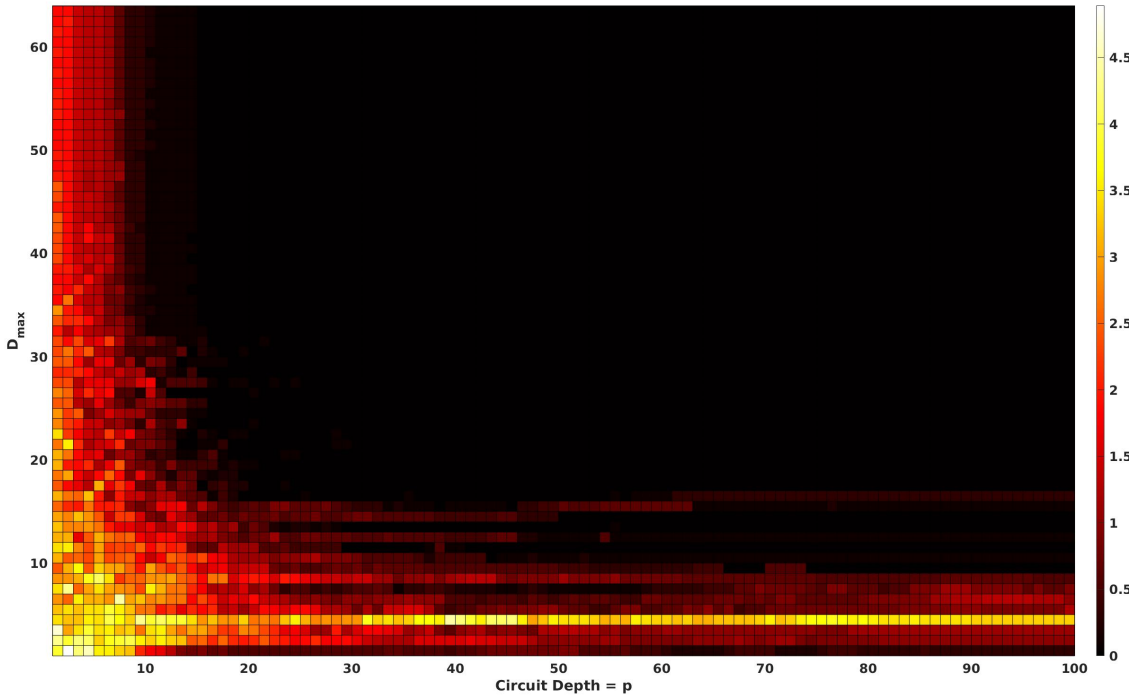
(a) Average $\text{Conf}(p, D_{\max})^{\text{DM}}$. **(b)** Average $\text{Conf}(p, D_{\max})^{\text{PRDM}}$.

Figure 9.11: Average Confidence Plots for 11 qubit Max-Cut instances.

Here too in Figure 9.11a, similar to Figure 9.7a, the average $\text{Conf}(p, D_{\max})^{\text{DM}}$ is noticeably greater than zero for $D_{\max} > \frac{1}{2}\left(2^{\lfloor 11/2 \rfloor}\right) = 16$. Nevertheless, the same reasoning provided for the 9 qubit case, applies here too, as they are both odd number of qubits.



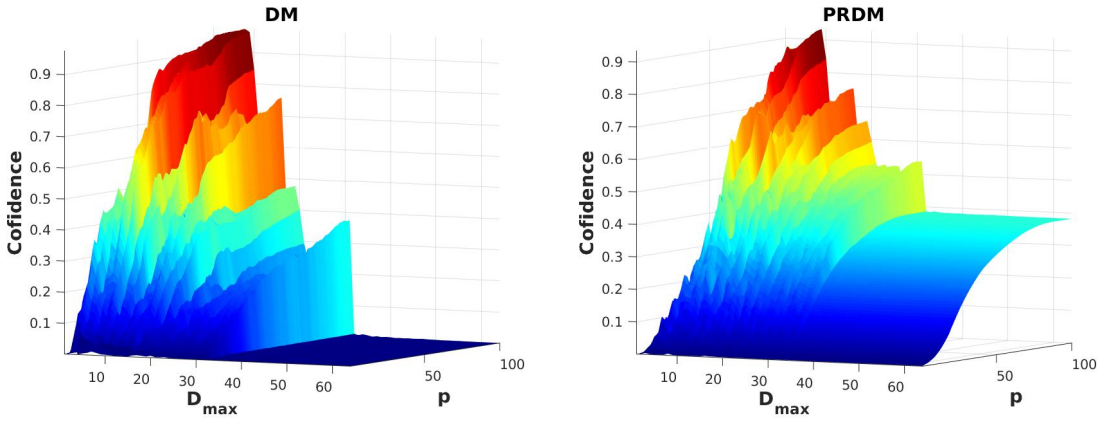
(a) DM method.



(b) PRDM method.

Figure 9.12: Average Hamming maps of 12-qubit Max-Cut instances

From Figure 9.12b, we can clearly see that QIOA on an average performs well for the PRDM method even for $D_{\max} < \frac{1}{2}\left(2^{\lfloor 12/2 \rfloor}\right) = 32$. This nature was not as clearly observable in the previous smaller instances.



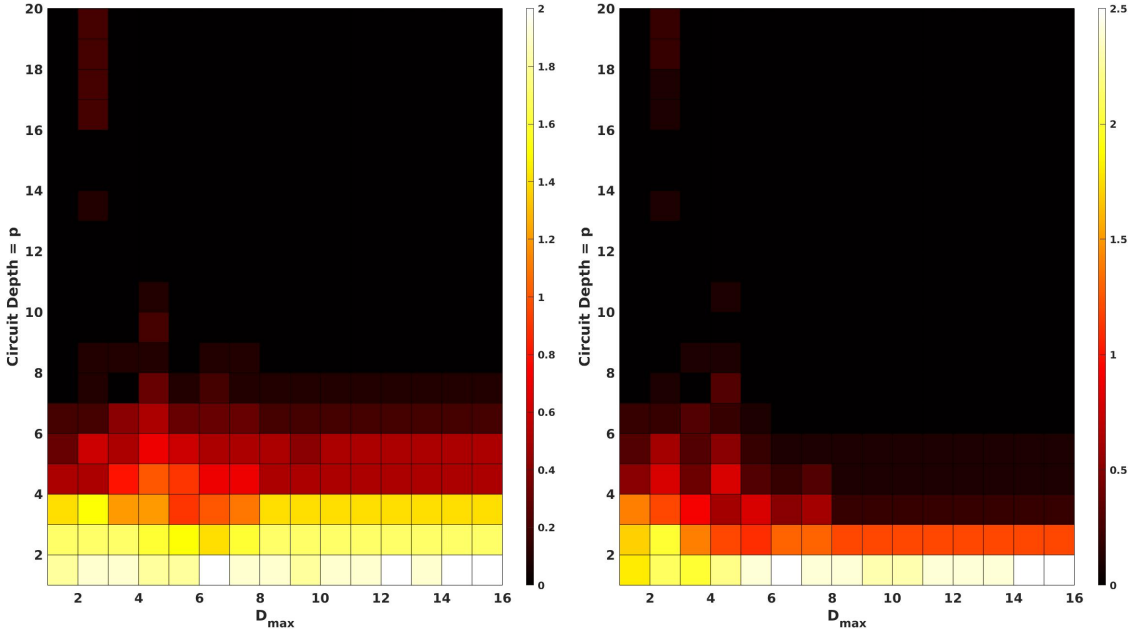
(a) Average $\text{Conf}(p, D_{\max})^{\text{DM}}$. **(b)** Average $\text{Conf}(p, D_{\max})^{\text{PRDM}}$.

Figure 9.13: Average Confidence Plots for 12 qubit Max-Cut instances.

Also, in Figure 9.13a, we see again that $\text{Conf}(p, D_{\max})^{\text{DM}} \approx 0$ for $D_{\max} > \frac{1}{2}\left(2^{\lfloor 12/2 \rfloor}\right) = 32$, as expected from an even-numbered instance.

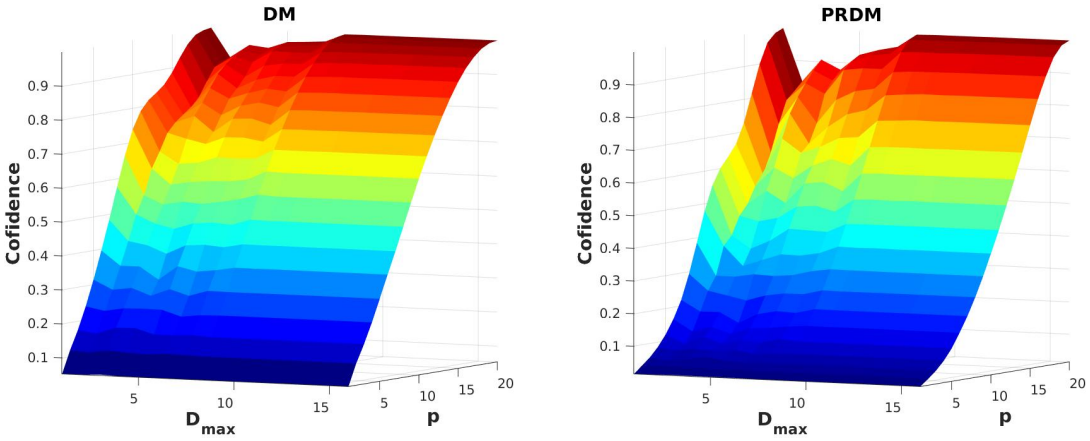
9.6 Exact-Cover QIOA Results

We repeated exactly the same calculations performed on the max-cut instances here too. Since adding individual plots will add little additional insights, the averaged behaviour of 8, 15, and 25 qubits are provided below, directly.



(a) DM method. **(b)** PRDM method.

Figure 9.14: Average Hamming maps for 8 qubit Exact-Cover instances.

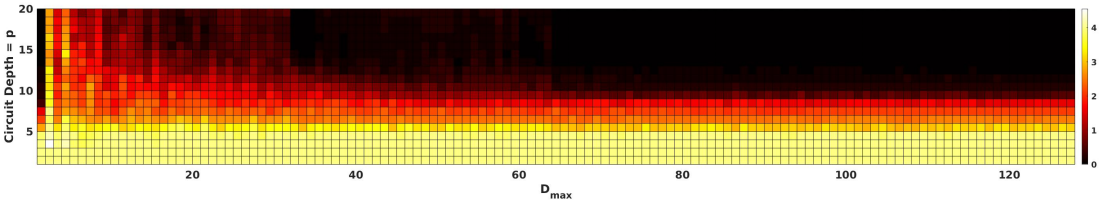


(a) Average Conf$(p, D_{\max})^{\mathrm{DM}}$. **(b)** Average Conf$(p, D_{\max})^{\mathrm{PRDM}}$.

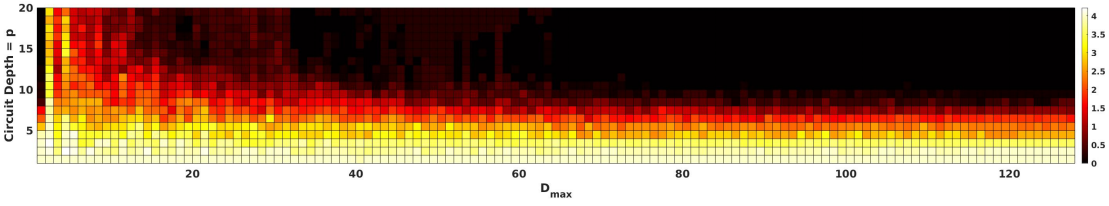
Figure 9.15: Average Confidence Plots 8 qubit Exact-Cover instances.

The Hamming maps presented in Figure 9.15, Figure 9.17, and Figure 9.19, present the average behaviour of 8, 15, and 25 qubit exact-cover instances respectively. Note that they have their $D_{\max}$ and p axes flipped with respect to that of the max-cut Hamming maps for aesthetic reasons. As previously anticipated in point 5 under section section 9.5, we see that both PRDM and DM methods present a comparable performance.

Nevertheless, we do observe that the PRDM methods starts working for a lower circuit depth of p in these instances and has a marginally better overall performance too. We attribute this to the fact that the different bits in the solution bit-string are not independent. Hence, the projection step performed in the PRDM method could be making use of these correlations to present a better performance.

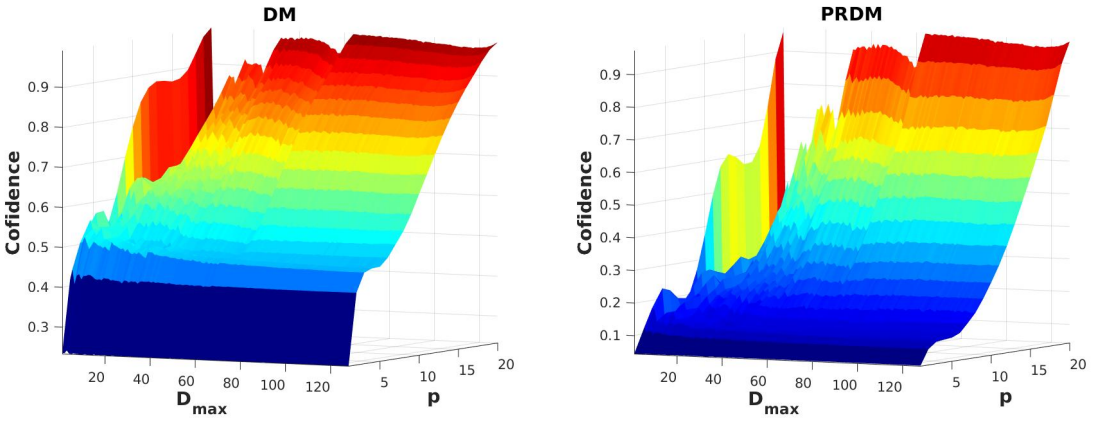


(a) DM method.



(b) PRDM method.

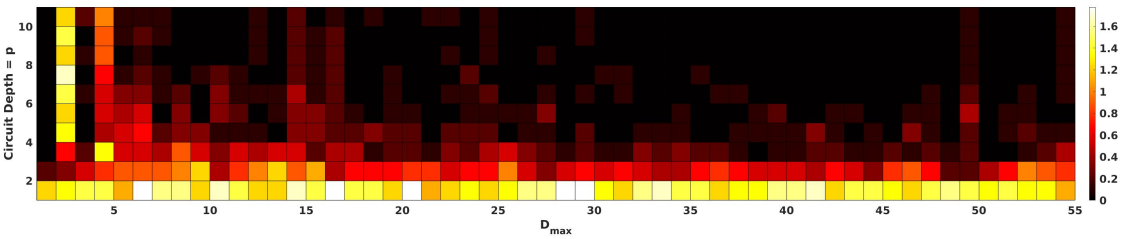
Figure 9.16: Average Hamming maps for 15 qubit Exact-Cover instances.



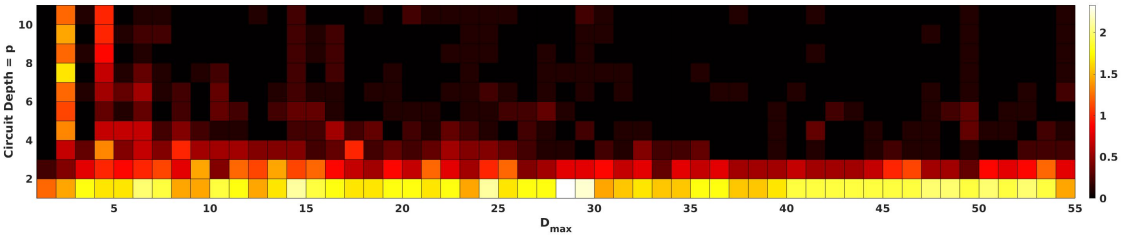
(a) Average $\text{Conf}(p, D_{\max})^{\text{DM}}$. **(b)** Average $\text{Conf}(p, D_{\max})^{\text{PRDM}}$.

Figure 9.17: Average Confidence Plots 15 qubit Exact-Cover instances.

There is also a significant difference between the confidence plots of exact-cover and max-cut instances. As previously pointed out, both $\text{Conf}(p, D_{\max})^{\text{DM}}$ and $\text{Conf}(p, D_{\max})^{\text{PRDM}}$ of the exact cover instances increases with increasing $D_{\max}$ and p and has no domains where they are approximately 0. To reiterate the point raised then, we expect this to be because of the lack of degeneracy, let alone maximally entangled degeneracy, in these exact-cover instances, hence avoiding the situation where both $'0'$ and $'1'$. This lack of degeneracy could also explain why the algorithm starts working for much lower circuit depth p for exact-cover problems, compared to the max-cut instances. They are also significantly less noisy. Another interesting observation is that lack of pronounced ridges perpendicular to the $D_{\max}$ axis as observed for max-cut confidence plots.

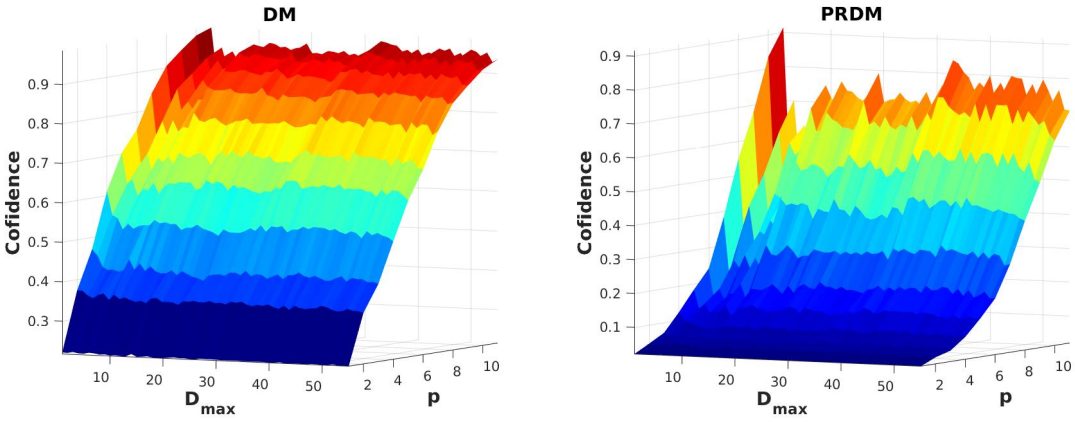


(a) DM method.



(b) PRDM method.

Figure 9.18: Average Hamming maps for 25 qubit Exact-Cover instances.



(a) Average Conf$(p, D_{\max})^{\mathrm{DM}}$. **(b)** Average Conf$(p, D_{\max})^{\mathrm{PRDM}}$.

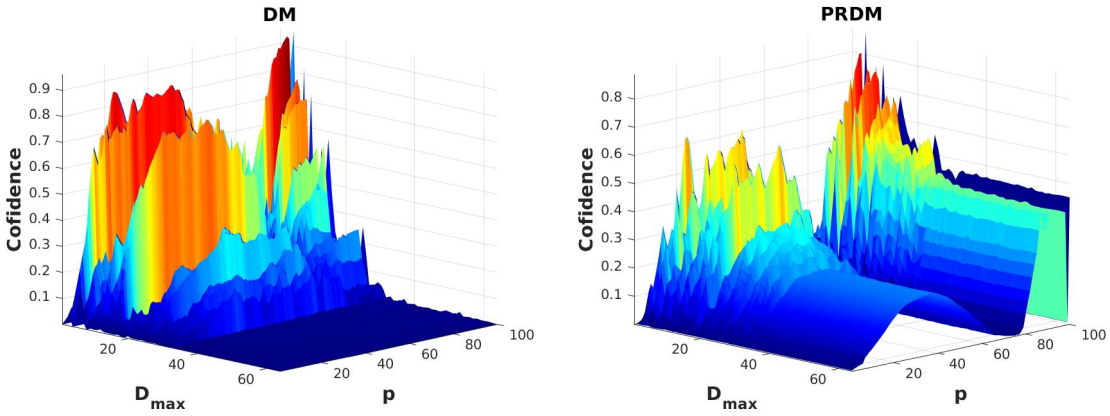
Figure 9.19: Average Confidence Plots 25 qubit Exact-Cover instances.

The next striking feature about these exact-cover instances is that all 29 of them start working when $D_{\max} = 1$ given enough circuit depths. For example, in all the 25 qubit instances shown in Figure 9.18a and Figure 9.18b, we see that the algorithm works when $D_{\max} = 1$ and $p > 2$. The reason for this could be tied back to chapter 8, section 8.3, where we see that the change in angles γ_{opt}, and β_{opt} are quite negligible with decreasing $D_{\max}$. Moreover, this could also be because of the ground state being a single product state too.

9.7 Outlier Instances

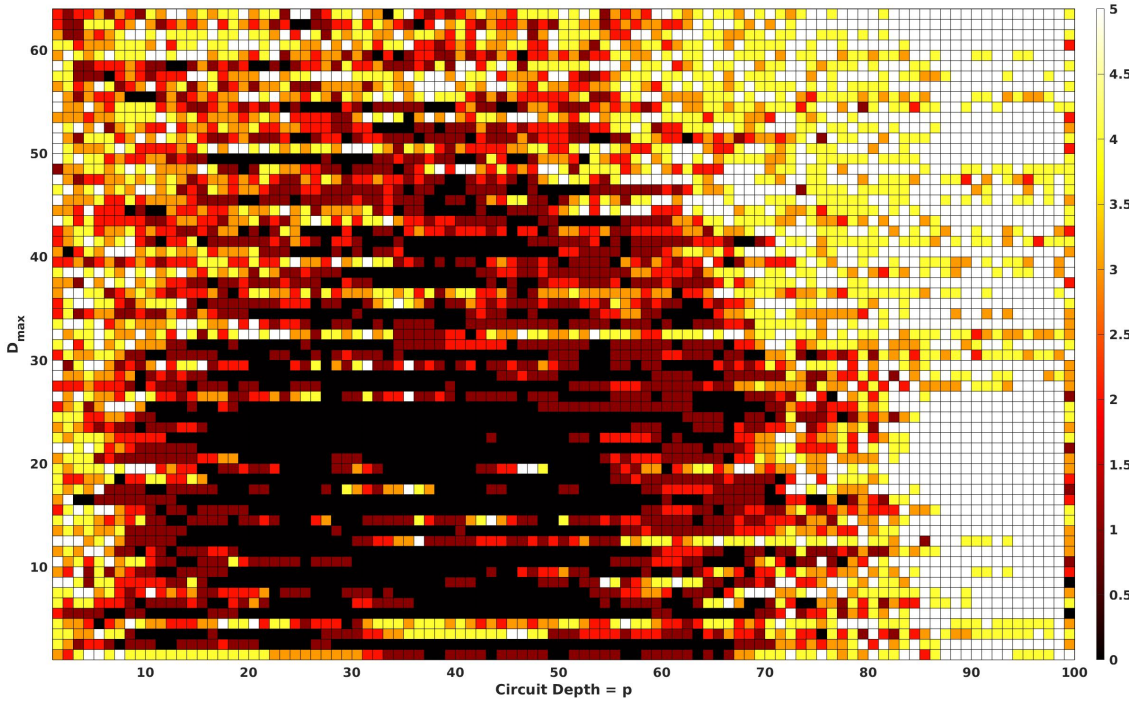
In this section, we present two outlier instances, one Max-Cut (Q12R4) and one Exact-Cover (Q25P8), both of which show characteristics different from their peers.

9.7.1 Q12R4 Max-Cut

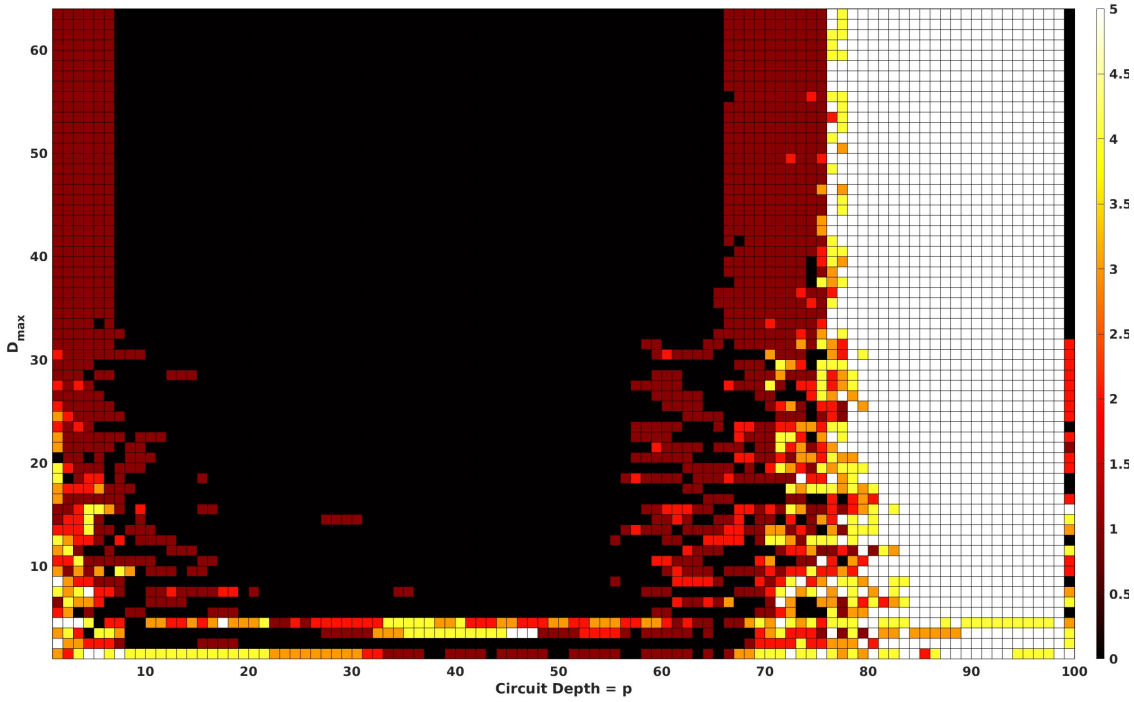


(a) Conf$(p, D_{\max})^{\mathrm{DM}}$ of Q12R4. **(b)** Conf$(p, D_{\max})^{\mathrm{PRDM}}$ of Q12R4.

Figure 9.20: Confidence Plots for Max-Cut instance Q12R4.



(a) DM method.



(b) PRDM method.

Figure 9.21: Hamming maps of the Max-Cut instance Q12R4

In Figure 9.21b, we observed a strange behaviour not seen in other instances. For smaller values of circuit depth p, the behaviour is as expected, both in the Hamming maps (Figure 9.21) and the confidence plots (Figure 9.20). But for larger p, we see a

large domain where the method stops working and gives the same bitstring as output, which has a minimum hamming distance of 5 from the set of 6 degenerate solutions. This is also reflected in Figure 9.20b, where we see a corresponding dip in the same domain where $\text{Conf}(p, D_{\max})^{\text{PRDM}} \approx 0$. We also see that the PRDM method starts working again for $p = 100$, and $\text{Conf}(p, D_{\max})^{\text{PRDM}}$ also goes back to its saturation value. A similar domain is also present in Figure 9.21a and Figure 9.20a of the DM method performance for this instance. Another interesting observation about Q12R4 is the existence of a second peak in its Solution landscape for $p = 1$, which although is the global maxima, does not correspond with any deep minima in its cost landscape. Hence, we suspect that this behaviour is a consequence of the linear interpolation algorithm we are using to calculate the angles for $p+1$ from the angles till p from Lukin et al. [46], and the true angles change for larger p, to the ones corresponding to this unaccounted for global maxima. At this point, this is still a hypothesis, and further studies are necessary.

9.7.2 Q25P8 Exact-Cover

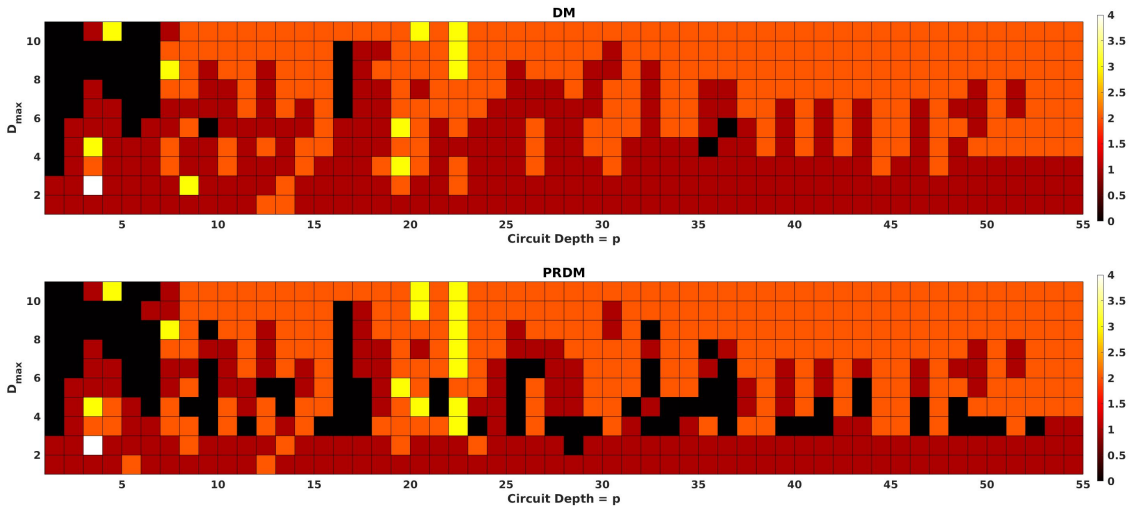


Figure 9.22: DM and PRDM Hamming maps for the Exact-Cover instance Q25P8

Figure 9.22 is different from the others within the same family because both DM and PRDM methods work in a considerably lesser set of $(p, D_{\max})$ pairs for this instance. Nevertheless, it is encouraging to see that the QIOA method still works when $D_{\max} = 1$, just as in all the other exact-cover cover instances. However, further analysis showed that in the large set of regions depicted in orange, where the hamming distance is 2 in both DM and PRDM methods, the algorithm has been finding the first excited state, instead of the ground state. This is because, for this particular problem, the difference between the first and ground state normalised energies is only equal to 4 units, while the highest energy state has an energy of 348 units. This means that this exception too, fits well into our story that QIOA is adept at finding low-energy states. Furthermore, increasing the circuit depth p values

higher than 11 should help to counter this type of issues caused by small spectral gaps in energy. Confidence plots of this instance show no diverging behaviour, further augmenting our claim.

9.8 Trainability of QIOA

In the current implementation of QIOA, we source the required $2p$ angles $\gamma_1, \ldots \gamma_p$ and $\beta_1, \ldots \beta_p$ through optimization protocols performed on cost landscapes calculated using full, non-truncated, $|\gamma, \beta\rangle$ states. As previously stated, these non-truncated states have an exponentially scaling bond-dimension $D_{\max}$ which quickly makes them intractable even using heuristic algorithms such as that of Lukin et al. [46]. Hence, to make QIOA a fully classical algorithm, one needs to find alternative ways to source the required $2p$ optimum angles.

Nevertheless, in the list below, we provide two arguments hinting at the possibility of training QIOA classically in polynomial time. That is, sourcing the $2p$ optimum angles classically.

1. First argument comes from chapter 8 where we discuss the possibility of classically training the parameters of vanilla-QAOA (only shown for circuit depth $p = 1$). Here, we note that the cost and solution landscapes largely retain their structure even after calculating them using highly truncated $|\gamma, \beta\rangle^{D_{\max}}$ states where $D_{\max}$ does not scale exponentially. However, one of the observations from this chapter was that although there is a slight drift though, in the optimum angles, the maxima of the solution landscape also shift accordingly. An important caveat to be raised here is that till now, we have only calculated the drift of these optimum angles with bond-dimension $D_{\max}$, and $p = 1$. Further calculations are required to study if this behaviour persists for larger circuit depth $p > 1$, which is imperative for QIOA to work.

2. Second argument comes from the Hamming maps presented in this chapter. There we see considerable noise for small values of $D_{\max}$, and in almost all the plots, we see that larger values of circuit depth p is required for QIOA to start working. We attribute this behaviour to the point raised above about the corresponding shift in solution landscape peaks. These shifts would imply that using the original angles lands us in sub-optimal regions in the solution landscape. Hence, we hypothesise that this noise could be because we are using the "wrong" angles for calculations. Further, we see that QIOA starts working for much smaller circuit depth p for exact-cover instances which have a lower drift in the optimum angles, and hence the solution maximas further supporting our claim.

These observations hint at the possibility of classically training QIOA using a full MPS based implementation with reduced bond-dimensions, which we plan to implement in a near future.

Part IV

Conclusion

10
Conclusion

10.1 Summary

In this thesis, we implemented our own matrix product state (MPS) based quantum circuit simulator using Google X's tensornetwork package [53]. Matrix product states are linearised representations of many-body quantum states, where every individual component within the full many-body state is represented as a rank 3 tensor (see chapter 2). We then used this simulator to study the characteristics of the Quantum Approximate Optimisation Algorithm [2] under different circumstances. Although any quantum state can be expressed exactly as an MPS, working with MPS provides an additional option to limit the amount of entanglement in the quantum register. This is done by performing Schmidt decompositions over each possible bi-partitions along the linear chain, and then limiting the maximum number of allowed Schmidt weights. This corresponds to limiting the amount of entanglement within the full state. This is because the entanglement between two partitions of a state is a function of Schmidt weights of that partition. Also, the Schmidt rank of a bi-partition is referred to as the bond-dimension corresponding to that partition. In highly entangled states, the number of non-zero Schmidt weights, and hence the bond-dimension, increases exponentially with the system size. It is this exponential increase in bond-dimension which makes it intractable for classical systems to represent and simulate highly entangled quantum states. Using MPS however, one could approximate a highly entangled, classically intractable, quantum state $|\psi\rangle$, to obtain another state $|\psi'\rangle$ where the entanglement scales non-exponentially, and is hence classically manageable. In this light, we used our MPS based quantum circuit simulator to study the effect of limiting the amount of entanglement in QAOA circuits, on the performance of QAOA. The amount of entanglement was limited by truncating the bond-dimensions and setting an upper-bound = $D_{\max}$, which does not scale exponentially with system size.

10.1.1 QAOA

The QAOA algorithm first maps the optimisation problem it is aiming to solve onto a cost Hamiltonian H_C. It then creates a parametrised quantum state $|\gamma, \beta\rangle$ by applying single and two-qubit gates derived from H_C and parametrised by β and γ respectively. Then, it calculates the optimum parameters γ_{opt} and β_{opt} that corresponds with a $|\gamma_{opt}, \beta_{opt}\rangle$ state which has the lowest energy expectation value with respect to H_C. This is referred to as the training phase of QAOA and is performed with the help of classical optimisation routines in the original QAOA.

The algorithm then creates this optimums state, $|\gamma_{opt}, \beta_{opt}\rangle$, on a QPU. This state is then sampled from in the computational basis, to find the basis state with maximum weight. The idea behind QAOA is that after applying "sufficient" circuit depth p, this state with the highest weight is the ground state of the problem it is attempting to solve. In the infinite limit when $p \to \infty$, the $|\gamma_{opt}, \beta_{opt}\rangle$ state is exactly equal to the ground state. The second phase where $|\gamma_{opt}, \beta_{opt}\rangle$ state is repeatedly sampled to find the solution is called the sampling phase of QAOA.

Before moving on to approximated QAOA calculations, we first verified if our simulator is performing as expected by benchmarking it with the results obtained from a vector-matrix implementation of QAOA as in [39]. We compared if both methods produce the same results when there are no approximations made and verified this to be the case. Next, we performed multiple calculations of the QAOA algorithm implemented on different sized instances of max-cut and exact-cover problems. The two main results obtained from these simulations are presented in chapter 8 and chapter 7.

10.1.2 Training Results for $p = 1$

First, the effect of limiting the bond-dimension to a user-defined maximum value $= D_{\max}$, and hence limiting the entanglement, on the training phase of QAOA for circuit depth $p = 1$ was studied. We calculated the cost and solution landscapes, defined as ${}^{D_{\max}}\langle\beta, \gamma|H_C|\gamma, \beta\rangle^{D_{\max}}$ and $\langle gs|\gamma, \beta\rangle^{D_{\max}}$ respectively, over multiple values of maximum bond-dimension limit $1 \leq D_{\max} \leq 2^{\lfloor n/2 \rfloor}$, and for $\gamma \in [0, 2\pi]$ and $\beta \in [0, \pi]$.

Here, quite surprisingly, we saw that the cost and solution landscapes largely retain their shape even on truncating the bond-dimension to very small values (see Figure 8.1 and Figure 8.9). This means that the minimas in the cost landscape and the maximas in the solution largely remain in the same position. Since all we really care about is the position of the minima in the cost landscape during the training phase of QAOA, this prompted us to suggest that it could be classically possible to find these minimas using our MPS based calculations. We presented the results and discussions on this possibility both for max-cut and exact-cover instances in section 8.2 and section 8.3 respectively.

Ultimately, our method needs to be tested on simulations with larger circuit depth, $p > 1$, to understand if the training behaviour discussed in chapter 8 still holds. An interesting avenue we hope to explore in the near future.

10.1.3 Sampling Results and QIOA

Our training phase results showed that approximated QAOA states $|\gamma, \beta\rangle^{D_{\max}}$ too, retained useful information, even for $D_{\max} \ll 2^{\lfloor n/2 \rfloor}$. This inspired us to investigate if the approximated optimum state $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ also, retained some of the intended

consequences of QAOA, such as an increase in the population of the solution state $|gs\rangle$ with increasing circuit depth p. That is, from the original QAOA algorithm, we know that the weight of the solution $|gs\rangle$ increases, and all the other weights decrease with increasing circuit depth p in the full, non-truncated $|\gamma_{opt}, \beta_{opt}\rangle$ state. We wanted to investigate if this same behaviour also persisted in truncated $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ states with maximum bond-dimension $D_{\max} \ll 2^{\lfloor n/2 \rfloor}$.

If we show that this behaviour persists in the truncated states also, and that it is possible to extract the state with the highest weight from this truncated state, then it would mean that the sampling phase of the original QAOA is classically tractable. Conversely, this also means that if we show that the sampling phase of QAOA could be classically manageable using truncated MPS calculations, then it would mean that the original behaviour of QAOA persists even on truncation. It is, in fact, the latter path we have chosen in this thesis. We designed a method to extract the state with the highest weight from a $|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$ state. Then, we test if this extracted state is the ground state of the cost Hamiltonian, and hence the solution to the problem we are trying to solve. We call this method QIOA or Quantum Inspired Optimisation Algorithm, and it has been described in detail in chapter 7. Before moving on to talk about the performance of QIOA, it is important to note that, in this investigation, we calculated the optimum parameters for larger circuit depth p using a vector matrix implementation of the original QAOA and not our MPS implementation. For max-cut instances, we calculated the optimum angles up to $p = 100$, and for the exact-cover instances, a circuit depth up to $p = 20$ was used for 8 and 15 qubit instances, while a $p = 11$ was used for 25-qubit instances. These results have been presented in chapter 9.

Expounding on the results, for the case of max-cut instances, we see that the QIOA method works for every bond-dimension $D_{\max}$ when $D_{\max} > \frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$ and for a circuit p as good as the one required in the original QAOA. This though, isn't very interesting because we still have an exponentially scaling bond-dimension. However, we also see that our algorithm works heuristically for smaller bond-dimensions $D_{\max}$ as well, but require higher circuit depth p on an average. The heuristic aspect of the algorithm is that for $D_{\max} < \frac{1}{2}\left(2^{\lfloor n/2 \rfloor}\right)$ given a circuit depth p, and a bond dimension $D_{\max}$, at the moment, it is not possible to know a priory if QIOA gives the correct solution or not. Further work is needed to better understand the predictability of the algorithm. This heuristic aspect, however, is not a major limitation, as one could overcome this by exploring a small classically tractable region within the $(p, D_{\max})$ space until one solution is found. In the case of exact-cover instances though, we see that QIOA works in all cases for $D_{\max} = 1$ and a circuit depth as large as needed in the original QAOA, which is very promising.

10.2 Applicability of QIOA method

In chapter 7, section 7.4, we have discussed multiple ways to extract the state $|s\rangle$ which holds the highest coefficient within a truncated matrix product representation

$|\gamma_{opt}, \beta_{opt}\rangle^{D_{\max}}$. At this point, it is important to clarify that we have no indications suggesting that the QIOA method can be used generally in all cases of sampling problems. To the best of our knowledge, the method only works well in cases where the coefficient of the state $|s\rangle$ is much higher than all the other coefficients. This is a very strong requirement that limits the applicability of QIOA. In that aspect, the QIOA method is assured to work exactly, only when extracting the most populous state from a full $|\gamma_{opt}, \beta_{opt}\rangle$ state with no truncations, and an infinite circuit depth p. This is because here, it is guaranteed that the population of the ground state $|gs\rangle$ increases and the population of other states decreases with increasing circuit depth p. However, exact representations of these $|\gamma_{opt}, \beta_{opt}\rangle$ states provably need an exponentially increasing number of parameters, which quickly becomes classically intractable.

All we are showing with the QIOA method in this thesis is that, in all the instances we have studied, we see that the main behaviours of QAOA persists even after making approximations to the entanglement. Hence, using the framework of tensor networks and matrix product states, we can extract the solution from a truncated state as well. This makes QIOA a promising avenue for interesting future explorations.

10.3 Future Work

Currently, we source the $2p$ required optimum parameters from non-approximated implementations of the original QAOA. This was possible only because of the small system sizes we looked at. Hence, to have a completely classical algorithm, we would need to extend our training results described in chapter 8 for $p > 1$ as well. This is something we are planning to work on in the near future.

Another big caveat in our study is that till now, we have only studied the performance of QIOA for very small system sizes. While the system sizes we have looked at are what is generally seen in literature when talking about quantum algorithms, the best classical algorithms today work with sizes at-least two orders of magnitude higher. Further work is required to elucidate the exact scaling and complexity of QIOA.

Another important future work is to perform full state tomographies of all the instances we have simulated to study more on if and how the behaviour of QAOA persists even for smaller bond-dimensions. At the moment, we only provide converse arguments that the behaviour of QAOA persists for smaller bond-dimensions because QIOA works for smaller bond-dimensions. More rigorous methods of study are required to further validate this claim. Full state tomographies over different bond-dimensions would allow us to explicitly list the role MPS plays in the QIOA method too. Moreover, studying the effect of truncating bond-dimensions in an MPS state on the weights of different energy levels is an interesting problem in itself.

As stated before, currently QIOA is a heuristic algorithm where it is not possible to not know if it works for a given $(p, D_{\max})$. Further work is required to provide more rigorous proofs or constraints on when QIOA works, and when it does not. We also need more investigations to study if QIOA is more generally applicable or has similarities to some sub-class of sampling problems.

It could also be interesting to see how well the QIOA method performs in the case of other variational quantum-classical hybrid algorithms, such as VQE [3], that encode problem solutions in ground states of Hamiltonians.

Bibliography

[1] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 8 2018.

[2] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. pages 1–16, 2014.

[3] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man Hong Yung, Xiao Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 7 2014.

[4] Amir Khoshaman, Walter Vinci, Brandon Denis, Evgeny Andriyash, and Mohammad H. Amin. Quantum variational autoencoder. *Quantum Science and Technology*, 4(1), 1 2019.

[5] Ian D. Kivlichan, Jarrod McClean, Nathan Wiebe, Craig Gidney, Alán Aspuru-Guzik, Garnet Kin Lic Chan, and Ryan Babbush. Quantum Simulation of Electronic Structure with Linear Depth and Connectivity. *Physical Review Letters*, 120(11), 3 2018.

[6] Charles E. Leiserson, Neil C. Thompson, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lampson, Daniel Sanchez, and Tao B. Schardl. There's plenty of room at the Top: What will drive computer performance after Moore's law?, 6 2020.

[7] E Drexler Richard Feynman, undefined Pasadena, and undefined 2009. There's Plenty of Room at the Bottom.

[8] Andreas Junk and Falk Riess. From an idea to a vision: There's plenty of room at the bottom. *American Journal of Physics*, 74(9):825–830, 9 2006.

[9] Richard P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6):507–531, 6 1986.

[10] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[11] Sanjeev Arora and Boaz Barak. Computational Complexity: A Modern Approach. *Complexity*, 1(January):155–160, 2007.

[12] R L Rivest, A Shamir, and L Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Technical report.

[13] Ronald De Wolf. The Potential Impact of Quantum Computers on Society. Technical report, 2017.

[14] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G S L Brandao, David A Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins,

William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P Harrigan, Michael J Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S Humble, Sergei V Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R Mcclean, Matthew Mcewen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C Platt, Chris Quintana, Eleanor G Rieffel, Pedram Roushan, Nicholas C Rubin, Daniel Sank, Kevin J Satzinger, Vadim Smelyanskiy, Kevin J Sung, Matthew D Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505, 2019.

[15] On "Quantum Supremacy" | IBM Research Blog.

[16] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. Technical report, 2019.

[17] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation system. In *Leibniz International Proceedings in Informatics, LIPIcs*, volume 67. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 11 2017.

[18] András Gilyén, Zhao Song, and Ewin Tang. An improved quantum-inspired algorithm for linear regression. *arXiv:2009.07268*, 9 2020.

[19] Samuel Mugel, Carlos Kuchkovsky, Escolastico Sanchez, Samuel Fernandez-Lorenzo, Jorge Luis-Hita, Enrique Lizaso, and Roman Orus. Dynamic Portfolio Optimization with Real Datasets Using Quantum Processors and Quantum-Inspired Tensor Networks. 6 2020.

[20] Chen Ding, Tian-Yi Bao, and He-Liang Huang. Quantum-Inspired Support Vector Machine. 14(8), 6 2019.

[21] John Realpe-Gómez and Nathan Killoran. Quantum-inspired memory-enhanced stochastic algorithms. 6 2019.

[22] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 1 2011.

[23] Jacob C. Bridgeman and Christopher T. Chubb. Hand-waving and interpretive dance: An introductory course on tensor networks, 5 2017.

[24] Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states, 2014.

[25] Itai Arad and Zeph Landau. Quantum computation and the evaluation of tensor networks. *SIAM Journal on Computing*, 39(7):3089–3121, 2010.

[26] Norbert Schuch, David Pérez-García, and Ignacio Cirac. Classifying quantum phases using matrix product states and projected entangled pair states. *Physical Review B - Condensed Matter and Materials Physics*, 84(16):165139, 10 2011.

[27] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2010.

[28] M. B. Hastings. An Area Law for One Dimensional Quantum Systems. 5 2007.

[29] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. Technical report, 2015.

[30] Boaz Barak, Ankur Moitra, Ryan O'donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, and John Wright. Beating the random assignment on constraint satisfaction problems of bounded degree. Technical report, 2015.

[31] Edward Farhi and Aram W Harrow. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. 2 2016.

[32] M. Streif and M. Leib. Training the quantum approximate optimization algorithm without access to a quantum processing unit. *Quantum Science and Technology*, 5(3), 2020.

[33] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor Rieffel, Davide Venturelli, and Rupak Biswas. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms*, 12(2):34, 2 2019.

[34] Andreas Bärtschi and Stephan Eidenbenz. Grover Mixers for QAOA: Shifting Complexity from Mixer Design to State Preparation. Technical report, 2020.

[35] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum Computation by Adiabatic Evolution. 1 2000.

[36] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2:1–14, 2 2014.

[37] About Us - Jeppesen.

[38] Mattias Grönkvist and Gr¨ Grönkvist. The Tail Assignment Problem. Technical report, 2005.

[39] Pontus Vikstål, Mattias Grönkvist, Marika Svensson, Martin Andersson, Göran Johansson, and Giulia Ferrini. Applying the Quantum Approximate Optimization Algorithm to the Tail Assignment Problem. *Physical Review Applied*, 14(3), 12 2019.

[40] M. Willsch, D. Willsch, F. Jin, H. DeRaedt, and K. Michielsen. Benchmarking the quantum approximate optimization algorithm. *Quantum Information Processing*, 19(7), 2020.

[41] Gavin E. Crooks. Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem. 11 2018.

[42] P Erdds and A R&wi. On random graphs I. Technical report.

[43] G Strang Wellesley-Cambridge Press and undefined Massachusetts. Introduction to linear algebra. 2003.

[44] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. Springer Texts in Statistics An Introduction to Statistical Learning. Technical report.

[45] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1 1965.

[46] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices. 12 2018.

[47] MR Gary and DS Johnson. Computers and Intractability: A Guide to the Theory of NP-completeness. 1979.

[48] Fernando G S L Brandão, Michael Broughton, Edward Farhi, Sam Gutmann, and Hartmut Neven. For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances. Technical report, 2018.

[49] Hadley Wickham and Lisa Stryjewski. 40 years of boxplots. Technical report, 2011.

[50] Jeffrey S. Simonoff. *Smoothing Methods in Statistics*. Springer Series in Statistics. Springer New York, New York, NY, 1996.

[51] RW Hamming. Coding and information theory. 1986. *Englewood Cliffs, NJ, Prentice Hall*.

[52] DP Williamson and DB Shmoys. The Design of Approximation Algorithms, 2011.

[53] Chase Roberts, Ashley Milsted, Martin Ganahl, Adam Zalcman, Bruce Fontaine, Yijian Zou, Jack Hidary, Guifre Vidal, and Stefan Leichenauer. TensorNetwork: A Library for Physics and Machine Learning. Technical report.

A

Appendix 1

Python Code

A.1 Function to create a QAOA MPS state

```
def QAOA_mps(gamma, beta, gamma_beta, Dmax):

    Q_ord = np.linspace(start = 0, stop = (n-1), num = n, dtype = int)

    SWAP = g.get_SWAP()


    for i in range(n):

        Rz = g.get_Rz(2*gamma*h[i])
        gamma_beta.apply_one_site_gate(Rz, i)

    for i in range(n):

        if (i < (n-1)):

            for k in range(n-1):

                if (Q_ord[k] < Q_ord[k+1]):

                    Jij = g.get_Jij(gamma, J[Q_ord[k]][Q_ord[k+1]])
                    gamma_beta.apply_two_site_gate(Jij, site1 = k, site2 = (k+1))

            gamma_beta,_ = trunc.truncate_mps(gamma_beta, Dmax)



        if (i%2 == 0):
            for s in range(l1):

                Q_ord[2*s],Q_ord[2*s+1] = Q_ord[2*s+1],Q_ord[2*s]
```

```
                gamma_beta.apply_two_site_gate(SWAP, site1 = (2*s), site2 = (2*s+1))

            gamma_beta,_ = trunc.truncate_mps(gamma_beta, Dmax)

        else:

            for s in range(l2):

                Q_ord[2*s+1],Q_ord[2*s+2] = Q_ord[2*s+2],Q_ord[2*s+1]
                gamma_beta.apply_two_site_gate(SWAP, site1 = (2*s+1), site2 = (2*s+2))

            gamma_beta,_ = trunc.truncate_mps(gamma_beta, Dmax)


    Rx = g.get_Rx(2*beta)

    for i in range(n):

        gamma_beta.apply_one_site_gate(Rx, i)

    gamma_beta = gamma_beta.tensors[::-1]
    gamma_beta = [gamma_beta[x].transpose([2,1,0]) for x in range(n)]
    gamma_beta = tn.FiniteMPS(gamma_beta,canonicalize = False)

    return gamma_beta
```

A.2 Function to Calculate the Cost of a QAOA state

```
def get_Cost_mpsIII(gamma_beta):

    '''
    Using network implementation for hi
    Takes 23 seconds.

    Fastest method.

    '''

    Cost = 0
    Sz1 = tn.Node(g.get_Z())

    for i in range(n):

```

```
        g_b = gamma_beta.tensors
        g_b = [tn.Node(g_b[x]) for x in range(n)]
        g_bcon = [tn.conj(g_b[x]) for x in range(n)]

        for k in range(n):

            if (k == i):

                g_b[k][1]^Sz1[0]
                g_bcon[k][1]^Sz1[1]

            else:

                g_bcon[k][1]^g_b[k][1]

            if (k == (n-1)):

                g_bcon[k][2]^g_bcon[0][0]
                g_b[k][2]^g_b[0][0]

            else:

                g_bcon[k][2]^g_bcon[k+1][0]
                g_b[k][2]^g_b[k+1][0]



        C = tn.contractors.greedy((g_bcon + [Sz1] + g_b))
        C = C.tensor
        C = np.real(C.item())
        C = h[i]*C
        Cost = Cost + C

    Sz2 = tn.Node(g.get_Z())

    for i in range(n-1):

        for j in range((n-1),i,-1):

            g_b = gamma_beta.tensors
            g_b = [tn.Node(g_b[x]) for x in range(n)]
            g_bcon = [tn.conj(g_b[x]) for x in range(n)]

            for k in range(n):

                if (k == i):
```

```
62
63                      g_b[k][1]^Sz1[0]
64                      g_bcon[k][1]^Sz1[1]
65
66                  elif (k == j):
67
68                      g_b[k][1]^Sz2[0]
69                      g_bcon[k][1]^Sz2[1]
70
71                  else:
72
73                      g_bcon[k][1]^g_b[k][1]
74
75                  if (k == (n-1)):
76
77                      g_bcon[k][2]^g_bcon[0][0]
78                      g_b[k][2]^g_b[0][0]
79
80                  else:
81
82                      g_bcon[k][2]^g_bcon[k+1][0]
83                      g_b[k][2]^g_b[k+1][0]
84
85
86              C = tn.contractors.greedy((g_bcon + [Sz1, Sz2] + g_b))
87              C = C.tensor
88              C = np.real(C.item())
89              C = J[i,j]*C
90              Cost = Cost + C
91              del g_b, g_bcon
92
93
94      return Cost
```

A.3 QIOA Main Method

This is the main method described in Figure 7.1

```
1  q = 12
2
3  l1 = int(np.floor((q)/2))
4  l2 = int(np.floor((q-1)/2))
5
6  for s in range(1):
7
```

```
    ####### Data Extraction
    location = '/home/ph30n1x/Chalmers/Thesis/QAOA/MaxCut/Q'

    C = np.load(location+str(q)+tag+str(s)+'/C.npy')
    n = len(C)

    Gamma0 = np.load(location+str(q)+tag+str(s)+'/Gamma0.npy')
    Beta0 = np.load(location+str(q)+tag+str(s)+'/Beta0.npy')


    Sol = []
    for j in range(1,33):

        try:

            S = np.load(location+str(n)+tag+str(s)+'/Fire/Sol_str_'+str(j)+'.npy')

            S = str(S)
            Sol.append(S)

        except:
            continue

    num_of_Sols = len(Sol)

    ####### Initialisation

    Pmax = len(Gamma0)
    Dlim = 100

    Paxis = list(range(1,(Pmax+1)))
    Daxis = list(range(1,(Dlim+1)))
    X,Y = np.meshgrid(Daxis,Paxis)

    qioa_best = np.ones([Pmax,Dlim])
    qioa_worst = np.ones([Pmax,Dlim])

    qioa_dm = np.ones([Pmax,Dlim])
    qioa_prdm = np.ones([Pmax,Dlim])
    qioa_la = np.ones([Pmax,Dlim])

    DM_confidence = np.zeros([Pmax,Dlim])
    PRDM_confidence = np.zeros([Pmax,Dlim])
    LA_confidence = np.zeros([Pmax,Dlim])


```

```
    ####### State Preparation

    for Dmax in range(1,Dlim+1):

        for p in range(1,Pmax+1):

            try:

                gamma_beta = tn.FiniteMPS([np.array([[[1/np.sqrt(2)],
                                                      [1/np.sqrt(2)]]]) for x in range(n)])

                GB = QAOA_mps(Gamma0[0],Beta0[0],gamma_beta,Dmax,C)

                Sol_prdm_pr = ''
                Sol_dm_pr = ''
                Sol_la_pr = ''

                for i in range(1,p):

                    GB = QAOA_mps(Gamma0[i],Beta0[i],GB,Dmax,C)


                GB = tn.FiniteMPS(GB.tensors, center_position = 0, canonicalize = True)

                ####### End State Preparation

                # print("\n\n start LA : ",p,Dmax,'\n\n')

                ########## Start Linear Algebra LA method ##########

                diff_la = np.zeros(q)
                tens = GB.tensors
                tens_0 = [np.matrix(tens[x][:,0,:]) for x in range(n)]
                tens_1 = [np.matrix(tens[x][:,1,:]) for x in range(n)]

                k0 = np.matmul(tens_0[0],tens_0[0].H)
                k1 = np.matmul(tens_1[0],tens_0[0].H)


                if (abs(k0.item()) > abs(k1.item())):

                    K_b = tens_0[0]
                    Sol_la_pr += '0'

                else:

```

```
100                 K_b = tens_1[0]
101                 Sol_la_pr += '1'
102 
103             diff_la[0] = abs(k0.item()) - abs(k1.item())
104 
105             for i in range(1,n):
106 
107                 k0 = np.matmul(K_b,tens_0[i])
108                 k1 = np.matmul(K_b,tens_1[i])
109 
110                 k0 = np.matrix(k0)
111                 k1 = np.matrix(k1)
112 
113                 print("Difference LA   = ",abs(np.matmul(k0,k0.H).item()) - abs(np.matmul(k1,k
114 
115                 diff_la[i] = abs(np.matmul(k0,k0.H).item()) - abs(np.matmul(k1,k1.H).item())
116 
117                 if ( abs(np.matmul(k0,k0.H).item()) > abs(np.matmul(k1,k1.H).item()) ):
118 
119                     K_b = k0
120                     Sol_la_pr += '0'
121 
122                 else:
123 
124                     K_b = k1
125                     Sol_la_pr += '1'
126 
127             LA_confidence[p-1,Dmax-1] = np.mean(abs(diff_la))
128 
129             ########## End Linear Algebra LA method ##########
130 
131             # print("\n\n start DM : ",p,Dmax,'\n\n')
132 
133             ########## Start DM method ##########
134 
135             diff_dm = np.zeros(q)
136             for i in range(n):
137 
138                 k0 = abs(get_DM(GB,i)[0,0])
139                 k1 = abs(get_DM(GB,i)[1,1])
140 
141                 diff_dm[i] = k0 - k1
142 
143                 print("Difference DM   = ",k0 - k1,' and Sum = ',k0 + k1)
144 
145                 if ( k0 > k1 ):
```

```
                    Sol_dm_pr += '0'

                else:

                    Sol_dm_pr += '1'

            DM_confidence[p-1,Dmax-1] = np.mean(abs(diff_dm))

            ########## End DM method ##########

            # print("\n\n start PM : ",p,Dmax,'\n\n')

            ########## Start PM method ##########

            diff_prdm = np.zeros(q)
            for i in range(n):

                s_prdm, k0_prdm,k1_prdm = get_PM(GB,Sol_prdm_pr)
                Sol_prdm_pr += s_prdm

                diff_prdm[i] = abs(k0_prdm) - abs(k1_prdm)

                print("Difference PRDM   = ",k0_prdm - k1_prdm,)

            PRDM_confidence[p-1,Dmax-1] = np.mean(abs(diff_prdm))

            ########## End PM method ##########

            ########## Check for DM method ##########

            if (Sol_dm_pr in Sol):

                print('  DM SUCCESS for '+str(q)+tag+str(s)+' for D = '+str(Dmax)+' and p = '+
                qioa_dm[p-1,Dmax-1] = 0

            else:

                Eq = []
                for rs in range(num_of_Sols):

                    Eq.append(Hamming_dist(Sol_dm_pr,Sol[rs]))

                qioa_dm[p-1,Dmax-1] = min(Eq)

                print('  DM Not working for '+str(q)+tag+str(s)+' for D = '+str(Dmax)+' and p
```

```
192                 print('# of mismatched characters = ',qioa_dm[p-1,Dmax-1],'\n')
193 
194             ########## End Check for DM method ##########
195 
196             ########## Check for LA method ##########
197 
198             if (Sol_la_pr in Sol):
199 
200                 print('  LA SUCCESS for '+str(q)+tag+str(s)+' for D = '+str(Dmax)+' and p = '+
201                 qioa_la[p-1,Dmax-1] = 0
202 
203             else:
204 
205                 Eq = []
206                 for rs in range(num_of_Sols):
207 
208                     Eq.append(Hamming_dist(Sol_la_pr,Sol[rs]))
209 
210                 qioa_la[p-1,Dmax-1] = min(Eq)
211 
212                 print('  LA Not working for '+str(q)+tag+str(s)+' for D = '+str(Dmax)+' and p
213                 print('# of mismatched characters = ',qioa_la[p-1,Dmax-1],'\n')
214 
215             ########## End Check for LA method ##########
216 
217             ########## Check for PRDM method ##########
218 
219             if (Sol_prdm_pr in Sol):
220 
221                 print('PRDM SUCCESS for '+str(q)+tag+str(s)+' for D = '+str(Dmax)+' and p = '+
222                 qioa_prdm[p-1,Dmax-1] = 0
223 
224             else:
225 
226                 Eq = []
227                 for rs in range(num_of_Sols):
228 
229                     Eq.append(Hamming_dist(Sol_prdm_pr,Sol[rs]))
230 
231                 qioa_prdm[p-1,Dmax-1] = min(Eq)
232 
233                 print('PRDM Not working for '+str(q)+tag+str(s)+' for D = '+str(Dmax)+' and p
234                 print('# of mismatched characters = ',qioa_prdm[p-1,Dmax-1],'\n')
235 
236             ########## End Check for PRDM method ##########
237 
```

```
                ########## BEST of QIOA ##########

                if ((Sol_dm_pr in Sol) or (Sol_la_pr in Sol) or (Sol_prdm_pr in Sol)):

                    qioa_best[p-1,Dmax-1] = 0

                else:

                    qioa_best[p-1,Dmax-1] = min([qioa_dm[p-1,Dmax-1],qioa_la[p-1,Dmax-1],qioa_prdm

                ########## End BEST of QIOA ##########

                ########## Worst of QIOA ##########

                if ((Sol_dm_pr in Sol) and (Sol_la_pr in Sol) and (Sol_prdm_pr in Sol)):

                    qioa_worst[p-1,Dmax-1] = 0

                else:

                    qioa_worst[p-1,Dmax-1] = max([qioa_dm[p-1,Dmax-1],qioa_la[p-1,Dmax-1],qioa_prd

                ########## End BEST of QIOA ##########

                print('\n\n')

            except:

                print('SVD Error!! for D = '+str(Dmax)+' and p = '+str(p)+'!\n\n')

                qioa_dm[p-1,Dmax-1] = -1
                qioa_la[p-1,Dmax-1] = -1
                qioa_prdm[p-1,Dmax-1] = -1
                qioa_worst[p-1,Dmax-1] = -1
                qioa_best[p-1,Dmax-1] = -1

                raise
```

A.4 PRDM Method code

```
def get_PRDM(gamma_beta,sol):

    l = len(sol)

    g_b = gamma_beta.tensors
```

X

```
6
7       g_b = [tn.Node(g_b[x]) for x in range(n)]
8       g_bcon = [tn.conj(g_b[x]) for x in range(n)]
9
10
11      for i in range(l):
12
13          z = tn.Node(np.array([1.0,0.0]))
14          z_conj = tn.conj(z)
15
16          o = tn.Node(np.array([0.0,1.0]))
17          o_conj = tn.conj(o)
18
19          if (sol[i] == '0'):
20
21              g_b[i][1]^z[0]
22              g_b[i] = tn.contract_between(g_b[i],z,
23                                           output_edge_order = [g_b[i][0],g_b[i][2]])
24
25              g_bcon[i][1]^z_conj[0]
26              g_bcon[i] = tn.contract_between(g_bcon[i],z_conj,
27                                              output_edge_order = [g_bcon[i][0],g_bcon[i][2]])
28
29          else:
30
31              g_b[i][1]^o[0]
32              g_b[i] = tn.contract_between(g_b[i],o,
33                                           output_edge_order = [g_b[i][0],g_b[i][2]])
34
35              g_bcon[i][1]^o_conj[0]
36              g_bcon[i] = tn.contract_between(g_bcon[i],o_conj,
37                                              output_edge_order = [g_bcon[i][0],g_bcon[i][2]])
38
39
40          if (i != (l-1)):
41
42              g_b[i][1]^g_b[i+1][0]
43              g_bcon[i][1]^g_bcon[i+1][0]
44
45
46      if (l != 0):
47
48          GL_proj = tn.contractors.greedy(g_b[:l],
49                                          output_edge_order = [g_b[0][0],g_b[l-1][1]])
50
51          GLcon_proj = tn.contractors.greedy(g_bcon[:l],
```

```
                                        output_edge_order = [g_bcon[0][0],g_bcon[l-1][1]])

    else:

        GL_proj = tn.Node(np.array([1.0]).reshape((1,1)))
        GLcon_proj = tn.Node(np.array([1.0]).reshape((1,1)))


    for i in range(l+1,n):

        if (i != (n-1)):

            g_b[i][2]^g_b[i+1][0]
            g_bcon[i][2]^g_bcon[i+1][0]

        g_bcon[i][1]^g_b[i][1]


    if (l != (n-1)):

        GR_proj = tn.contractors.greedy((g_b[(l+1):] + g_bcon[(l+1):]),
                                        output_edge_order = [g_bcon[l+1][0],g_b[l+1][0],g_bcon[-1]|

    else:

        GR_proj = tn.Node(np.array([1.0]).reshape((1,1,1,1)))

    GLcon_proj[1]^g_bcon[l][0]
    GL_proj[1]^g_b[l][0]

    GLcon_proj[0]^GR_proj[2]
    GL_proj[0]^GR_proj[3]

    g_bcon[l][2]^GR_proj[0]
    g_b[l][2]^GR_proj[1]

    K = tn.contractors.greedy([GLcon_proj, GL_proj, g_bcon[l], g_b[l], GR_proj],
                              output_edge_order = [g_bcon[l][1],g_b[l][1]])

    k0_prdm = abs(K.tensor[0,0])
    k1_prdm = abs(K.tensor[1,1])

    s_prdm = ''

    if ( abs(k0_prdm) > abs(k1_prdm)):

```

```
98          s_prdm = '0'
99
100     else:
101
102         s_prdm = '1'
103
104     return s_prdm,k0_prdm,k1_prdm
```