

https://dspace.org/

# An efficiency comparison of quantum variational and classical satisfiability algorithms for debugging combinational logic circuits

**Demetriou, Peter**

**2021**
https://hdl.handle.net/10539/33833

Downloaded from a DSpace repository. https://dspace.org/



UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

Master of Science in Engineering

April 11, 2022

# An Efficiency Comparison of Quantum Variational and Classical Satisfiability Algorithms for Debugging Combinational Logic Circuits

Peter Demetriou

*A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg in fulfilment of the requirements for the degree of Master of Science in Engineering.*

# Declaration

I declare that this dissertation is my own unaided work, except where otherwise stated. It is being submitted for the degree of Master of Science in Engineering to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other university.



(Signature of Candidate)

On the $22^{nd}$ day of December 2021

# Abstract

This research presents an extension and contribution to combinational logic circuit debugging for future use in Very Large-Scale Integration system design. It also extends application of quantum computation to an engineering problem. Although previous work in this area has sped up approximate debugging solutions through the use of satisfiability solvers, the exact solutions become classically intractable as the number of gates in the system increases. Additionally, whilst many quantum algorithms show promise in solving classically intractable problems in theory, there have been few domain-specific implementations, particularly where hard and soft constraints exist, multiple solutions are required and noisy intermediate-scale quantum hardware is used. Two variational quantum algorithms are applied to a reduction of the satisfiability problem into a maximum independent set problem. While the Quantum Alternating Operator Ansatz approach constrains the optimisation to feasible solutions, the resulting ansatz is too large for current devices. A custom ansatz for a Variational Quantum Eigensolver along with a solution enumeration method is however able to make use of current hardware for the debugging problem. A small problem set is generated and solved using the aforementioned methods. The accuracy and complexity of the algorithms are benchmarked against an exact and approximate classical solver. The quantum approach outperforms the exact solver in both temporal and spatial scaling. With a theoretical mathematical result, it is argued that the quantum approach will likely outperform the approximate solver in the same way for future larger problems when quantum hardware size increases and becomes more robust to noise.

# Acknowledgments

I would like to thank Professor Kenneth J. Nixon for his guidance and encouragement.

I would also like to thank

- the Department of Science and Innovation for funding on behalf of the South African Quantum Technology Initiative.
- the University for their subscription to the International Business Machines Corporation quantum hardware.
- Qiskit, Pennylane, PySAT and NetworkX for use of, or inspiration from, their Python modules.

Thank you to my friends and family for their continued interest in my work and my wife for her encouragement to pursue further study.

*Soli Deo Gloria*

# Contents


**1 Introduction** **1**

1.1 Structure of the Dissertation . . . . . 2

**2 Background** **6**

2.1 Contribution of Dissertation . . . . . 6

2.2 Scope of Dissertation . . . . . 7

2.3 The Research Question and Objectives . . . . . 8

2.4 Hard Problems in Computing . . . . . 8

2.5 Debugging . . . . . 9

2.6 Graph Problems . . . . . 11

2.6.1 APX-Complete Problems . . . . . 12

2.6.2 Minimal, Maximal, Minimum and Maximum . . . . . 12

2.7 Quantum Computing . . . . . 13

2.7.1 The Ising Model . . . . . 13

2.7.2 Adiabatic Quantum Computing . . . . . 15

2.7.3 Gate Model Quantum computing . . . . . 18

2.7.4 Phase Estimation and Grover's Algorithm . . . . . 18

2.7.5 The Noisy Intermediate-Scale Quantum Era . . . . . 19

2.7.6 The Variational Quantum Eigensolver . . . . . 21

2.7.7 The Quantum Approximate Optimisation Algorithm . . . . . 22

2.7.8 The Quantum Alternating Operator Ansatz . . . . . 22

2.8 Chapter Summary . . . . . 24

**3 Methodology** **26**
3.1 Motivation . . . . . 26
3.2 Proposed Methodology . . . . . 27
3.3 Chapter Summary . . . . . 28

**4 Methods** **30**
4.1 Structure of Approach . . . . . 30
4.1.1 Pipeline of Transformations . . . . . 32
4.2 Classical Preprocessing . . . . . 34
4.2.1 Tseitin Transforms . . . . . 34
4.2.2 Independent Set . . . . . 36
4.2.3 Graph Constraints . . . . . 37
4.3 Quantum Optimisation . . . . . 38
4.3.1 Classical and Quantum Hybrid Approaches . . . . . 38
4.3.2 Bit Driver Hamiltonian . . . . . 41
4.3.3 Penalty and Reward Hamiltonians . . . . . 41
4.3.4 Edge Driver Hamiltonian . . . . . 42
4.3.5 X Mixer Hamiltonian . . . . . 43
4.3.6 Bit Flip Mixer Hamiltonian . . . . . 43
4.3.7 Quantum Approximate Optimisation Algorithm . . . . . 44
4.3.8 Quantum Alternating Operator Ansatz . . . . . 44
4.3.9 Variational Quantum Eigensolver . . . . . 47
4.4 Classical Postprocessing . . . . . 48
4.4.1 Bit Strings to Maximum Independent Set . . . . . 48
4.4.2 Maximum Independent Subset to Minimal Correction Subset Clauses . . . . . 49
4.4.3 Minimal Correction Subset Clauses to Minimal Correction Subset Gates . . . . . 49
4.5 Chapter Summary . . . . . 49

**5 Application** **50**
5.1 Problem Generation . . . 50
5.1.1 Circuits . . . 50
5.1.2 Specifications . . . 54
5.1.3 Gate Errors . . . 55
5.1.4 WDIMACS Format . . . 58
5.1.5 Graphs . . . 60
5.2 Adaptive Hamiltonians and Ansatz . . . 62
5.3 Quantum Circuit Transpilation . . . 64
5.3.1 Implemented Reductions . . . 65
5.4 Noise Models . . . 69
5.5 Parameter Optimisers . . . 71
5.6 Chapter Summary . . . 72

**6 Results** **74**
6.1 Overview . . . 74
6.2 Problem Encoding . . . 75
6.3 Scaling of Solver Solutions . . . 76
6.3.1 Classical Solvers . . . 76
6.3.2 Hybrid Algorithm Components . . . 77
6.3.3 Memory . . . 78
6.3.4 Quantum Algorithm Comparison . . . 78
6.4 Quantum Resources . . . 82
6.5 Accuracy . . . 84
6.5.1 Eigenvalues . . . 85
6.5.2 Error Calculations . . . 87
6.6 Chapter Summary . . . 89

**7 Discussion** **91**
7.1 Problem Encoding . . . 91
7.2 Scaling of Solver Solutions . . . 92

7.2.1 Classical Solvers . . . 93
7.2.2 Hybrid Algorithm Components . . . 93
7.2.3 Memory . . . 93
7.2.4 Quantum Algorithm Comparison . . . 93
7.3 Quantum Resources . . . 94
7.4 Accuracy . . . 95
7.4.1 Eigenvalues . . . 97
7.4.2 Error Calculations . . . 97
7.5 Unexplained Results . . . 98
7.6 Chapter Summary . . . 98

**8 Conclusion** **100**
8.1 Contribution of Dissertation . . . 101
8.2 Future Recommendations . . . 101

**A Source Code** **119**
A.1 Programming Approach . . . 119
A.1.1 Problem Generation . . . 120
A.1.2 Cirsat Module . . . 120
A.1.3 Preprocessing . . . 120
A.1.4 Postprocessing . . . 126
A.1.5 Qcircsat Module . . . 128
A.1.6 Qtools . . . 129
A.1.7 Drivers . . . 135
A.1.8 Mixers . . . 138
A.1.9 Constraints . . . 140
A.1.10 Problems . . . 141
A.1.11 Placement of Vertices in Graph Drawing . . . 142

**B Conference Paper** **143**

**C The Manhattan QPU** **150**

C.1 IBM Manhattan QPU . . . . . 150
C.2 Error Calibration Data . . . . . 151

# List of Figures

2.1 Illustration of a system of bar magnet configurations and the associated energy of the configurations. The local minimum energy A, is caused by external magnetic influences whilst the global minimum B, is a reachable state when the system is free of constraints. This can be modelled by a classical Ising model. . . . . 15

4.1 A high level overview of the structure of approach to solving the debugging problem. The arrows joining the blocks, or representation of the problem, represent outputs of the blocks unless they are numbered in which case they are transformations from one block into the next. The blocks are grouped into problem generation, solvers and classical pre and post processing. . . . . 32

4.2 A small example of a combinational logic circuit that the solver approach can easily be illustrated on. . . . . 34

4.3 A graph with one coloured maximum independent set, corresponding to a solution of the circuit example in figure 4.2 with all its inputs and outputs set to true as the constraining test vector. . . . . 37

4.4 A small parametrised quantum circuit to illustrate hybrid quantum computation. The circuit can be classically optimised to find the lowest expectation value measured in the Pauli $\sigma_z$ basis. . . . . 38

4.5 Two functionally equivalent quantum circuits. The right circuit decomposes the Toffoli gates of the left circuit. The decomposition is into gates commonly available in many quantum computing software packages. The quantum circuits are sub-mixer circuits that could appear in a QAOA ansatz. . . . . 46
4.6 A partially decomposed QAOA ansatz without any repetitions ($p = 1$). The initial state, phase and sub-mixer components are separated by barriers. Some mild optimisation that leaves out sub-mixers concerning already determined graph colours are left out. This ansatz is used to solve the debugging problem in figure 4.2 with the test vectors all set to true. . . . . 47
4.7 A simple ansatz used for VQE problems with one layer of rotation gates and one layer of linear entanglement. . . . . 48

5.1 A one-to-two line binary decoder circuit for use as a data point in the generated circuit problem set. . . . . 52
5.2 A half adder circuit for use as a data point in the generated circuit problem set. . . . . 52
5.3 A half subtractor circuit for use as a data point in the generated circuit problem set. . . . . 52
5.4 A two-to-one multiplexer circuit for use as a data point in the generated circuit problem set. . . . . 53
5.5 A full adder circuit for use as a data point in the generated circuit problem set. . . . . 53
5.6 A two-to-four line binary decoder circuit for use as a data point in the generated circuit problem set. . . . . 54
5.7 A full subtractor circuit for use as a data point in the generated circuit problem set. . . . . 54
5.8 A full subtractor circuit with an erroneous gate implementation set. The erroneous gate implementation is to be located by the debugging solvers. . . . . 56

5.9 A graph structure on which solving the maximum independent set problem encodes the solution to the debugging problem on the full subtractor circuit in figure 5.7 . . . . . 62
5.10 A graph structure on which solving the maximum independent set problem encodes the solution to the debugging problem on the half subtractor circuit in figure 5.3 . . . . . 66
5.11 A transpiled custom ansatz corresponding to the half subtractor graph in figure 5.10 with localised circular entangling layers to be used with the VQE solver. . . . . 68
5.12 A partially optimised ansatz corresponding to the half subtractor graph in figure 5.10 with no repetitions to be used with the QAOA solver. . 70

6.1 The relationship between graph vertices, or the number of qubits, as the number of classical circuit gates increase. . . . . 75
6.2 Comparison of the averaged runtime of classical simulations of the quantum solvers and the exact classical solver. The time is normalised per solver so the scaling rather than the speed of the solvers are apparent. 76
6.3 A comparison of the average total optimiser time of the VQE solver and the average time on the QPU as the problem size increases. . . . 78
6.4 A Comparison of the averaged runtime between the exact eigensolver and VQE. The time is normalised per solver so the scaling rather than the speed of the solvers are apparent. The continuous functions are fit based on the shape of the discrete data before normalisation. . . . . 79
6.5 A Comparison of the averaged runtime between the exact eigensolver and VQE and the approximate classical solver M22. The time is normalised per solver so the scaling rather than the speed of the solvers are apparent. The continuous functions are fit based on the shape of the discrete data before normalisation. . . . . 80

6.6 A Comparison of the averaged runtime between the exact eigensolver and VQE and the approximate classical solver M22. The time is normalised per solver so the scaling rather than the speed of the solvers are apparent. The continuous functions are fit based on the shape of the discrete data before normalisation. . . . . 81
6.7 A Comparison of the averaged runtime between the exact eigensolver and VQE and the approximate classical solver M22. The speed of the solvers are apparent as well as the scaling since the data are shifted to start at the same smallest runtime point. . . . . 82
6.8 The maximum number of `CNOT` and parametrised quantum gate resources required by each of the custom ansatz as the size of the problem increases. . . . . 83
6.9 Comparison of maximum ansatz depth for the VQAs as the size of the problem increases. . . . . 84
6.10 Classical simulations of quantum solvers and exact eigensolver average energy expectation values as the problem size increases. The expectation values of the optimiser over the ansatz and the most sampled solution (the approximate value) are compared to the exact and lowest expectation values for the simulations of the quantum solvers. . . . . 86
6.11 A comparison of VQE average energy expectation values as the problem size increases. The expectation values of the optimiser over the ansatz and the most sampled solution (the approximate value) are compared to the exact and lowest expectation values when classical exact solutions exist . . . . . 87
6.12 The error factors (larger values are better) for the maximum independent set as the as the size of the problem increases for the VQE solver. . . . . 89

7.1 The optimiser energy plotted against the optimiser evaluations for the first run on the two-to-one multiplexer with an error in the first `NAND` gate and second row of the truth table as the test vector. . . . . 96

7.2 The output bit string probability distribution for the Pennylane implementation of QAOA at differenet levels of ansatz repetitions. . . . . 98

C.1.1 The Manhattan QPU topology represented as a graph. . . . . 151

# List of Tables

4.1 Tseitin transformations for all of the common logic gates. . . . . 34
4.2 Definitions for the mathematical notation used to describe the graph structures optimised in the quantum circuits. . . . . 40

5.1 Truth table for a full subtractor combination logic circuit. Using numeric notation for symbols. . . . . 55

C.1 Error calibration data for IBM Manhattan. . . . . 152

# Acronyms

**APX-Complete** Approximable Polynomial Time Complete. 12, 24

**ASIC** Application Specific Integrated Circuit. 1

**BDD** Binary Decision Diagram. 10

**CAD** Computer Aided Design. 1

**CIRCUIT-SAT** Circuit Satisfiability. 9

**CMOS** Complementary Metal-Oxide-Semiconductor. 11

**CNF** Conjunctive Normal Form. 9, 28, 32–36, 49–52, 60–62, 80, 102, 121, 126

**COBYLA** Constrained Optimization By Linear Approximation. 71

**CPU** Central Processing Unit. 31

**FFT** Fast Fourier Transform. 8

**HDL** Hardware Description Language. 1

**HHL** Harrow-Hassidim-Lloyd. 20

**IO** Input-Output. 1, 2, 9, 28, 31, 33, 35, 37, 45, 120, 121

**IS** Independent Set. 36, 45, 47, 60, 88

**ISCAS-85** International Symposium of Circuits and Systems 1985 Benchmark Suite. 10, 55

**M22** MiniSAT 22. xi, xii, 31, 51, 59, 79–82, 84, 88, 90, 92–94, 96, 98–100

**MAX-SAT** Maximum Satisfiability. 10, 28, 31, 36, 58, 92, 96

**MCSC** Minimum Correction Subset Clause. 10, 28, 33, 49, 84, 88, 92, 93, 120, 126

**MCSG** Minimum Correction Subset Gate. 31–33, 49, 126

**MiniSAT** Minimalistic, Open-Source Satisfiability Solver. 31, 33

**MIS** Maximum Independent Set. 22, 28, 31, 33, 36, 38, 41–44, 46, 48, 49, 51, 60, 63, 84, 85, 87, 88, 95, 96, 98, 102, 128, 129, 141

**MLCSC** Minimal Correction Subset Clause. 10, 33, 84, 88, 92, 94, 96, 120

**MLCSG** Minimal Correction Subset Gate. 31

**MLIS** Maximal Independent Set. 11, 92

**MSC** Maximum Satisfiable Subset Clause. 28

**NISQ** Noisy Intermediate-Scale Quantum. 2, 8, 18–20, 22, 24, 27, 51, 55, 65, 69, 83, 87, 90, 94–97, 100

**NP-Complete** Nondeterministic Polynomial Time Complete. 4, 8, 9, 12, 18, 27, 28, 36, 101, 102, 119

**NP-Hard** Nondeterministic Polynomial Time Hard. 2, 9, 12–14, 17, 23, 64

**P** Polynomial Time. 12, 18, 36, 90

**PMAX-SAT** Partial Maximum Satisfiability. 2, 28, 29, 59, 72, 121

**QAOA** Quantum Alternating Operator Ansatz. x, xi, xiii, 4, 5, 7, 22–24, 27, 28, 32, 39, 43, 44, 46, 47, 65, 66, 69, 70, 82, 90, 91, 94, 95, 98–102, 141, 143

**QN-SPSA** Quantum Natural Simultaneous Perturbation Stochastic Approximation. 72, 73, 77, 79

**QPE** Quantum Phase Estimation. 18, 24

**QPU** Quantum Processing Unit. xi, xiii, 4, 5, 10, 13, 18, 28, 32, 50, 64, 65, 71, 77, 78, 83, 85, 93, 99, 150, 151

**QUBO** Quadratic Unconstrained Binary Optimisation problem. 11, 16, 17

**RTL** Register-Transfer Level. 1

**SAT** Satisfiability (or Satisfiable). 2, 3, 6–11, 17, 18, 23, 24, 28, 34–37, 49, 51, 57–59, 95–97, 102, 126

**SPSA** Simultaneous Perturbation Stochastic Approximation. 71, 72

**UAV** Unmanned Aerial Vehicle. 11

**UNSAT** Unsatisfiable. 9, 28, 38, 49, 60, 96, 97

**VC** Vertex Cover. 22, 36, 42, 44

**VLSI** Very Large-Scale Integration. 1, 5, 100, 101, 143

**VQA** Variational Quantum Algorithm. xii, 3, 4, 6, 19, 27, 30, 31, 33, 38, 50, 63–66, 71, 76, 77, 84, 85, 93

**VQE** Variational Quantum Eigensolver. x–xii, 4, 7, 19, 21, 24, 27, 28, 47, 48, 66–69, 72, 77–85, 87, 89–95, 99–101, 141, 150

**WCNF** Weighted Conjunctive Normal Form. 31, 33, 120, 121

**WDIMACS** Center for Discrete Mathematics and Theoretical Computer Science weighted format. 33, 56, 58, 59, 120, 128

**WMAX-SAT** Weighted Maximum Satisfiability. 58

**WMIS** Weighted Maximum Independent Set. 11

# Chapter 1

# Introduction

Very Large-Scale Integration (VLSI) uses hardware description languages before manufacturing to verify the design of the integrated circuits. Estimates report that verification and the subsequent debugging task take up to 70 % of the VLSI design process and that 60 % of Application Specific Integrated Circuits (ASICs) fail due to functional errors [1]. A debugging solution forming part of Computer Aided Design (CAD) software that scales can, therefore, decrease the manufacturing cost and time of quality products. VLSI uses millions of logic gates and the systems continue to grow even larger, research is therefore conducted on finding powerful debugging solutions.

Various faults may cause the circuitry to not function as intended such as physical defects in components. There are many fault diagnosis models employed in VLSI design such as the stuck-at, bridging, open or break, fault models [2]. Furthermore, advanced modeling of faults have been used for reversible and quantum circuits [3, 4, 5, 6, 7]. This research presents a model-free logic debugging approach to finding errors. The errors are found in the early stages of the VLSI design cycle, in the Hardware Description Language (HDL) and Register-Transfer Level (RTL) stages, before physical implementation. Given a circuit specification along with expected Input-Outputs (IOs) pairs this approach finds gate implementation errors along with

their locations. These sorts of errors are often caused by specification changes, bugs in automated software and human error [8].

Automated debugging solutions for multiple fault detection in combinational logic circuits encoded as Satisfiability (or Satisfiable) (SAT) problems have been proposed, and extended to Partial Maximum Satisfiability (PMAX-SAT) problems for IO constraints [8, 9, 10]. However, the solution search space for these problems increases exponentially as the circuit size increases since SAT is in the complexity class NP-Hard [11]. This makes SAT classically intractable for large problems [12]. If an approach is able to make the problems tractable it is worthwhile pursuing it.

There are quantum computing algorithms that provide an exponential speed-up over the best known classical alternatives [13, 14, 15, 16]. These algorithms cannot be efficiently simulated on a classical device, suggesting a quantum advantage for certain problems [17, 18, 19]. Although quantum computing currently has limitations concerning software and hardware, there is fruitful research to be performed on the available Noisy Intermediate-Scale Quantum (NISQ) devices [20, 21]. These devices are prone to error, but classical-quantum hybrid algorithms called variational algorithms, that implement a cost function dependent on variable quantum gate parameters, do show immediate promise [22, 23].

This research provides methods for solving the debugging problem with classical-quantum hybrid algorithms and benchmarking the scaling of these algorithms against exact and approximate classical alternatives.

## 1.1 Structure of the Dissertation

This section gives a high-level overview of the rest of the dissertation. Chapter 2 is the first main part of the dissertation and introduces problems in computation along with the specific problem and the motivation behind engineering a solution for it. It discusses previous approaches to the problem as well as some approaches in the field of quantum computing that are of interest to a new approach to the problem. It

also discusses the intended scope and contribution of the research. The second major part involves chapters 3 to 5, which focuses on the reasoning behind the methods with a guided explanation and some of the theory required for understanding the approach. Finally, some specific implementation details are explicated. The final part; chapters 6 to 8 presents some of the data of the investigation and discusses implications of the data. Appendices A to B supply ancillary information.

Summaries of chapters 2 to 8 along with appendices A to B are provide below:

Chapter 2: This chapter starts with the intended contribution and scope of the research, it then introduces a number of background topics for understanding the work. First the concept of hard problems in computing is introduced, followed by the problem under consideration in the field of electrical engineering. Previous classical SAT based solutions of the debugging problem are discussed. A discussion of optimisation problems over graphs is given. Quantum computing is presented as a method for efficiently solving the debugging problem. An Ising problem is built on to introduce Variational Quantum Algorithms (VQAs). Some discussion on existing optimisation approaches in quantum computing is presented. The current state of quantum computing and the currently applicable algorithms are introduced.

Chapter 3: This chapter introduces the reasoning behind the choice to use quantum computing algorithms for the problem. The proposed methodology for the debugging problem is discussed, providing a high-level outline of the reasoning to the methods in the research.

Chapter 4: This chapter starts with showcasing the structure of the approach in the application of this research. Using this structure it separates the methods into classical pre-processing and post-processing sections sandwiching the intermediate quantum process. The classical preprocessing contains the transforms required to solve the debugging problem using VQAs. The quantum approach discusses the idea behind VQAs and then the mathematical structures, and their quantum circuit representations they operate on. Finally, the classical postprocessing shows how the resultant sampled output distribution of the quantum algorithms provide a set of

suspect gate errors.

Chapter 5: This chapter describes the problem generation approach. As well as how the quantum circuits are built to suit the problem on a Quantum Processing Unit (QPU). Further discussions concerning the ansatz optimisation and simulations of noise are provided. Finally, the optimisation process of the VQAs is compared.

Chapter 6: This chapter summarises the results from the runs of the various algorithms as graphical comparisons. It presents the data as summaries of the individual runs. The results are given for scaling of the problem encodings, algorithm run times and spatial elements of the ansatz. Lastly, the accuracy of the solvers is summarised.

Chapter 7: The structure of the discussion follows the same structure of chapter 6 and comments on its contents. The focus is on what can be said from chapter 6, due to the nature of the problems there are not many data points and inference to the best explanation of the data are required. Where improvements naturally follow from the results they are mentioned. Distinctions to separate the approximate classical solver and the Variational Quantum Eigensolver (VQE) are made. The temporal and spatial requirements of the algorithms are discussed. The merit of the solvers as current solutions are also discussed, as well an unexplained result from a previous approach to the Quantum Alternating Operator Ansatz (QAOA). Conditions under which the quantum algorithm may obtain an advantage are mentioned.

Chapter 8: Summaries of the findings are made and contributions are presented along with recommendations for future work.

Appendix A: This appendix contains the source code of the modules developed for implementation of the various algorithms. The Python modules are not specific to this work and may be used to implement the quantum algorithms for many Nondeterministic Polynomial Time Complete (NP-Complete) problems and with a little adjustment all of them. It also includes the modules for the circuit debugging problem that may be used with any classical solvers and is used to set up and transform the problem for the quantum solutions. The scripts that were used to run

the specific trials of the algorithms in this dissertation can be found on the project git pages: qcircsat and circsat.

Appendix B: This conference paper is published by the 2021 IEEE Computer Society Annual Symposium on VLSI. It represents a snapshot of the dissertation at an earlier stage where QAOA was the focus.

Appendix C: This appendix contains information on the specific QPU used. It contains the topology of the hardware as well as the calibration data taken before the runs.

# Chapter 2

# Background

This chapter starts with the intended contribution and scope of the research, it then introduces a number of background topics for understanding the work. First the concept of hard problems in computing is introduced, followed by the problem under consideration in the field of electrical engineering. Previous classical SAT based solutions of the debugging problem are discussed. A discussion of optimisation problems over graphs is given. Quantum computing is presented as a method for efficiently solving the debugging problem. An Ising problem is built on to introduce Variational Quantum Algorithms (VQAs). Some discussion on existing optimisation approaches in quantum computing is presented. The current state of quantum computing and the currently applicable algorithms are introduced.

## 2.1 Contribution of Dissertation

The research area, in general, is new and there is not much on application-specific quantum computation although the field is moving forward quickly and more applications have come to fruition since the inception of the original idea. The dissertation provides methods to transform the debugging problem into a usable quantum circuit and get the result from the computation. The transformations are general enough

to be applied to other problems and to support different avenues of quantum computation. The debugging problem is a problem in industry and it also fills some perceived gaps in the literature where the source material, on how Tseitin transforms are performed, is not accessible.

An analysis of the run times of the various algorithms along with the accuracy achievable is given. It is shown that the quantum approaches can provide accurate results and scale in sub-exponential time. The scaling is comparable to a classical approximate solver and is expected to provide scaling advantages over even the classical approximate solver as the quantum hardware size increases and becomes more robust to noise.

This approach also has the virtues of being empirically accurate for deducing gate implementation errors and for guiding new research for circuit debugging and quantum approaches to engineering applications.

## 2.2 Scope of Dissertation

The scope of the dissertation is to detail the preparation of VQE and QAOA such that a SAT based debugging approach can be run on quantum hardware. These runs record data about the accuracy and run times of the algorithms to compare the accuracy and efficiency in scaling to classical algorithms.

An exact classical solver is used to benchmark the accuracy of the solutions for all the other solvers in addition to demonstrating the infeasibility of exactly solving the problem classically. A very fast classical approximate solver is used as another point of comparison to the quantum algorithm scaling and accuracy.

This work is not intended to provide a rigorous mathematical proof of the temporal or spatial complexity of the algorithms but rather to provide some evidence towards the idea that quantum algorithms can overcome the intractability of the classical algorithms for the debugging problem.

The methods to encode and transform the problem structure in various ways to make it applicable to the various solvers, and to adjust the standard solvers such that they are feasible for the hardware to be used, is not the primary goal of the dissertation. However many new approaches are used to do this, and it may serve as starting point for improved and different approaches to solving NP-Complete problems on NISQ devices. Furthermore, they are required to arrive at a position where the comparisons and main goal of the research can be pursued.

Lastly, the main focus is on quantum computation and its application to the debugging problem and not on the application of optimal SAT solving methods or optimal debugging methods.

## 2.3 The Research Question and Objectives

Positively stated, this research seeks to answer the following question: can the use of quantum computation solve a circuit debugging problem more efficiently than without it? The objectives are to develop methods that can solve the circuit debugging problem using current quantum algorithms and benchmark these methods' efficiency against the best known classical approach.

## 2.4 Hard Problems in Computing

Algorithm speed in computing is often measured by how many operations need to be performed in the worst-case scenario, as a function of the input size of the problem operated on. The standard by-hand matrix multiplication algorithm requires $O(n^3)$ operations. By comparison the Fast Fourier Transform (FFT) has a time complexity of $O(n \log n)$. For larger sizes of $n$ efficient algorithms grow more slowly and are practical to run. The memory required for computation as problems grow is another important factor to consider for algorithm efficiency.

Exhaustive algorithms that search through all candidate solutions, discarding results that are not solutions and continuing the search often lead to time complexities

of $O(2^n)$ or worse. An exact algorithm that searches every solution for a graph colourability problem, where the goal is to decide if colouring a graph with $n$ vertices and $k$ colours such that adjacent vertices never have the same colour is possible, finishes in $O(k^n)$ time. There are an entire class of problems for which no sub-exponential time algorithms are known. These problems are known as Nondeterministic Polynomial Time Hard (NP-Hard) problems. There is a further subset of these problems where all the problems are reducible to another version of the problem, these are known as Nondeterministic Polynomial Time Complete (NP-Complete) problems [11]. The main focus of this work is on such a problem and gathering data to see if quantum algorithms *on quantum machines* fair better than their classical counterparts.

## 2.5 Debugging

Circuit Satisfiability (CIRCUIT-SAT) is the decision problem of determining if a logic circuit has possible input values that result in a true output. If there is such an assignment the circuit is SAT, otherwise it is Unsatisfiable (UNSAT). There is an algorithm that solves this problem in $O(2^{0.4058m})$ where $m$ is the number of gates in the problem [24]. A more general Conjunctive Normal Form (CNF) SAT problem can be solved in $O(2^{0.29m})$ where $m$ is the number of clauses in the problem [25].

This research differs in that it is not merely a decision problem but locates errors in the implementations assuming that the circuit is UNSAT. However, it may also be applied to SAT problems, in which case the output is just the empty set of gate errors. Additionally, the circuits in question have specific IO specifications that vary on a case by case basis that is dependent on their application. This assumes knowledge of the circuit behaviour beforehand and makes the solution dependent on whether or not good test vectors are used. Some tests vectors may by chance work as expected for an incorrectly implemented circuit. As an illustration to see why this is the case consider the `AND` and `OR` logical operators. They share overlap in half of the possible configurations of their truth tables and therefore if they are incorrectly substituted in place of each other they will provide equivalent outputs for half of the possible

inputs. The International Symposium of Circuits and Systems 1985 Benchmark Suite (ISCAS-85) combinational circuits are used as problems to benchmark solvers to these sorts of problems, but these circuits are too large for current Quantum Processing Units (QPUs) so part of the research is generating benchmark circuits [26, 3, 27].

Debugging strategy using Boolean SAT that solves multiple design errors and constitutes a model-free debugging approach has been achieved classically [8]. This work assumes a netlist with test vector specifications is available. This debugging occurs before the physical fault diagnosis. The clauses in the Boolean formula correspond to lines in the truth table of the circuit. The result encodes the location of the fault as well as the cardinality of the fault, that is how many clauses are implicated in the Minimal Correction Subset Clauses (MLCSCs). Additional heuristics, to reduce memory usage and run time, were also considered.

The error cardinality refers to the maximum amount of erroneous gates in the solution. To find the Minimum Correction Subset Clauses (MCSCs) all the solutions up to the error cardinality need to be enumerated. Traditionally simulation path tracing over Binary Decision Diagrams (BDDs) are used for this, but BDDs often scale exponentially in size if a poor variable ordering is used. MLCSCs are shown to map directly to possible error clauses. The error clause solution space grows exponentially creating a space complexity problem to overcome, therefore a MAX-SAT and approximate MAX-SAT two-step analysis is used [9]. This indicates solutions get less helpful as the size grows since there are exponentially many places the issues could potentially be.

Methods of enumerating more solutions from the current solution is a common approach in SAT solvers [10]. However one MCSC is sufficient since enumeration of MLCSCs is for the purpose of finding the MCSC.

The preceding SAT approaches to debugging combinational logical circuits form the classical basis of the problem for which this work seeks to leverage quantum computation. It also shows the formal logical representations required for SAT debugging of the logic circuits (page 34 showcases how the transformations are

achieved) as well as providing a discussion of the significance of SAT for debugging applications.

## 2.6 Graph Problems

Graph problems study the pairwise relationships between vertices or nodes representing some discrete variable. Whether or not a relationship exists is visualised with a connecting edge between the vertices. Graphs are useful in representing combinatorial problems with $k$-ary variable assignments which can be represented by colourings over $n$ vertices. Maximising or minimising the number of assignments due to some constraints can then represent an optimisation problem.

Combinatorial optimisation problems have many practical applications in engineering. A Maximal Independent Set (MLIS) problem may be reduced to a graph colouring problem for application in frequency allocation for a cellular network [28]. A message scheduling problem encoded as an Weighted Maximum Independent Set (WMIS) problem can be solved using a classical Ising formulation [29]. Graph colouring problems have been used as abstractions of parallel computation to optimise scheduling of tasks so that maximum parallelism is achieved [30]. Graph colouring has also been used to schedule classes [31]. A knapsack problem may be used to reduce energy consumption for a smart grid whilst considering desired appliance usage as constraints [32]. A graph colouring framework has been used to allocate power resources for a communication application [33]. Using a QUBO form to model the facility location problem has an application to Unmanned Aerial Vehicle (UAV) connections to the shortest distanced wireless node [34]. A classical Ising mapping of a rectangle packing problem has been solved on a Complementary Metal-Oxide-Semiconductor (CMOS) annealing machine [35].

The practical applications provide methods to cast these problems in familiar mathematical representations. They also build a conceptual set of what sorts of engineering problems these mathematical representations often arise in. Helpfully, classical Ising models of these applications are as ubiquitous as Ising models are in quantum com-

puting and therefore form a bridge between the engineering problems and quantum computing (see page 13 for a discussion of the Ising model) [36, 37].

### 2.6.1 APX-Complete Problems

There are a class of problems that have no known Polynomial Time (P) algorithms, however, there can be approximate algorithms that solve them in P. The solutions to the problems can also be verified in P. NP-Complete problems have the property that solving any one of them allows one to find the solution to any of the other problems in the set in P. Approximable Polynomial Time Complete (APX-Complete) problems are the set of the aforementioned problems that can be efficiently solved using a P algorithm up to some approximation factor.

The main problem under focus in this work is APX-Complete and admits efficient but approximate solutions. The approximation ratio is dependent on the specific instance of the problem in question, such as the maximum degree of the graph and other properties.

### 2.6.2 Minimal, Maximal, Minimum and Maximum

It is important to note the difference in nomenclature concerning solving optimisation problems over graphs. Maximal sets are sets that are solutions for a particular problem such that if one adds any elements to the set it no longer constitutes a solution with the stipulated properties of the problem. There may be more than one such maximal set, the superset containing a set of maximal solutions to the problem, the largest element in the set being the maximum set. Minimal solutions are defined such that removing any element in the set violates some stipulated property of the problem in question. As in the case of the maximal counterpart the smallest of such minimal sets is the minimum set. The maximal and minimal solutions to graph problems do have tractable classical solutions and it is the enumeration of all of these solutions to find the maximum and minimum solution that places the problem in the NP-Hard class.

## 2.7 Quantum Computing

Some quantum algorithms are known to scale better than classical algorithms. The scaling of an algorithm is different to how fast the problem is completed. It may well be the case for smaller systems that an algorithm that scales badly may complete faster than a superior algorithm. Even if more operations are required for the badly scaling algorithm the hardware may complete the operations much faster than the superior algorithm run on faster hardware. However, if the system size is incremented by a modest size it may not matter how much faster the hardware is. In the case of QPUs, the present hardware is in its infancy and often completes problems at a slow rate even disregarding the networking and queueing times involved in the current cloud-based model.

What is sought is a quantum algorithm that scales better than its classical counterpart. This comparison can be made if the time taken for a problem to complete on the different hardware is normalised or if a mathematical proof of the time complexity of the algorithms in question is given.

### 2.7.1 The Ising Model

The Ising model is a Hamiltonian that models interactions of magnetic dipoles as spin variables in a lattice that was originally solved in a single dimension by its namesake [38]. These dipoles are present in amorphous magnetic materials such that the spins do not exhibit a periodic structure. The amorphous magnetic structure is analogous to the amorphous structure of glass and therefore the material is often referred to as a spin glass. This means the spin orientation fluctuates randomly. However, at some critical temperature close to zero Kelvin the spins are able to reach their lowest energy state [39]. For a two dimensional planar lattice in a magnetic field or for a three dimensional planar lattice, finding the ground state energy is known to be NP-Hard [40]. A quantum mechanical version of the Hamiltonian known as the Transverse-Field Ising Model in equation (2.1), models quantum spins in the presence of external magnetic fields. This causes the neighbouring spins, signified by $< i, j >$

to align or anti-align and reach local or global minimum energy states.

$$\hat{H} = -\sum_{<i,j>} J_{ij}\hat{\sigma}_i^Z\hat{\sigma}_j^Z - \sum_i h_i\hat{\sigma}_i^Z - \sum_i g_i\hat{\sigma}_i^X \tag{2.1}$$

In the following example the lowest energy state is assumed to be antiferromagnetic rather than ferromagnetic. Suppose there is a system containing bar magnets next to each other, represented by the arrows in figure figure 2.1, in this system the lowest energy state, each magnet anti-aligning, will eventually be reached. If they have differing magnetic coupling strengths and there is a strong external field around them some of the magnets may align with their neighbours. The first two terms of the Transverse-Field Ising model describe this system. Counting the magnets that have north up and multiplying by minus one for energy minimisation, the ground state reached can be a local minimum. This is a hard problem for nature to resolve and many problems which are NP-Hard can be represented by the Ising model. Quantum implementations can tunnel through the local minimum energy barrier. A transverse field that represents superposition is added as the third term and the resultant model is what is implemented on quantum annealers [41].

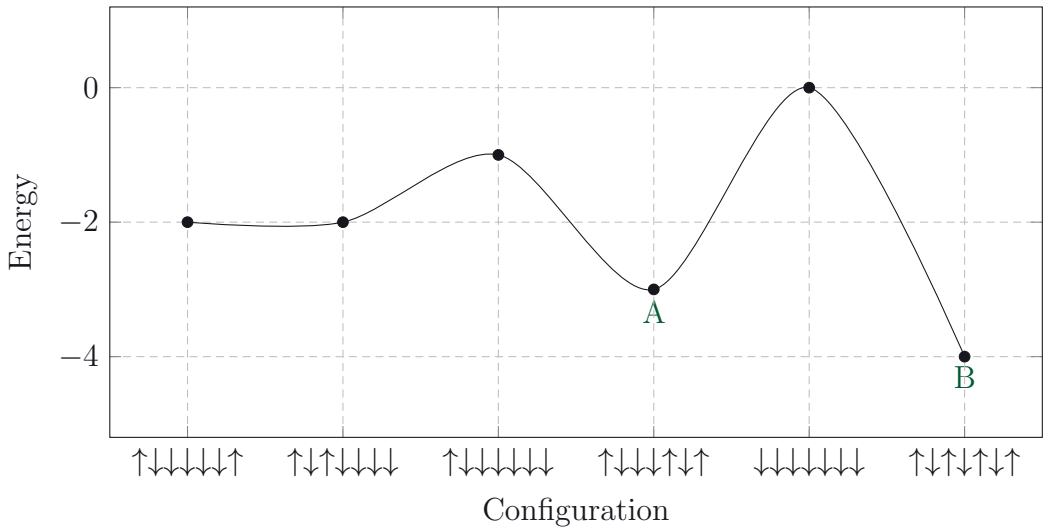


Figure 2.1: Illustration of a system of bar magnet configurations and the associated energy of the configurations. The local minimum energy A, is caused by external magnetic influences whilst the global minimum B, is a reachable state when the system is free of constraints. This can be modelled by a classical Ising model.

When using this model care needs to be taken that

- the Ising model formulation is not prohibitively large.
- the energy gap between the ground state and first excited state is not too small.
- the model does not require a highly connected quantum architecture.

### 2.7.2 Adiabatic Quantum Computing

There are different types of quantum computing, but this research focuses on gate-model quantum computing, adiabatic computing is useful for understanding the quantum algorithm to be used. In an adiabatic process, the time evolution of the system is such that there is no heat transfer concerning the system. In a quantum system with energy states described by an initial Hamiltonian $H_i$, in its lowest energy state, evolving into some final Hamiltonian $H_f$, the system will stay in its energy

configuration in $H_f$ subject to the speed of reconfiguration and the gaps between the energy spectra. This is the adiabatic theorem. This may be utilized by encoding the solution of a classical problem as a ground state of a quantum many-body system and annealing (repeatedly re-configuring) a simple Hamiltonian, such as an equal superposition, into the encoded system. At absolute zero the ground state may be read out directly, at other temperatures the corresponding Boltzmann distribution may be sampled from to determine the ground state as described in equation (2.2) and equation (2.3)

$$P(i) = \frac{e^{\frac{-H_i}{kT}}}{Z} \tag{2.2}$$

$$Z = \sum_{j=1}^{M} e^{\frac{-H_j}{kT}} \tag{2.3}$$

The time evolution of a quantum system in equation (2.4), is governed by the Schrödinger equation:

$$i\hbar \frac{d}{dt} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle \tag{2.4}$$

whose solution in equation (2.5), is the unitary operator:

$$\hat{U}(t) = e^{\frac{-it}{\hbar}\hat{H}} \tag{2.5}$$

that provides the gate-model state time evolution in equation (2.6) for $t > t_0$

$$|\Psi(t)\rangle = \hat{U}(t) |\Psi(t_0)\rangle . \tag{2.6}$$

In adiabatic quantum computing QUBOs are encoded into the Ising model form. There

are many examples of QUBOs being amenable to adiabatic quantum computation. QUBOs involve combinatorial optimisation and many NP-Hard problems can be formulated as QUBOs. There are many examples of QUBOs being solved on quantum annealers. Since available quantum annealers are not universal quantum computers but the gate model quantum computers to be made use of are, these efforts set a precedent for and give insight into what can be achieved. Using a quantum annealer has found application in small operational planning problems [42]. They explore the effect various mappings of the problems to a QUBO have. They find that the different mappings do affect the annealer performance. Classical approaches retain their performance advantage. The SAT problem can be encoded as a QUBO by directly mapping logical operations to Ising functions and reducing hardware embedding requirements for quantum annealers [43]. By encoding a routeing problem as a QUBO and embedding it on a quantum annealer an application in fault finding on an electrical network succeeded, but the classical counterpart ran out of memory [44]. This is a comparatively large problem for quantum computation. Design approaches to remove the requirement of additional penalty terms to constrain driver Hamiltonian's for constrained optimisation on quantum annealers are sought to reduce the required connectivity of the annealing architecture [45]. A general SAT problem can use `XOR` operators in place of `OR` operators in the SAT clauses, allowing the encoding of the problem onto a smaller sub-graph of the hardware topology [46].

This research on QUBOs provides demonstrations of specific encodings from the classical problem to a quantum version of the problem. Furthermore, it shows that consideration of the resource requirements of the encoding is required for practical speed benefits and executable problem size on the specific quantum architecture. It also provides an idea of the size of problems that are being solved in practice, although it should be noted that the available quantum annealers have a much larger qubit count than the universal gate model hardware. Importantly, there are signs of quantum advantages.

### 2.7.3 Gate Model Quantum computing

Unlike adiabatic quantum computing the gate model of quantum computing does not implement a specific Hamiltonian and evolve it. It instead uses gates to build quantum circuits. These gates are unitary operators that evolve the systems wave function as shown in equation (2.6). The gates are analogous to classical logic gates except that they are always reversible. Gate model quantum computing is universal, unlike current quantum annealer implementations.

### 2.7.4 Phase Estimation and Grover's Algorithm

To find the smallest eigenvalue of a system the Quantum Phase Estimation (QPE) algorithm can be used to find the eigenvalues of a unitary operator and therefore find the lowest eigenvalue [47]. It does this by estimating $\theta$ in equation (2.7). To get an accurate result using this method a deep quantum circuit is required and so this method is not suited for current QPUs.

$$U\left|\psi\right\rangle = e^{2\pi j\theta}\left|\psi\right\rangle \tag{2.7}$$

Grover's algorithm searches for a solution to an oracle in an unstructured solution space with $O\left(\sqrt{n}\right)$ and $\Omega(\sqrt{N})$ [48]. This suggests that Grover's algorithm is optimal and that NP-Complete problems are not in P for quantum algorithms [49, 50]. Its generalisation, amplitude amplification, can be used to solve SAT problems with multiple solutions [51, 52]. An $O\left(1.153^n\right)$ complexity algorithm using amplitude amplification in conjunction with a classical algorithm provides a quadratic speed up over the best known classical 3-SAT algorithm [53]. However, at this moment the circuits required are far too large to run on current NISQ devices even for very small problems. Although this might be a better solution in the future since it does not require NP-Complete reductions and this means that although the circuit might be deeper it doesn't duplicate SAT literals and make the circuit wider.

### 2.7.5 The Noisy Intermediate-Scale Quantum Era

The current state of quantum computing is such that the physical qubits used are extremely sensitive to noise and the devices have few physical qubits [20]. To implement error correction for a single ideally behaving qubit, many physical qubits are required. However, the difficulty of this is that it becomes increasingly difficult to put this many physical qubits into superposition or entangle them for a sustained period. However, there are enough qubits available such that classical simulations of some quantum-hardware-implementable quantum circuits are almost out of reach on current classical hardware. To this end, a class of quantum algorithms that are more robust to noise called variational algorithms are currently in extensive use and a large focus of current research.

Variational algorithms that involve optimization of parametrised quantum gates have shown immediate promise on NISQ devices. For a comprehensive review of NISQ algorithms, their applications, and associated concepts see Cerezo et al. and Bharti et al. [54, 55]. Examples include where an evolutionary algorithm was used to generate an ansatz for a Variational Quantum Eigensolver (VQE) with volume and noise resilience benefits, as well as a study where two versions of a novel adaptive optimizer required fewer measurements [56, 23]. See Moll et al. for a detailed and precise discussion of the quantum volume metric in the context of VQAs [57]. Grimsley et al. introduce an adaptive variational algorithm that generates an ansatz based on the specific molecule being simulated. The ansatz generated is both shallow and highly accurate as demonstrated with numerical simulations of the algorithm [58]. A further improvement over the aforementioned algorithm has been achieved, providing an even shallower circuit without any reduction in accuracy [59]. Furthermore, a variational circuit with shallow parametrised unitary gate circuits that operate on amplitude encoded training data and performs classification has been created [60]. The noise resilience and comparatively shallow parameterised unitary gate variational circuits help negate many of the problems encountered on NISQ devices as well as allowing for solutions to problems where the exact circuit implementation is not known.

One way researchers have sought to gain advantages in machine learning is to introduce quantum subroutines for expensive calculations in classical machine learning algorithms. There are many accessible reviews of quantum subroutines that speed up conventional machine learning algorithms. In general either expensive similarity measures are substituted by quantum algorithms or the probabilistic nature of quantum computation is leveraged for stochastic models. There are various advantages and disadvantages to the two aforementioned approaches [61, 62, 63, 64, 65]. Quantum subroutines for a similarity measure have been used in a K-Nearest Neighbours algorithm [66]. A modified Grover's search algorithm has been used to replace the softmax function intended for use with multi-class classifiers that may represent a quadratic speed-up [67]. Quantum computing has been used to train a deep restricted Boltzmann machine [68].

The approach of using quantum subroutines can provide a valuable starting point to enhancing machine learning since much is already known about the behaviour and implementations of the classical algorithms. However, these quantum-enhanced machine learning approaches are not immediately suitable for combinatorial optimisation and do not have the quantum noise-robust properties of the variational methods. Lastly, many machine learning methods lend themselves to parallel computation and are thus optimised fast enough even on classical hardware.

Examples of oracle based combinatorial optimisation for solving a maximal clique problem that obtains a quadratic speedup over classical methods exist [69]. The oracle excludes illegal cliques and classifies legal cliques. However, attempts to reduce other NP-Complete problems to this clique problem forfeits speed improvement. This direct approach required a specific quantum circuit implementation for the problem which may not be known. Furthermore, the resource requirement may be too large for NISQ devices.

Inspired by the not yet realisable Harrow-Hassidim-Lloyd (HHL) algorithm, which has an exponential improvement over the best classical algorithm for solving a system of linear equations, near term variational algorithms have also been used to solve the

same problem [70, 71, 72, 73, 74, 75]. Variational techniques are therefore immediately promising for a large variety of problems.

### 2.7.6 The Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) is a variational algorithm used for gate-model quantum computers that finds the smallest eigenvalue of a system described by a Hamiltonian. VQE drastically reduces the coherence requirements for finding an eigenvalue compared to phase estimation [76]. The Hamiltonian describes the energy of a specific configuration of the system. To find every configuration of the system's minimum eigenvalue is not feasible for even modestly sized systems. The Hamiltonian is therefore numerically approximated. This is done through the use of the Ritz principle [77]. Since the eigenvalues of a Hamiltonian are the energy spectrum of the quantum system, the VQE is used for finding ground-states in quantum chemistry, by using several mappings from fermionic operators to qubit operators, and is less frequently applied to combinatorial optimisation [78, 79, 80, 81].

An approximation of a wave function $|\Psi(\theta)\rangle$, called an ansatz, has its expectation value minimised by varying the ansatz parameters. The ansatz needs to be comprehensive enough to reach the minimum eigenvalue through its parameters ($\vec{\theta}$), yet small enough to be implementable on real hardware.

It can be shown using the Born rule and spectral decomposition of $\hat{H}$ in equation (2.8),

$$\left\langle \hat{H} \right\rangle = \left\langle \Psi(\theta) \middle| \hat{H} \middle| \Psi(\theta) \right\rangle = \langle \psi | \left( \sum_{i=1}^{N} \lambda_i \left| \psi_i \right\rangle \langle \psi_i | \right) | \psi \rangle \,, \tag{2.8}$$

that the ground state energy, $\lambda_0$ in equation (2.9), satisfies the inequality:

$$\lambda_0 \leq \sum_{i=1}^{N} \lambda_i | \left\langle \psi_i | \psi \right\rangle |^2 \tag{2.9}$$

That is, the ground-state energy is less than or equal to this value. The ansatz will always give an expectation value larger than or equal to the ground energy. This is called the variational principle.

### 2.7.7 The Quantum Approximate Optimisation Algorithm

Farhi et al. are the progenitors of the Quantum Approximate Optimisation Algorithm which approximates adiabatic quantum computing and can encode classical combinatorial problems using Ising Models [82]. The algorithm is most commonly used for combinatorial optimization. Since its inception, a classical algorithm more efficient for bounded degree constraint satisfaction problems and more accurate than random assignment has been produced [83]. However, the classical algorithm only has better accuracy guarantees than the Quantum Approximate Optimisation Algorithm for cases without ansatz repetitions [84]. A generalization of the aforementioned algorithm that does not require the problem to be encoded as an Ising Model has been presented [85]. The generalization uses the same abbreviation, "QAOA", which henceforth will refer only to the Quantum Alternating Operator Ansatz (QAOA). This allows for constraining the space of possible solutions to only feasible solutions. Design patterns and considerations for many constrained combinatorial optimization problems expanding on previous work can be found in [86]. Zhou et al. suggest that QAOA is not prone to the same vanishing spectral gaps as adiabatic quantum computing [87]. The QAOA Maximum Independent Set (MIS) solution is dependent on the choices of initial states and produces non-symmetric output distributions in contrast to the maximum cut solution [88]. Borle et al. use QAOA for binary linear least squares, and although they achieve promising simulation results, they were unable to replicate the results on a NISQ device [89]. Cook et al., provide an example of QAOA on a Vertex Cover (VC) problem [90].

### 2.7.8 The Quantum Alternating Operator Ansatz

QAOA provides a method for solving combinatorial optimisation problems such that the search space of the solution can be constrained to only feasible solutions. This is

a boon on top of the other benefits of variational algorithms. Furthermore, there are design methods for the circuits of many of the NP-Hard problems available (page 60 provides a method for transforming the SAT problem to a combinatorial optimisation problem with an available QAOA design pattern). This is, therefore, a well-suited variational method for application to engineering problems (page 44 explains the implementation of QAOA on the problem).

Attention needs to be given to the optimiser used for QAOA such that solutions approximately converge to optimums in the least shots. Stochastic gradient descent has been modified for use in quantum computation [91]. Non-derivative based optimisation may be less expensive for optimisation of quantum circuits [92]. By using a parameter shift rule the gradients of variational circuits can be found whilst the circuits can be treated as a black box system [93].

Since every iteration of the optimisation may require many re-runs of the quantum circuit a trade-off between the accuracy of the optimisation and the time taken to converge to a solution is developed. Therefore, many optimisations are to be compared.

QAOA is a variational algorithm for gate-model quantum computers that uses the intuition of quantum annealing [82]. The adiabatic path evolution is split into $p$ steps using the Trotterisation technique in equation (2.10). Whitfield et al. provide a brief discussion of Trotterisation which has been extended to show the conceptual analogy for comparing Trotterisation to quantum annealing [94, 95]. Cohen et al. discuss the attribution of the original Lie product formula that was later generalised by Trotter and eventually became known as the Suzuki-Trotter decomposition [96, 97].

$$U\left(t_i\right) = \prod_{i=0}^{p} e^{-iH_m\beta_i} e^{-iH_p\gamma_i} \tag{2.10}$$

The $p$ steps influence the approximation accuracy. Each discrete time step $i$has parameters $\beta_i$ and $\gamma_i$ that operate as one of the angles or time steps for the general

unitary transformations performed on the Bloch sphere and given by equation (2.11):

$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i\lambda+i\phi}\cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{2.11}$$

These parameters are classically optimized such that the energy of the system given by the objective function is optimal. The Hamiltonian's $H_m$ and $H_p$ are the mixer and phase Hamiltonian's, each encoding part of the combinatorial problem. QAOA includes more operators than only those corresponding to the time evolution of a fixed local Hamiltonian. It can, therefore, support ansatz where the evolution stays entirely within the feasible solutions using the mixer to constrain the problem rather than requiring additional terms to be added to an Ising model to penalise infeasible solutions [86, 98, 85].

## 2.8 Chapter Summary

Comparisons of algorithmic efficiency were discussed. Combinational logic circuit debugging is discussed as the specific problem, for which there is no known efficient algorithm, under consideration. Previous SAT based solutions to the debugging design are discussed and are used as the classical analogue to the quantum approaches. Combinatorial optimisation for graph colouring is introduced as one of the basic problem representations. The problem is an APX-Complete optimisation problem. The Ising model and Adiabatic quantum computing are introduced to explain the physical process behind gate model quantum computing. Some algorithms applicable to the debugging approach, QPE, Grover's were discussed and reasons for not using them were provided. The current state of NISQ computation is discussed and the VQE and QAOA algorithms are introduced as contending solvers. The intended scope of the dissertation is to detail the preparation of two major near term quantum algorithms to the debugging problem and to compare their efficiency to exact and approximate classical algorithms by gathering data of the runs.

Chapter 3 will focus on the reasoning behind the methods chosen to approach the debugging problem.

# Chapter 3

# Methodology

This chapter introduces the reasoning behind the choice to use quantum computing algorithms for the problem. The proposed methodology for the debugging problem is discussed, providing a high-level outline of the reasoning to the methods in the research.

## 3.1 Motivation

This work focuses on an engineering perspective to Quantum Computing. Whilst a lot of interesting theoretical work is being done around creating algorithms there are not a lot of engineering specific applications due to the field being mainly in the physicist's domain. Approaching Quantum Computing from an engineering perspective may supply the appropriate cognitive distance from the field that is sometimes necessary for new ideas to be applied.

This dissertation does not make use of existing data as the principal guide to the research, instead it is guided by broad ideas of how quantum computing *ought* to turn out. However, the results serve to furnish at least some empirical evidence that the quantum solutions will scale better than their classical counterparts, which puts it in contention for a suitable choice of algorithm for the debugging problem.

Many systems of even a small size have optimisation problems of engineering interest that cannot be solved exactly and at least one of the NISQ algorithms is exact in the limit with the other algorithms being able to approximate to less than or equal to the exact result. Furthermore, solving these problems exactly will be of great general benefit for two reasons. First, the NP-Complete class permits solutions to all the other problems in the class and second, the problems are applicable in many domains of study.

Originally only the QAOA algorithm was considered as a method to solve the problem. However, it was soon obvious that the circuits would be far too deep to run on NISQ devices. The rest of the research was therefore guided by the idea that using similar methods and ideas with VQE may place the solution within NISQ reach. Different versions of the variational algorithms are available to compare but this along with the hyperparameters and the long run times leaves too many options to explore. Therefore, only the options that seem most feasible are explored.

Both VQAs are expected to scale polynomially in some instances classical resources scale exponentially. The expectation is that this may happen in the qubits required to encode the problem, in the number of gates required in the ansatz and the classical optimisation of the ansatz. Note that if the gates required scales exponentially or if the classical optimisation scales exponentially any advantage may be lost and so careful consideration is required in these cases.

## 3.2 Proposed Methodology

As the debugging problem is NP-Complete, a quantum algorithm is an appropriate candidate to seek a scaling advantage. QAOA is a VQA that constrains solutions to feasible states, has gate resources that scale linearly, is exact in the limit, and has been applied to combinatorial optimisation problems. Therefore, it is the first quantum heuristic implemented. However, the VQE although not constraining solutions has very shallow circuit depths and is used as an additional algorithm to make comparisons with.

The Tseitin transforms map circuit problems into SAT problems that grow linearly with the input circuit size and can be performed in linear time. Since the specific SAT problems produced are NP-Complete it can be reduced into many forms amenable to quantum computation.

The circuits can be made UNSAT in one of two ways; first, a circuit with known IO pairs can be built and then have its truth table bits altered, second the circuit's gates can be substituted with incorrect gates whilst keeping the truth table specification constant. The second option seems to be closer to how such errors would normally arise in actual cases (errors in gate implementation with expected valid test vectors) and is therefore the preferred method of creating problem instances.

The MIS reduction from MAX-SAT is a simple reduction that can be performed in polynomial time. This further transformation allows quantum computation that solves the original MAX-SAT problem. The inverse transformations to ultimately identify candidate gate implementation errors are similarly resource-efficient.

The complement of the Partial Maximum Satisfiability (PMAX-SAT) solution which are Maximum Satisfiable Subset Clauses (MSCs) are MCSCs the smallest set of clauses for which the removal of any elements would make the circuit SAT. The MCSCs of which there may be more than one are enumerated as the error tuples at the Conjunctive Normal Form (CNF) level. The CNF level corresponds to fine-grained fault finding at the truth table level of the gate [8]. Only the graph PMAX-SAT is solved on QPUs. The transforms to and from the graph problem and solutions are performed as classical preprocessing and postprocessing steps.

## 3.3 Chapter Summary

The research is guided by broad ideas of how quantum computing *ought* to turn out. The decision to employ VQE in addition to QAOA is due to the physical limitations on current quantum hardware. It is hoped that an engineering approach to the domain of study can encourage new ideas and approaches in quantum computing. The methods

employed in this dissertation are guided by the scaling of the transformations into the quantum amenable problem structures as well as the ability of the solvers in use to solve the problem in the given encoding. This is the reduction of the debugging approach to PMAX-SAT, to a graph optimisation problem and then to a quantum variational approach.

Chapter 4 discusses the specific implementation of the methods to be used to achieve the solutions to the debugging problem.

# Chapter 4

# Methods

This chapter starts with showcasing the structure of the approach in the application of this research. Using this structure it separates the methods into classical pre-processing and post-processing sections sandwiching the intermediate quantum process. The classical preprocessing contains the transforms required to solve the debugging problem using VQAs. The quantum approach discusses the idea behind VQAs and then the mathematical structures, and their quantum circuit representations they operate on. Finally, the classical postprocessing shows how the resultant sampled output distribution of the quantum algorithms provide a set of suspect gate errors.

## 4.1 Structure of Approach

Figure 4.1 gives a high-level overview of how the various structures are related to solvers along with the outputs and inputs of each. Some blocks and some structure-specific properties are for the sake of simplicity omitted. The lines joining the blocks in the diagram are transformations if they are numbered, otherwise they are only direct inputs or outputs, the blocks themselves being the structures created. There are a few major components to the structure of the approach, the first being problem generation. The problem generation uses a dictionary to store gates and their labels

as a netlist that is used as a description of a circuit. The IO pairs are used as descriptions of how the circuit should behave ideally. These pairs form truth tables that fully describe some common combinational logic circuits. Within generation the circuits that are built are not the circuits described by the aforementioned truth tables, rather they are circuits with combinations of errors.

The next major section is the solver block. Within this block, everything is a classical solver in the sense that even the quantum solutions, which are hybrid algorithms, make use of a Central Processing Unit (CPU). The NumPy eigensolver and Minimalistic, Open-Source Satisfiability Solver (MiniSAT) specifically MiniSAT 22 (M22), are classical only solvers which are used as comparisons to the VQAs. The eigensolver uses the same input structure of the problem as the VQAs and gets results in the same format, which makes it helpful to make a fair comparison. M22 however does not require the input problem to be in the same format and skips some of the pre and post transforms required by the other solvers. Additionally, it computes an approximate solution to the MAX-SAT problem. This is used since it is a competitively used solver [99].

The last two sections concern the classical pre-processing and post-processing that transforms that original problem into a format that is amenable to the solvers. The pre-processing and post-processing are largely the transformations and their associated inverse transformations. The preprocessing transforms and writes the faux circuits to file using the PySAT library [99]. The files can then be loaded into a WCNF format and transformed into a graph ready for solving the MIS problem.

The postprocessing most commonly starts with a bit string result, or many, with associated sampled probabilities, that is then transformed into a MIS. From the MIS and using information from the circuit, although this happens in the module internals, the final MCSGs or in the case of the M22 solver MLCSGs, solutions are written to file.

Some of the missing pieces in the diagram include some performance metrics and structure snapshots written to file during and sometimes after computation, and some

helper functions to do with diagramming the results and properties of solvers and hardware. The performance metrics overlap significantly in the solvers that solve the instance of a graph problem making comparisons between them at the energy level, the CNF level and MCSG possible. In addition to some of the blocks not being represented some of the functionality is not used such as the unconstrained QAOA since it creates a larger ansatz than the constrained version and since the QAOA ansatz are too large in general they are not run on a QPU either. Noisy simulation results are also not recorded for reasons that will be discussed later on.

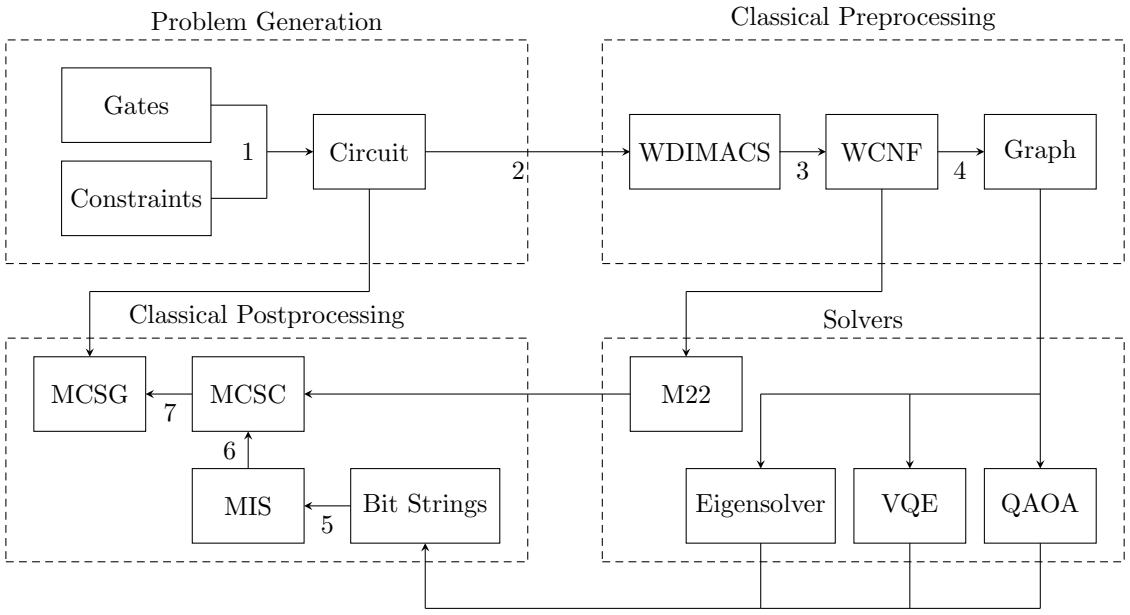


Figure 4.1: A high level overview of the structure of approach to solving the debugging problem. The arrows joining the blocks, or representation of the problem, represent outputs of the blocks unless they are numbered in which case they are transformations from one block into the next. The blocks are grouped into problem generation, solvers and classical pre and post processing.

### 4.1.1 Pipeline of Transformations

The numbering on the arrows in figure 4.1 corresponds to the high-level overview explanations of the transformations in the enumeration below. The transformations

discussed below omit discussion of the Hamiltonians and quantum circuits used for the VQAs, which are discussed in detail in section 4.3.

1. The IO constraints are rows in the truth table of the expected circuit specification. The gates are in CNF form and contain labels as keys in a dictionary. This dictionary structure along with IO pairs form the actual circuit implementation with its constraint properties.

2. Tseitin transforms of the circuit are saved into the WDIMACS file format. This contains the clauses in the circuit gates along with the weighting each clause is given, with the constraints forming extra clauses. A comment is inserted into the file with the original intention to tell the file reader which of the constraints are inputs and which are outputs, although it was subsequently not required.

3. The WDIMACS format is read from file and separated into a conjunction of the clauses.

4. The WCNF format is transformed into a graph format by relabelling the literals and joining both literals in a clause and contradicted literals with edges. It is on this structure that the MIS can be solved. The graph to Hamiltonian transformations and the Hamiltonian to quantum circuit transformations differ between all the solvers and remain to be discussed.

5. The resulting bit string results are ordered if there are more than one and the most likely solution is taken as the result. The result is then transformed back into the labels that correspond to graph vertices that together form the MIS.

6. The MCSC or MLCSC are either solved directly in the case of the MiniSAT or they are obtained from the graph complement of the MIS.

7. The MCSG are obtained from matching the clauses to the circuit netlist labels to identify the candidate gate errors.

## 4.2 Classical Preprocessing

This section serves as an example for understanding the methods used to set up the problem instances for quantum computation. A small circuit is transformed into the required graph problem as an initial example.

### 4.2.1 Tseitin Transforms

Since SAT solutions have been useful for this type of debugging problem, the circuits first need to be reduced to a logical form to solve their associated SAT problem. Suppose the following circuit in figure 4.2, with inputs and outputs all true, is to be debugged. Trivially, either the `NOT` gate or `AND` gate are erroneous.

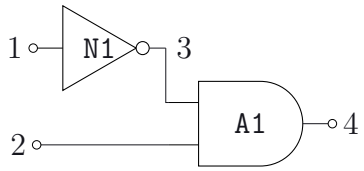


Figure 4.2: A small example of a combinational logic circuit that the solver approach can easily be illustrated on.

Table 4.1 shows the transformations of the gates in their conjunctive normal form. The inverse gates share the same Boolean formula but with the output literals negated.

Table 4.1: Tseitin transformations for all of the common logic gates.

| Gate | CNF |
|---|---|
| $\wedge$ | $(\neg i_0 \vee \neg i_1 \vee o) \wedge (i_0 \vee \neg o) \wedge (i_1 \vee \neg o)$ |
| $\vee$ | $(i_0 \vee i_1 \vee \neg o) \wedge (\neg i_0 \vee o) \wedge (\neg i_1 \vee o)$ |
| $\neg$ | $(\neg i_0 \vee \neg o) \wedge (i_0 \vee o)$ |
| $\oplus$ | $(\neg i_0 \vee \neg i_1 \vee \neg o) \wedge (i_0 \vee i_1 \vee \neg o) \wedge (i_0 \vee \neg i_1 \vee o) \wedge (\neg i_0 \vee i_1 \vee o)$ |

The SAT problem requires a Boolean representation that will evaluate to true if, and

only if, the representation conforms to possible IO assignments of the represented gate. For combinational logic gates, the Boolean representations need chained conjunctive clauses each containing a disjunction of literals, known as CNF. Furthermore, each clause may contain a maximum of three literals (3-CNF) but is referred to as CNF throughout the report for brevity, which is a canonical form for solving the SAT problem. This requires the Tseitin transform [100]. The proof below provides a derivation of the transform for the `AND` gate using propositional logic since the usually cited work does not seem to contain this information [101].

*Proof.*

1. $o \rightarrow i_o \wedge i_1$
2. $\neg o \rightarrow \neg(i_o \wedge i_1)$
3. $\neg o \rightarrow \neg i_o \vee \neg i_1$ — 2 DeM
4. $\neg\neg o \vee (\neg i_o \vee \neg i_1)$ — 3 Imp
5. $o \vee (\neg i_o \vee \neg i_1)$ — 4 DN
6. $o \vee \neg i_o \vee \neg i_1$ — 5 Assoc
7. $\neg o \vee (i_o \wedge i_1)$ — 1 Imp
8. $(\neg o \vee i_o) \wedge (\neg o \vee i_1)$ — 7 Dist
9. $(o \vee \neg i_o \vee \neg i_1) \wedge (\neg o \vee i_o) \wedge (\neg o \vee i_1)$ — 6, 8 Conj

□

Note that the transform introduces a constant factor of literals per gate and is, therefore, $O(n)$ in the number of gates. Note the two premises in 1 and 2 exhaustively describe states of an `AND` gate, the derivation for the other gates can be found in similar fashion, and the conclusion in premise 9 is the CNF representation.

The `NOT` gate in the circuit in figure 4.2 may be represented by equation (4.1) (the superscripts are explained in section 4.2.2):

$$(\neg i_0^0 \vee \neg o_0^5) \wedge (i_0^1 \vee o_0^6), \tag{4.1}$$

and the `AND` gate as equation (4.2):

$$(\neg o_0^2 \vee \neg i_1^7 \vee o_1^8) \wedge (o_0^3 \vee \neg o_1^9) \wedge (i_1^4 \vee \neg o_1^{10}). \tag{4.2}$$

### 4.2.2 Independent Set

MAX-SAT on 3-CNF is NP-Complete and therefore can be reduced to several problems such as IS, maximum clique, and VC [102]. It is these NP-Complete representations of the problem for which there are available variational formulations. Therefore a reduction from MAX-SAT to one of these forms is a requirement.

Although each of these reductions only costs P and their implementations do not confer any resource advantages over each other, they can require an extra step like inverting the IS graph for the maximum clique representation [85]. Next, the MAX-3SAT problem is reduced to the MIS problem. Given some graph $G = (V, E)$, an IS is a set of vertices $S$ such that no pair of vertices in the set share a common edge. The MIS problem is the problem of finding the largest such set. In figure 4.3 (note that in this example the literals in the clauses are grouped vertically from left to right for visual clarity) the Conjunctive Normal Form (CNF) formula is represented as a graph and one of the MIS is coloured in, the superscripts are shared with equations equation (4.1) and equation (4.2), and are the bit string indices in the problem encoding.

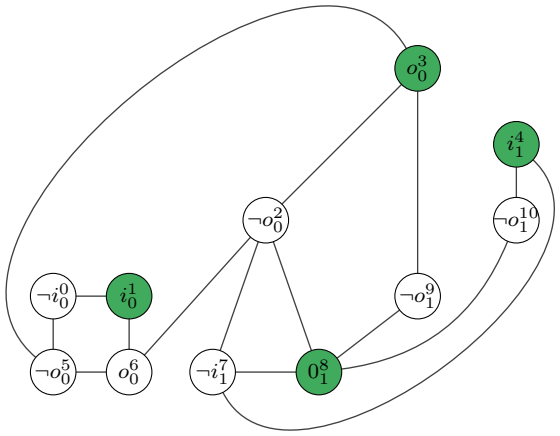


Figure 4.3: A graph with one coloured maximum independent set, corresponding to a solution of the circuit example in figure 4.2 with all its inputs and outputs set to true as the constraining test vector.

The graph may be constructed by creating adjacent vertices for each clause (that is with edges between them) such that at least one of the literals in the disjunction is forced to be true. However, since the truth assignments are to be consistent, a connecting edge between all literals and their negations is required.

There are no coloured vertices in the first `NOT` gate clause depicted by the figure 4.3 solution, identifying it as a suspect gate error. There may be multiple graph solutions that enumerate gate errors.

### 4.2.3 Graph Constraints

The original SAT problem contains constraints and the transformations need to preserve this for the solution to honour the IO pairs in the problem. The on and off constraints of the circuit are vertices that are coloured in the graph. Note that the truth content of a node may be false and so to colour, it indicates that the node is set to false in the SAT formula even though it is considered as true in the graph bit string solutions.

Setting these nodes as an initial step is a further classical preprocessing step that takes some computation off the solvers. If a node is true all its immediate neighbours may be set to false. Occasionally contradictions in truth value may be found in this first step where a node is to be coloured as a constraint and switched off, that is not coloured, at the same time. For the set of nodes where there are contrary results (as is to be expected in general for UNSAT cases) the node is not set as an initial state and instead resolved by the MIS solver in question. There may be a concern that this preprocessing step may add a large overhead to the algorithm overall but that is not the case since the values are known beforehand and are just set, therefore this setting is at most some fraction smaller than the edge size of the graph.

## 4.3 Quantum Optimisation

This section goes into detail on how the Hamiltonians are set up for the various VQAs. It starts with a general overview and example of classical-quantum hybrid algorithms.

### 4.3.1 Classical and Quantum Hybrid Approaches

As an excellent introduction to hybrid optimisation originally a Pennylane tutorial, one may consider figure 4.4 as a quantum circuit [103].

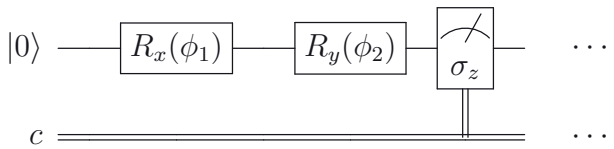


Figure 4.4: A small parametrised quantum circuit to illustrate hybrid quantum computation. The circuit can be classically optimised to find the lowest expectation value measured in the Pauli $\sigma_z$ basis.

Writing out the equation for the following circuit and its expectation in equation (4.3) and equation (4.4)

$$|\psi\rangle = R_y(\phi_2)R_x(\phi_1)\,|0\rangle \tag{4.3}$$

$$\begin{aligned}\langle\psi|\sigma_z|\psi\rangle &= \langle 0|\, R_x^\dagger(\phi_1)R_y^\dagger(\phi_2)\sigma_z R_y(\phi_2)R_x(\phi_1)\,|0\rangle \\ &= cos(\phi_1)cos(\phi_2),\end{aligned} \tag{4.4}$$

It can be deduced that the expected value can therefore vary between a maximum value of 1 and a minimum value of -1. Suppose the output desired was to rotate a qubit in its $|0\rangle$ state to its $|1\rangle$ state. Classically minimising the expected value by varying $\phi_1$ and $\phi_2$ will result in an expected value of -1. The eigenvalue of $|1\rangle$ in the $\sigma_z$ basis is -1 which would be the lowest eigenvalue the circuit could achieve if the cost function is exactly what the circuit outputs as seen in equation (4.5).

$$\sigma_z\,|1\rangle = -1\,|1\rangle \tag{4.5}$$

The design parameters for QAOA are the initial state, the phase operators, and the mixing operators. A derivation for the design parameters is given below [104, 105, 86, 98, 85, 88, 106].

Consider the following notation for Pauli operators defined by equation (4.6)

$$\delta_{mn} = \begin{cases} 0 & \text{if } m \neq n, \\ 1 & \text{if } m = n. \end{cases} \tag{4.6}$$

equation (4.7)

$$\sigma^m = \begin{pmatrix} \delta_{m3} & \delta_{m1} - j\,\delta_{m2} \\ \delta_{m1} + j\,\delta_{m2} & -\delta_{m3} \end{pmatrix}, \tag{4.7}$$

and equation (4.8)

$$m = \{1, 2, 3\} \mapsto \{x, y, z\}. \tag{4.8}$$

Given this notation any of the Pauli operators matrices can be calculated. It is important to note the convention that $j = \sqrt{-1}$.

To aid in the readers understanding consider the following example in equation (4.9)

$$\begin{aligned} \sigma^z &= \sigma^3 \\ \sigma^z &= \begin{pmatrix} \delta_{33} & \delta_{31} - j\,\delta_{32} \\ \delta_{31} + j\,\delta_{32} & -\delta_{33} \end{pmatrix} \\ \sigma^z &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \end{aligned} \tag{4.9}$$

The superscript represents which Pauli operator is under consideration so that the subscript remains available to index the operator on a specific vertex in the quantum circuit design.

Table 4.2 contains additional definitions of the notation used in sections 4.3.2 to 4.3.6.

Table 4.2: Definitions for the mathematical notation used to describe the graph structures optimised in the quantum circuits.

| Notation | Definition |
|---|---|
| $G$ | a graph |
| $V(G)$ | the vertices in $G$ |
| $E(G)$ | the edges in $G$ |
| $v$ | single vertex in $V(G)$ |
| $N(v)$ | $V(G)$ that neighbour any $v$ |
| $d(v)$ | the degree of $v$ |
| $\sigma_i^m$ | the $m^{th}$ Pauli operator at the $i^{th}$ $v$ |

The following sections 4.3.2 to 4.3.6 provide the Hamiltonians that describe the optimisation task. In each of the instances the Hamiltonians are applicable to a much wider range of problems than just the MIS problem, although that remains the focus.

### 4.3.2 Bit Driver Hamiltonian

Let $x_v$ denote an n-bit string $\{0,1\}^n$ corresponding to a graph colouring. The classical MIS objective function is the sum of the bits:

$$f(x_v) = \sum_{v \in V(G)}^{n} x_v. \tag{4.10}$$

The Boolean formula needs to be encoded as a Hamiltonian operator. Transformations are provided in [98, 107, 106, 108]. Using equation (4.10) the bit driver Hamiltonian equation (4.11) often called the phase Hamiltonian is

$$H_p = \frac{1}{2} \sum_{v \in V(G)} \left( \mathbb{I} - \sigma_v^Z \right). \tag{4.11}$$

which may be written as equation (4.12) (ignoring global phases)

$$H_p = \sum_{v \in V(G)} \sigma_v^Z. \tag{4.12}$$

A penalty or rewarding term may be introduced into the bit driver to bias certain terms. The bit driver is the cost Hamiltonian.

### 4.3.3 Penalty and Reward Hamiltonians

Penalisation of specific wires representing a node may be achieved by assigning it a lower value (the problems are maximised by finding the lowest negative value of the cost) and thus making it less favourable for the optimiser to seek to use the node

to decrease the Hamiltonian expectation value. The opposite sign is given to the Hamiltonian that rewards a node. Equation (4.13) represents the term that may be added to the bit driver Hamiltonian to achieve a penalising of certain terms.

$$H = -\frac{1}{2}\sigma_v^Z, \tag{4.13}$$

### 4.3.4 Edge Driver Hamiltonian

For graph colouring problems that colour with only two colours, the state of each vertex may be encoded in the computational basis. Binary coloured graphs are the concern of this research and optimisation problems over graphs in general need to have functions that reward certain pairwise assignments of colours between vertices. To this end each edge between a vertex can be described by the states in the following set: $\{|00\rangle, |10\rangle, |01\rangle, |11\rangle\}$. This signifies if adjacent vertices are not coloured, opposite colours or both coloured. It should be clear that for a MIS only the states $\{|10\rangle, |01\rangle\}$ are to be rewarded, whilst $|00\rangle$ is permissible and $|11\rangle$ is a violation. Depending on what is rewarded and penalised a VC or clique problem could also be solved using this strategy. Specifically, if a state is rewarded then it is assigned lower energy and if it is penalised it is assigned higher energy. This can be seen by creating the bit driver Hamiltonian and calculating the expectation $\langle\psi|H|\psi\rangle$ for each state. It should also be noted that $\langle 10|H|10\rangle$ and $\langle 01|H|01\rangle$ have the same energy, which is because the graphs considered are undirected and therefore it makes no difference in a pair of vertices which is coloured, to this end these states are always rewarded or penalised together. In a MIS problem $\{|00\rangle, |10\rangle, |01\rangle\}$ is rewarded and so a fully connected triangular graph with the colouring $\langle 001|H|001\rangle$ will have a lower energy than a graph with $\langle 011|H|011\rangle$, which in turn will be lower than $\langle 111|H|111\rangle$, since in the first example each pairwise colouring of $|00\rangle$, $|01\rangle$ and $|10\rangle$ is in the rewarded set, the second configuration has one violation and the final configuration violates all three edge constraints.

The bit driver, depending on the sign will maximise or minimise colouring in a graph

while the edge driver serves to bias the problem to adhere to the constraints. They are therefore typically summed together with the edge driver given a large weight coefficient to form a full Hamiltonian for a specific problem. Equation (4.14) may be used as the classical constraint function for the MIS problem:

$$H_e = \sum_{(i,j)\in E(G)} (x_i)\,(x_j), \tag{4.14}$$

Using the following transformation that takes the eigenvalues of the Paul matrix $\sigma^Z$ to 0 and 1 respectively, $x_i = \frac{1}{2}\left(1+\sigma_i^Z\right)$, the quantum edge driver can be written as equation (4.15):

$$H_e = \frac{1}{4}\sum_{(i,j)\in E(G)} \left(1+\sigma_i^Z\right)\left(1+\sigma_j^Z\right). \tag{4.15}$$

### 4.3.5 X Mixer Hamiltonian

The X mixer is defined in equation (4.16) and used in the unconstrained QAOA algorithm [82].

$$H_m \;=\; \sum_i \sigma_i^X \tag{4.16}$$

### 4.3.6 Bit Flip Mixer Hamiltonian

In order to implement the hard constraints of the MIS problem for a coloured vertex, equation (4.10) is constrained by equation (4.17). For each vertex $w$ included in the set of neighbours to vertex $v$, $w$ is required to be off (zero) such that the negation and then conjunction of all of the neighbours is true.

$$\bigcap_{w\in N(v)} \neg x_w = 1. \tag{4.17}$$

The bit-flip mixer Hamiltonian is described in equation (4.18), $b = 0$ is for MIS and $b = 1$ is for a VC:

$$H_m = \sum_{v \in V(G)} \frac{\sigma_v^X}{2^{d(v)}} \prod_{w \in N(v)} \left( \mathbb{I} + (-1)^b \sigma_w^Z \right). \tag{4.18}$$

The expansions of all the Hamiltonians assume that multiplication between the Pauli operator is tensor multiplication such that $\sigma_v^X \sigma_w^Z$ is short-hand for $\sigma_v^X \otimes \sigma_w^Z$.

### 4.3.7 Quantum Approximate Optimisation Algorithm

The phase is implemented partly with the edge driver and the bit driver with greater weight being put on the edge driver since this penalises or rewards graph colourings consistent with independent sets. The mixer is implemented as the $X$ mixer. The resulting Hamiltonians used are therefore equations (4.19) to (4.20)

$$H_p = \frac{3}{4} \sum_{(i,j) \in E(G)} \left(1 + \sigma_i^Z\right) \left(1 + \sigma_j^Z\right) + \sum_{v \in V(G)} \sigma_v^Z. \tag{4.19}$$

$$H_m = \sum_i \sigma_i^X. \tag{4.20}$$

### 4.3.8 Quantum Alternating Operator Ansatz

The QAOA uses the bit driver and bit flip mixer to produce the constrained Hamiltonians in equations (4.21) to (4.22). Since unfeasible solutions are not possible no weight coefficient is given to the bit flip mixer unlike in the unconstrained case.

$$H_p = \sum_{v \in V(G)} \sigma_v^Z. \tag{4.21}$$

$$H_m = \sum_{v \in V(G)} \frac{\sigma_v^X}{2^{d(v)}} \prod_{w \in N(v)} \left( \mathbb{I} + (-1)^b \, \sigma_w^Z \right). \tag{4.22}$$

Using Trotterisation from equation (2.10) the alternating ansatz can be prepared. The initial state $|s\rangle$ must be a valid IS since the solution is constrained only to feasible states during the optimisation. It is therefore recommended to use $|0\rangle^{\otimes n}$ since this is always a valid IS.

However, because this application contains hard constraints concerning the circuit IO values, the initial state is set to make the inputs and outputs true with a single $\sigma^X$ gate. Additionally, since these values are predetermined and by implication so are their negations, which in this case would be set to false, the initial state that is prepared in the figure 4.2 example is, therefore, $|01001000100\rangle$. Note, the bit indexing is from left to right following figure 4.3.

Equation equation (4.11) transforms the eigenvalues of the Pauli $\sigma^Z$ operator to 0 and 1 and is implemented by $R_z$ gates. Equation (4.18) may be implemented by a multiply open-controlled (switches on zero, not one) Toffoli gate with a bit flip as a target and a full uncomputation of the Toffoli gate after the target. Even though the graph is undirected every vertex needs to be treated as both a target and a neighbour and therefore the index iteration order in equation (4.18) is important. If such a gate is not available in the software used it is required to decompose it into available gates.

Figure 4.5 shows the two equivalent circuits side by side. If the ansatz is set up in this manner it will require the maximum degree of the graph $\Delta(G)$, minus one extra ancilla qubit. The open controls may be implemented by a $X$ (equivalent to $\sigma^X$) gate to flip the bit in front of the graph and an immediate additional $X$ gate to uncompute thereafter. The multiply controlled Toffoli is then implemented by chaining the outputs and inputs of ordinary Toffoli gates [50]. The mixer, therefore, corresponds to the neighbours of each vertex controlling whether it gets flipped, based on whether or not they have been flipped. Lastly, note that the two inner $X$ may be dropped as shown in the full ansatz in Figure 4.6.

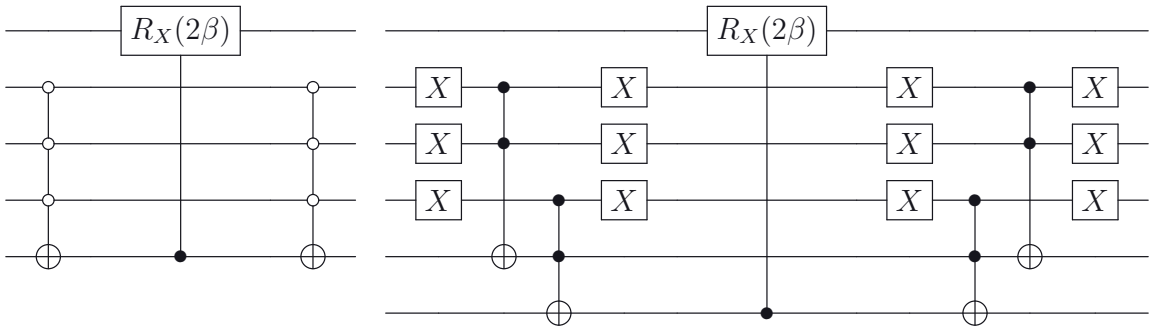


Figure 4.5: Two functionally equivalent quantum circuits. The right circuit decomposes the Toffoli gates of the left circuit. The decomposition is into gates commonly available in many quantum computing software packages. The quantum circuits are sub-mixer circuits that could appear in a QAOA ansatz.

In figure 4.6 the ansatz has barriers for the initial state, phase operator and each of the sequential mixing operators. The sequential mixing operators may contain adjacent $X$ gates that can be optimized out depending on the ordering. The order of the mixing operators also affects results in the case that there is more than one MIS since the first sequential mixer will flip a specific qubit that will therefore constrain the rest of the possible flips. This effect appears to diminish as $p$ increases.

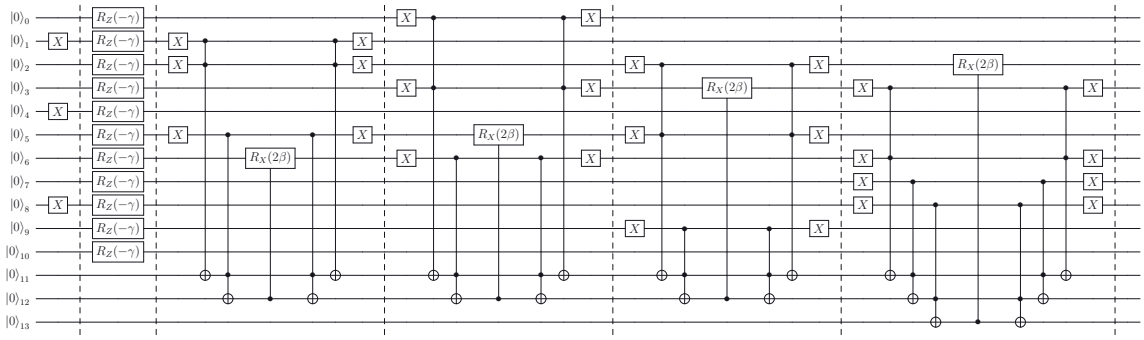

Figure 4.6: A partially decomposed QAOA ansatz without any repetitions ($p = 1$). The initial state, phase and sub-mixer components are separated by barriers. Some mild optimisation that leaves out sub-mixers concerning already determined graph colours are left out. This ansatz is used to solve the debugging problem in figure 4.2 with the test vectors all set to true.

### 4.3.9 Variational Quantum Eigensolver

The phase is implemented partly with the edge driver and the bit driver with greater weight being put on the edge driver since this penalises or rewards graph colourings consistent with ISs. There is no mixer required in this instance, and the Hamiltonian in equation (4.23) may be implemented for a classical solution such as annealing also.

$$H = \frac{3}{4} \sum_{(i,j) \in E(G)} \left(1 + \sigma_i^Z\right) \left(1 + \sigma_j^Z\right) + \sum_{v \in V(G)} \sigma_v^Z \tag{4.23}$$

Using a layer of rotations followed by a layer using a linear entanglement strategy, a simple but standard ansatz, over which equation (4.23) can be utilised, is built corresponding to figure 4.3 for VQE [109].

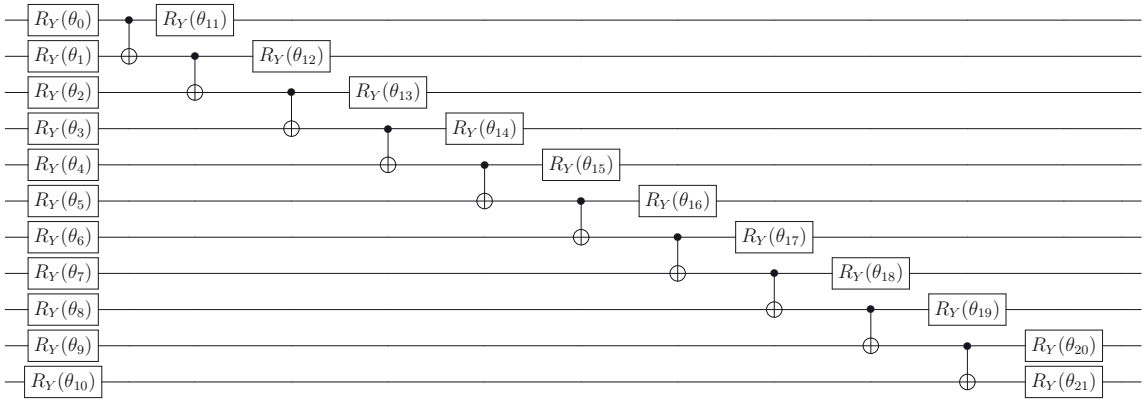


Figure 4.7: A simple ansatz used for VQE problems with one layer of rotation gates and one layer of linear entanglement.

## 4.4 Classical Postprocessing

This section serves as an example to understanding the methods used to go from sampling bit strings from quantum circuit measurements to a solution in the original problem domain and continues the use of the small circuit example for illustration.

### 4.4.1 Bit Strings to Maximum Independent Set

Suppose the following classical bit string measurements $x_i$ are sampled as the output of figure 4.6 01011000100 and 01001100100, this can be transformed into the MIS by simply finding the index of every high bit in the bit strings as in equation (4.24), where $\mathcal{I} = \{1, 2, \ldots, n\}$.

$$\mathcal{M} = \{i : i \in \mathcal{I}, x_i = 1\} \tag{4.24}$$

The two MIS would therefore be $\mathcal{M} = \{1, 3, 4, 8\}$ and $\mathcal{M} = \{1, 4, 5, 8\}$ respectively.

### 4.4.2 Maximum Independent Subset to Minimal Correction Subset Clauses

Using equations (4.1) to (4.2) superscripts and the MIS results from section 4.4.1, which needs to be labelled to keep track of the variables for the CNF to MIS transformation, the MCSC can be retrieved. The transformation is one-to-many and so the which-order information needs to be saved to invert the transform.

Doing so results in $\{(i_0^1 \vee o_0^6), (o_0^3 \vee \neg o_1^9), (i_1^4 \vee \neg o_1^{10}), (\neg o_0^2 \vee \neg i_1^7 \vee o_1^8)\}$ and $\{(i_0^1 \vee o_0^6), (i_1^4 \vee \neg o_1^{10}), (\neg i_0^0 \vee \neg o_0^5), (\neg o_0^2 \vee \neg i_1^7 \vee o_1^8)\}$

### 4.4.3 Minimal Correction Subset Clauses to Minimal Correction Subset Gates

By keeping track of which gates were transformed into which CNF form the MCSGs can be found by seeing which gates contain the UNSAT clauses. In the current example the MCSGs are $\{\texttt{NOT}, \texttt{AND}, \texttt{AND}, \texttt{AND}\}$ and $\{\texttt{NOT}, \texttt{AND}, \texttt{NOT}, \texttt{AND}\}$ respectively.

## 4.5 Chapter Summary

The general structure of the approach is broken down into the following components: The problem generation, the classical preprocessing using Tseitin transforms and reduction of SAT to a graph problem, the various solvers to be compared that run classically or on hybrid hardware architecture and lastly the classical postprocessing which transforms the quantum output distribution into a graph solution and graph solution into suspect gate errors. A small example of an UNSAT circuit is used to showcase the various transformations the problem goes through as well as the outputs of each step from start to finish.

Chapter 5 explains the application of the theoretical methods presented in this chapter. Along with additional considerations and problems to be overcome for the specific implementation.

# Chapter 5

# Application

This chapter describes the problem generation approach. As well as how the quantum circuits are built to suit the problem on a Quantum Processing Unit (QPU). Further discussions concerning the ansatz optimisation and simulations of noise are provided. Finally, the optimisation process of the VQAs is compared.

## 5.1 Problem Generation

Although figure 4.3 has the vertices carefully labelled to correspond to the CNF expressions and the example ansatz in figure 4.6 this chapter switches notation to be consistent with the developed Python modules.

The following sections 5.1.1 to 5.1.5 describe the circuit problem, specification set and the format used to describe the various structures in the transformations to the final constrained graph problems.

### 5.1.1 Circuits

The classical combinational logic circuits that are used as instances for the problem are showcased in figures 5.1 to 5.7. These instances are chosen based on three criteria,

1. that the circuits increment by one gate in each instance,
2. that the circuits are all small instances of the problem that will be possible to run on NISQ hardware and
3. that the circuits are instances of functional circuits such that the gate errors would be visually obvious if the functions are known.

Since the comparison of the solution is made to M22 the problem instances cannot start as graph instances. It also should not start as graph instances because that would ignore the use case of the solvers that are to operate for debugging circuits rather than solving a generic graph problem. Additionally, although all the circuits can be represented as a MIS problem and although every MIS problem can be represented in SAT and therefore also CNF, it should be obvious that there are cases of CNF that cannot be circuits. It is also well known that algorithms have different performances on different types of graphs, this is the case for quantum algorithms in [110]. No analysis has been performed to randomly generate the subset of graphs that are also circuits since this would violate the condition that the circuits be functionally useful and its behaviour known a priori. This would also create an additional problem in finding circuit graphs that are small and scale by one.

There is a drawback to this approach in that whilst the circuits may scale gate by gate the graphs will not. This is because each gate is always represented by multiple vertices which are often token duplicates of the same literal type in different clauses. This creates a sparse increase in the number of qubits required to run each successive circuit problem and means that the size of circuits that can be simulated quickly reaches a cap because it is not classically feasible to simulate graph problems that have much more than thirty vertices. However, the quantum algorithms are still usable in this case where the classical approach starts faltering from the spatial scaling in the MIS instances.

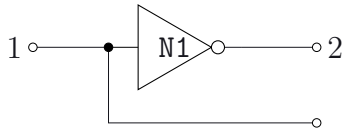


Figure 5.1: A one-to-two line binary decoder circuit for use as a data point in the generated circuit problem set.

It should be noted that the label of the inputs and outputs of every gate is significant and is used as the label for the CNF. It should also be noted that whilst in chapter 4 the literals are signified by letters which are common to symbolise a placeholder, from now the literal symbols are numbers and so "7" does not represent a number but a truthy label of the $7^{th}$ input or output whilst "-7" is falsey.

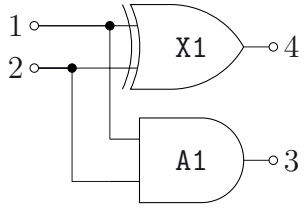


Figure 5.2: A half adder circuit for use as a data point in the generated circuit problem set.

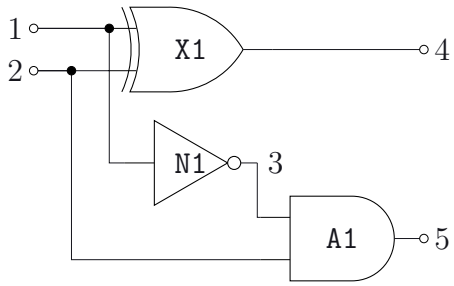


Figure 5.3: A half subtractor circuit for use as a data point in the generated circuit problem set.

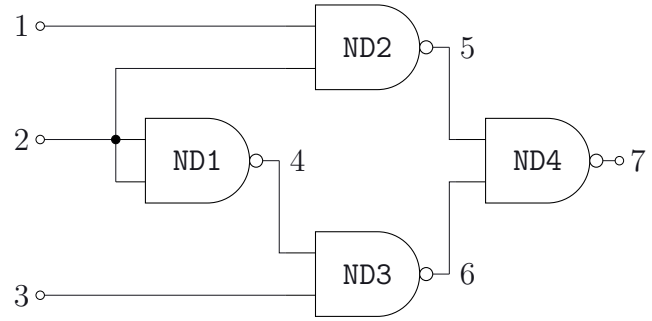


Figure 5.4: A two-to-one multiplexer circuit for use as a data point in the generated circuit problem set.

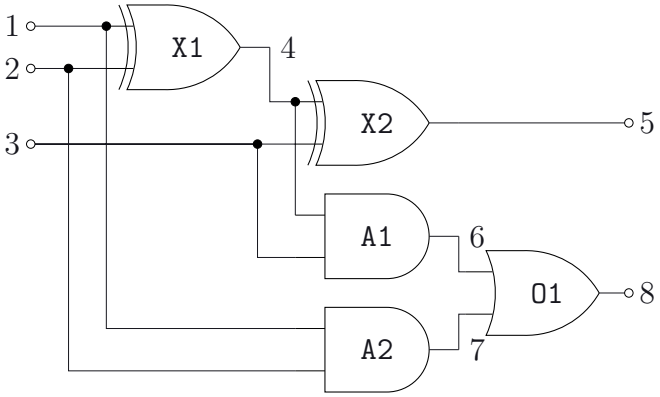


Figure 5.5: A full adder circuit for use as a data point in the generated circuit problem set.

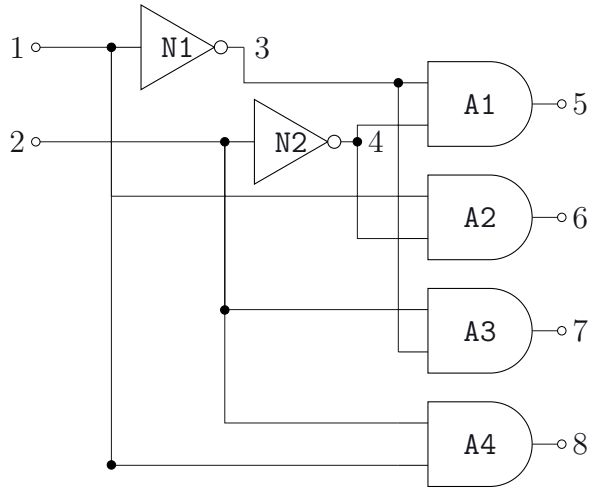


Figure 5.6: A two-to-four line binary decoder circuit for use as a data point in the generated circuit problem set.

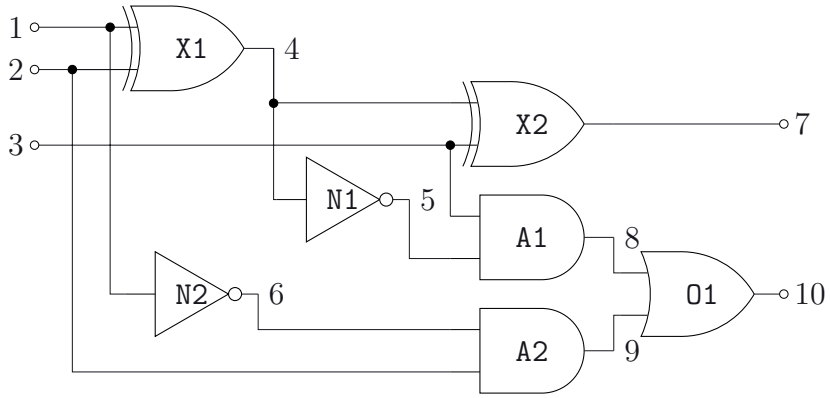


Figure 5.7: A full subtractor circuit for use as a data point in the generated circuit problem set.

### 5.1.2 Specifications

The specifications for the behaviour of the circuits are based on their functionality. Simulation software may provide this information for larger cases but the focus of the dissertation is not on the problem generation even though it is required for

benchmarking the solvers since the ISCAS-85 problems are too large for NISQ devices. Table 5.1 is the truth table for figure 5.7 and serves as the specification of the circuit.

Table 5.1: Truth table for a full subtractor combination logic circuit. Using numeric notation for symbols.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| -1 | -2 | -3 | -7 | -10 |
| -1 | -2 | 3 | 7 | 10 |
| -1 | 2 | -3 | 7 | 10 |
| -1 | 2 | 3 | -7 | 10 |
| 1 | -2 | -3 | 7 | -10 |
| 1 | -2 | 3 | -7 | -10 |
| 1 | 2 | -3 | -7 | -10 |
| 1 | 2 | 3 | 7 | 10 |

The approach taken is brute generation of test vectors which correspond to every row of the truth table. This constitutes the expected behaviour and ground truth for the circuit behaviour in each case.

### 5.1.3 Gate Errors

The circuit structure that is built then constitutes the incorrect implementation, figure 5.8 shows some gates that are swapped with their inverses (compare to figure 5.7).

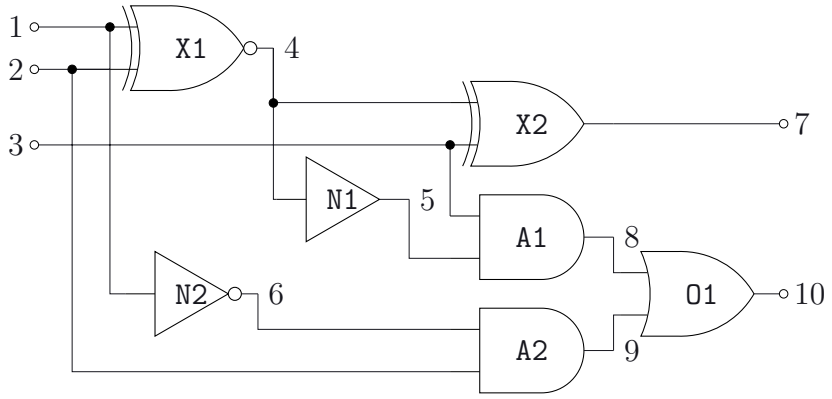


Figure 5.8: A full subtractor circuit with an erroneous gate implementation set. The erroneous gate implementation is to be located by the debugging solvers.

Then for every combination of the inverse gate, in place of the correct gate, an erroneous gate is swapped into the circuit. Every one of these erroneous circuits is matched with all of the test vectors. It should be obvious that these two brute approaches of problem generation used together are not a scalable way to create anything but the smallest problems. This generates $2^{i+g}$ problem instances where $i$ is the number of inputs to the circuit and $g$ is the number of gates in the circuit. In this case that is 1024 problem instances or circuits saved to file in `json` format as in listing 1. Note that listing 1 does not show every gate due to the many lines of `json` and instead the removed gates are replaced by the `json` key "Missing Gates". Additionally, the same amount of problem specifications are saved to file in the WDIMACS format, although this is an artefact of how things are implemented since only $2^i$ specifications are strictly required.

Listing 1: SAT circuit implementation as written to file with a visually compressed `json` format. This circuit stucture corresponds to the full subtractor circuit in figure 5.7.

```
{
        "X1": [
                [-1, -2, -4],
                [1, 2, -4],
                [1, -2, 4],
                [-1, 2, 4]
        ],
        "Missing Gates": [],
        "O1": [
                [8, 9, -10],
                [-8, 10],
                [-9, 10]
        ]
}
```

There are a few reasons this should not be a concern. Primarily because creating the problems is not the focus of this research and test vector generation is a separate problem. Additionally in practice, not every possible test vector will be used and not every possible combination of error should occur. The errors should be rare cases and are unlikely to be more than a tiny fraction of the gates in the circuit. Generation of this many problem instances is just helpful for gathering data exhaustively. Useful or likely results can be filtered later on and better problem generation strategies can be sought after for larger problems.

It may be the case for some type of swaps of erroneous gates that most of the truth table remains SAT but it will always have at least one error in the entire table except in the rare case that the gate that is erroneous creates a logically equivalent circuit. An assumption made is that this is more likely the more the truth table of the gate

is similar to its replacement. This is because the equivalent truth table scenarios are eliminated by using the gate inverse rather than just a different gate. However, this will also depend on how this gate relates to subsequent steps and so this assumption should be formally verified. Additionally exchanging the gates for a gate with a different number of terminals is also possible. However, this would in reality be a less likely error as it will require both an error in the gate implementation and simultaneously the wiring of the circuitry. For this reason, this is not explored.

It is worth noting that the more problems there are in the circuit the less helpful the debug information will be. In other words, the solution might consider most of the circuit to be problematic and produce too many actionable combinations of solutions that could potentially provide a SAT circuit. However, this is just what one would expect if most of the gates were erroneous, in practice it is expected that relative to the circuit size there would be very few errors thus the solutions provided would pick out more specific problems and therefore be more helpful.

### 5.1.4 WDIMACS Format

The WDIMACS format is not required as part of the transforms to get a working solution. However, it is useful for maintaining a problem set on disk that does need to be generated on the fly. Additionally, it is useful as a standard since every SAT solver can take this input format. It is useful to describe the standard by successively building on SAT problems.

A standard MAX-SAT problem can be described by adding comments on a line beginning with the character “c” (for comment). A line starting with the character “p” (for parameters) states the format of the problem as a “cnf” problem with $n$ literal types and $m$ number of clauses. Each line thereafter should be a clause, using the numbering as variable labelling mentioned previously, ending with 0. This precludes using 0 as a label.

For a WMAX-SAT problem the “cnf” parameters is replaced with “wcnf”. The start of the clause lines can contain a whole number as a weight.

In a PMAX-SAT problem an extra parameter $t$ is appended to indicate the weights of the hard clauses. If a clause line begins with this number the clause must be satisfied. Soft clauses should be lower than this value. Depending on the SAT solver there may be extra stipulations on the number. For the M22 solver used in this research, the top weight is required to be larger than the sum of the soft clause weights. Listing 2 corresponds to the figure 5.7 in the SAT case and the first line of the truth table in table 5.1. The line comment is used so that a circuit structure can be created using the input WDIMACS file (although it remained unused), which requires that the columns split on inputs and outputs for the circuit specification be given, in this case, the inputs stop after column 3, refer to table 5.1.

Listing 2: Illustration of PMAX-SAT in the WDIMACS format as a circuit specification for the full subtractor circuit in figure 5.7. This is for the SAT case and corresponds to the first line of the truth table in table 5.1.

```
c 3
p wcnf 10 26 22
1 -1 -2 -4 0
1 1 2 -4 0
1 1 -2 4 0
1 -1 2 4 0
1 -3 -4 -7 0
1 3 4 -7 0
1 3 -4 7 0
1 -3 4 7 0
1 -4 -5 0
1 4 5 0
1 -1 -6 0
1 1 6 0
1 -3 -5 8 0
1 3 -8 0
1 5 -8 0
1 -2 -6 9 0
1 2 -9 0
1 6 -9 0
1 8 9 -10 0
```

```
1 -8 10 0
1 -9 10 0
22 -1 0
22 -2 0
22 -3 0
22 -7 0
22 -10 0
```

### 5.1.5 Graphs

A graph is built from the conjunctions of the CNF clauses. There are two components of the graph to be considered for this application. The first is the creation of the graph itself. The second is the setting of some of the nodes based on the constraints. It must be emphasized that since nodes can represent truthy or falsey literals that switching them to on does not mean that the literal type is true or on.

The literal tokens in CNF are encoded by simply assigning them a whole value sequentially as a key and keeping track of the associated literals as values, these numerical values must not be confused with the numerical values (which are symbols) in the CNF formula. This is required to match the literals that are constraints with constraints in the graph. Additional nodes or vertices are uncoloured based on whether their neighbours are previously coloured. This assures that some of the required literals in the MIS are preset even before optimisation. If during this step a value that would be uncoloured is coloured by the previous step the node is not considered a constraint and so is uncoloured and left for the optimiser to decide. This occurs because during the setting of immediate neighbours to off some UNSAT behaviour is already present; however, it is not at this stage known whether it forms part of a MIS rather than just an IS and it is not therefore immediately added to the final result.

Edges are assigned to every unordered pairwise combination of literals, $(l_1, l_2)$ within

every CNF clause, $\mathcal{C}$ and itself as in equation (5.1). This creates fully connected sub-graphs for each clause.

$$\{(l_1, l_2) : l_1 \in \mathcal{C} \wedge l_2 \in \mathcal{C}, l_1 \neq l_2\}. \tag{5.1}$$

Additionally, edges are assigned between all literals and their negations in the entire CNF represented as the Tseitin transform, $\mathcal{T}$ as shown in equation (5.2).

$$\{(l, \neg l) : l \in \mathcal{T}\}.. \tag{5.2}$$

Together equations (5.1) to (5.2) result in a set of two-tuples that represent the pairwise relationship between vertices in the graph equation (5.3).

$$\{(0,1),(0,2),(0,3),\ldots,(48,52),(49,50),(51,52)\} \tag{5.3}$$

In this case figure 5.9 (the edges curved to make the neighbouring connections clearer), represents the figure 5.7 circuit.

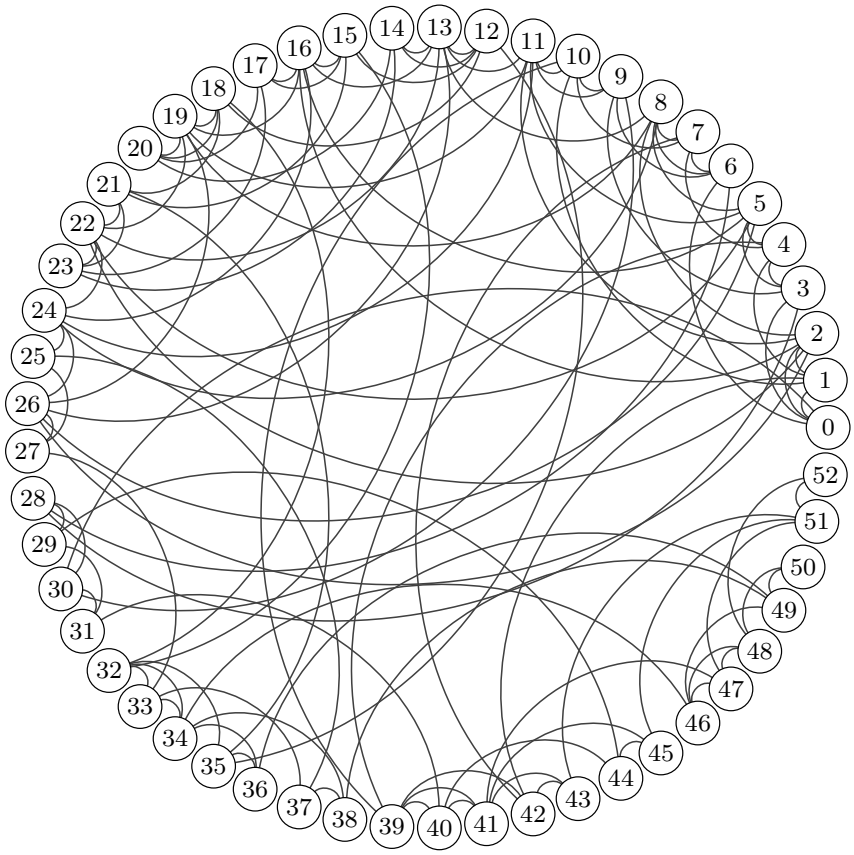


Figure 5.9: A graph structure on which solving the maximum independent set problem encodes the solution to the debugging problem on the full subtractor circuit in figure 5.7

## 5.2 Adaptive Hamiltonians and Ansatz

The graph created contains an additional tuple of which vertices to constrain. These terms are on or off terms. It may be recalled that every node that is to be assigned a colour due to a literal constraint in the CNF is on. However, the returned uncoloured nodes from the preset colouring in the graph domain are for purposes of penalising or rewarding certain Pauli terms in the Hamiltonians and can be uncoloured or not. If the node is to be on then a reward is added, if off then a penalty is added cf.,

section 4.3.3. These terms are added to the Hamiltonian that acts as a cost function for the VQA.

There are commonly multiple MISs for any given graph. A common approach to enumerating additional solutions is to ban the previously occurring solutions by conjunction of the negation of the previous solutions. This ensures that the negation of the solution needs to be satisfied also. The solution as a whole is banned, and not parts of it, because the MISs may overlap. This approach is not feasible in this case as it increases the size of the graph for every run and adds bias terms in the Hamiltonian which may not be already existing terms.

Instead, terms in the Hamiltonian are rewarded if they are not in previous solutions and thus will bias them in the inclusion of the next iteration only if they are not also a constraint. If these terms are not part of any MIS they are not forced into the solution since the node is only possibly part of the set and if they are form part of a non-MIS solution the enumeration can stop.

The Hamiltonian is adapted to bias other solutions in the solution enumeration by repeatedly creating a new set of vertices $\mathcal{G}$ by removing the already appearing coloured set of vertices $V(C)$ from previous solutions, including those coloured due to the initial constraints. This bans repeatedly outputting the same solution. Equation (5.4) represents this process for an $i^{th}$ iteration. $\mathcal{G}$ is randomly sampled to reward one of the nodes until there are no more nodes left to reward and the enumeration is considered complete. If the biasing Pauli terms do not already exist in the Hamiltonian then the Hamiltonian will contain more terms as the enumeration continues, and the classical optimisation load is increased, but by negligent amounts. The ansatz itself is not affected by this and neither is the graph.

$$\mathcal{G}_i = V(G) \setminus (V_{i-1}(C) \wedge \mathcal{G}_{i-1}) \tag{5.4}$$

Similarly, the ansatz is adapted by prepending a Pauli $X$ gate to vertices sampled from $\mathcal{G}$. It should be noted that simply setting the corresponding bit in the solution

and excluding the corresponding wire would lessen the width of the ansatz and guarantee the bit is set according to the bias. However, it is not certain that the bias bit, unlike the constraints, should be set. Rather the bias adds selective pressure to certain nodes.

## 5.3 Quantum Circuit Transpilation

Whilst limiting the number of gates used in the classical simulations of the quantum circuits matters to limit the number of matrix operations there is otherwise no considerations of which gates to use and where to place them. For running the quantum circuits on a QPU there is a significant impact of the gate types and layout of the circuit.

Two types of transpilation are strictly required for a QPU. The first is mapping the virtual qubits in the circuit to the physical device topology. The second process is like the first but requires swapping wire terminals to support entangling gates on the hardware architecture where only some physically entangled qubits are available. This needs to be done without creating knock-on effects on the rest of the circuit placement. In both these cases, the transformation that takes place needs to be reversed such that correctly ordered bit strings are measured and such that the routing does not affect the circuit functionality.

Transpilation often involves processes that are themselves NP-Hard. Furthermore, it is not obvious whether the target for optimisation ought to be depth at the expense of some other important factor and to what extent. For example including more entangling gates in the circuit that have higher error rates than other gates or including less of them but at the same time slice in the circuit, causing cross talk noise. The target for optimisation could change depending on the problem size since for smaller problems more general ansatz will give better results without being too large. In fact, for the smallest circuit problem figure 5.1 a more general ansatz approach is used since the custom approach does not permit parametrised gates at that size which is a requirement for VQA. Additionally, transpilation may require very well thought

out placement of SWAP gates. This is yet another complication since these gates are not basis gates and are implemented with three CNOT gates. The initial mapping of virtual onto physical qubits will affect where SWAP gates need to be used. Because of the difficulty of optimal transpilation, it is not a major focus of this work but much work on it has been conducted in [111].

The circuits run on the QPU are transpiled to the specific basis gates and aggressive automatic optimisation is set. However, the runtime environment used for the QPU runs does not support automatic transpilation options for the backend. Since the aim of this research is not optimal transpilation the manual process of transpilation required is not pursued as even if the manual process was in place it would additionally require the Hamiltonians to be remapped to match the resultant circuit.

Appendix C contains the topology of the Manhattan backend in figure C.1.1 whilst table C.1 shows properties of the device that affect the accuracy of results. Notably, there are some highlighted CNOT errors and T1 and T2 times that are likely to affect the results. Additionally, it can be seen from a comparison between figure C.1.1 and figure 5.9 that many SWAP gates will be required to embed the quantum circuit onto the hardware.

### 5.3.1 Implemented Reductions

To get accurate results with a VQA, the space of possible solutions must be reachable with the ansatz. This can require entangling every qubit with every other qubit but this is not practical for a NISQ application. For a graph representation of a circuit, only the neighbouring vertices need to be entangled, but even this creates too many entanglements for a NISQ device. Using entangling gates in a linear ladder formation can produce accurate results and does not require entangling vertices that have more than one neighbour with all its neighbours as is the case in figure 4.7. Because the QAOA is not run on a QPU in this work it does not try to reduce the usage of these gates.

The ansatz is customised specifically to be as shallow as possible to run on the

hardware whilst still being as appropriately specified to the problem as possible to allow reaching all the possible states. These solutions may be compared to the Hamiltonian driven building of the ansatz, which results in a significantly deeper ansatz. These reductions are done both based on understanding what the ansatz function is meant to do, especially in the QAOA case, as well as experimenting with variations of available ansatz in the VQE case.

The subsequent ansatz in figures 5.11 to 5.12 are based on figure 5.10 which in turn corresponds with a medium sized circuit (the half subtractor, figure 5.3) in the set of generated problems.

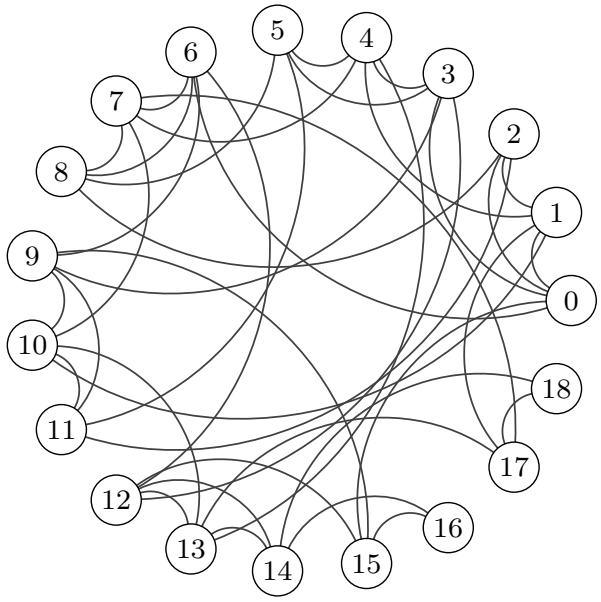


Figure 5.10: A graph structure on which solving the maximum independent set problem encodes the solution to the debugging problem on the half subtractor circuit in figure 5.3

To follow the figure 5.10 correspondence to the figures 5.11 to 5.12 take note that the endianness of the quantum circuit is opposite to the graph. This is because the Qiskit VQA classes reverse the Hamiltonian cost functions but not the ansatz therefore the ansatz needs to be manually reversed to be consistent with the Hamiltonian. This

also means the biasing of the ansatz circuit in the enumeration needs to be reversed, although not for the cost function.

The VQE reduced ansatz operates on unknown truth assignments of vertices and its neighbour with a `CNOT` ladder that entangles every node to its neighbouring wire with the final wire considered a neighbour to the first as shown in figure 5.11. This is also called circular entanglement for the aforementioned reason and this ensures that every wire is a control and a target rather than in the case of linear entanglement where the first and last wire are only targeted or only controlled. Where truth assignments are known entanglement is not required, and if the vertex truth value is unknown composed rotation gates are used instead. This ensures small localised circular entanglements and significantly decreases the circuit depth without impacting the result. For wide circuits, this last entanglement can require a large number of swaps if the target physical qubits are many wires apart from each other. However, because these circular entanglements are restricted to connected vertices they are close to each other. The exception to the previous statement arises where there are wire connections between classical gates with a large number of gates in between such as might occur in sequential combinational circuits.

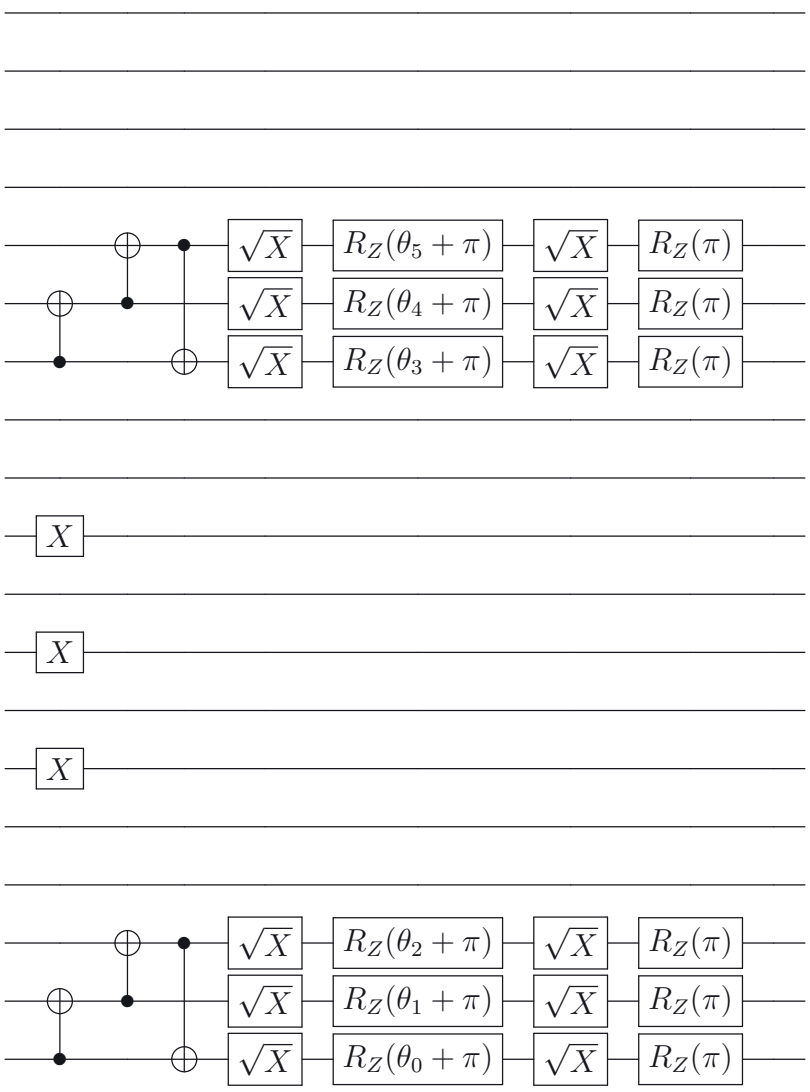


Figure 5.11: A transpiled custom ansatz corresponding to the half subtractor graph in figure 5.10 with localised circular entangling layers to be used with the VQE solver.

Figure 5.12 showcases the reductions performed on the QAOA ansatz. These are simply to remove sub-mixers that operate on already known vertices. This diagram is not decomposed into basis gates such that its relationship to figure 5.10 is clear. Additional improvements such as moving commuting mixers into the same time slice (see the first and fifth mixer in figure 5.12) are not implemented since it is too large to run on a NISQ device even with further optimisations.

In future work, the trivially true and false inputs may be completely omitted from the ansatz as they are fixed. This will increase the problem sizes the methods can run on.

## 5.4 Noise Models

The simulation noise model results are more expensive than their ideal counterparts to run, and not practically useful in the VQE case, since they do not show much difference between themselves and the ideal results indicating a well set up quantum circuit for a NISQ target. Noise reduction techniques such as the measurement error fitter, likewise are dependent on the calibration of the quantum backend which happens in between long runs and is expensive. They are therefore not pursued.

The noise models can be used as a basis for knowing how many measurement shots to take of the circuit. During simulation it was seen that 1000 shots is adequate for good results and does not significantly contribute to the run time of the algorithm. In the NISQ case finding the optimal number of shots for the problem would require fine-tuning parameters in conjunction with other parameters which is not feasible. It would also not provide explanations as to why those parameters are best. The noise models may also be used to provide estimates for the number of iterations required by, and the performance of, the optimisers in the presence of noise.

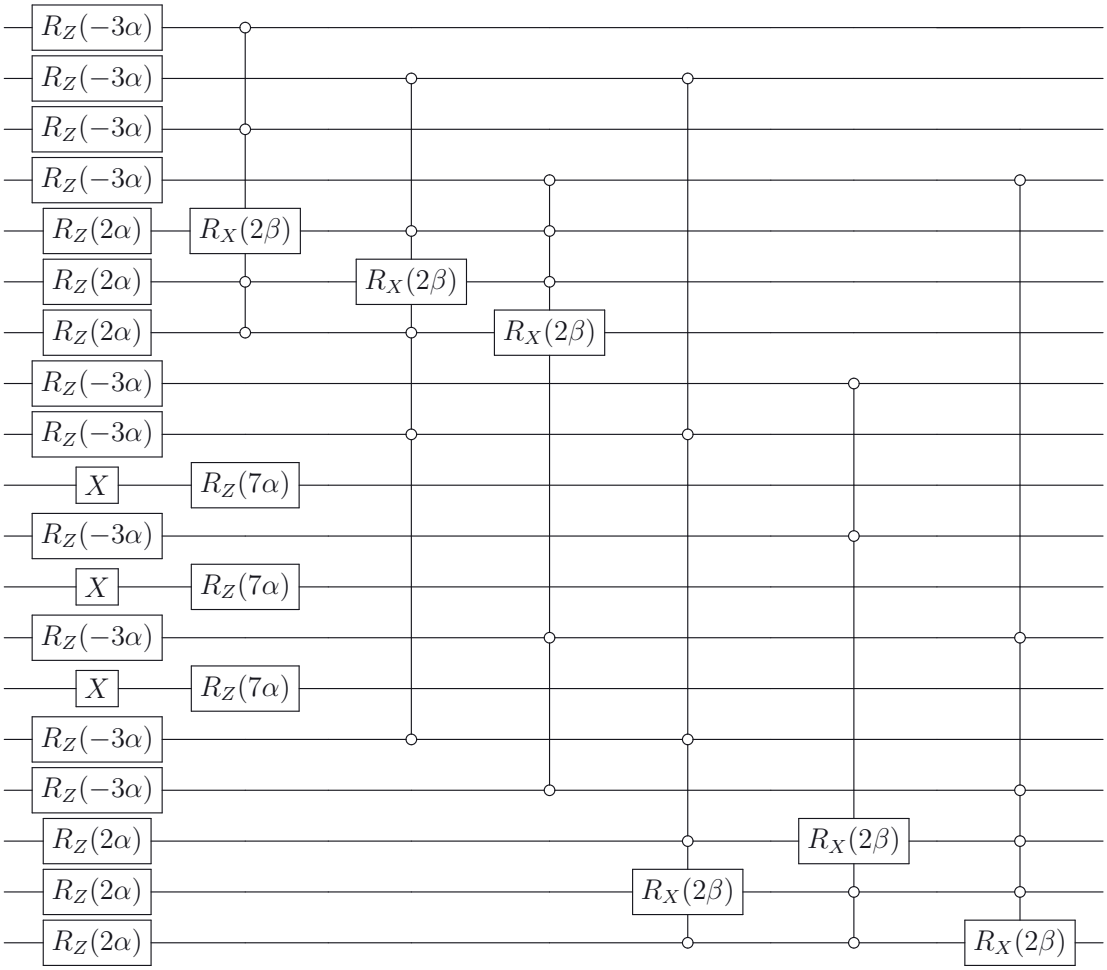


Figure 5.12: A partially optimised ansatz corresponding to the half subtractor graph in figure 5.10 with no repetitions to be used with the QAOA solver.

## 5.5 Parameter Optimisers

A key factor to consider in choosing optimisers is based on the difference between optimiser iterations and objective function evaluations. For many optimisers these vary independently and whilst few iterations are required for the energy of the parametrised circuit to converge, each iteration may require one or more energy evaluations or objective function evaluations of the circuit. This can be expensive in terms of QPU time.

Gradient descent methods are generally not used for two reasons: the number of circuit evaluations performed for optimizing the parameters grows with the number of parameters and there are often flat regions in the cost functions parameter space called barren plateaus, which do not allow for gradient-based optimization [112]. However, since these particular problems are small enough for barren plateaus to be unlikely this issue has not occurred and gradient-based methods are still considered live options.

COBYLA only performs one objective function evaluation per iteration regardless of the number of parameters present in the circuit. However, it is more sensitive to noise in the optimisation process. Because of this COBYLA is commonly used for simulations where noise is not a factor.

SPSA is used in cases where there is noise in the energy evaluations. However, it comes at the cost of two evaluations per iteration, although this is also independent of the parameter set size. It additionally requires extra objective function evaluations to calibrate and during this stage the energy fluctuates without converging. It functions by perturbation of the VQA parameters to approximate the gradient and can also find global minima [113, 114]. For the aforementioned reasons it is the standard optimiser for use on QPUs.

Conventionally the gradient descent parameter update rule uses the gradient calculated with respect to the parameter or weight space. It does not distinguish how much a parameter may influence the cost function evaluations. By using the Fisher

information matrix $F$ the direction of steepest descent in the output distribution space rather than the parameter space may be calculated as follows in equation (5.5):

$$\theta_{t+1} = \theta_t - \eta F^{-1} \nabla \mathcal{L}(\theta). \tag{5.5}$$

The quantum output distribution space may be represented by the Fubini-Study metric tensor $g_{ij}$ which is the analogue to the Fisher information matrix. The pseudo-inverse of the metric is represented as $g^+$ [115]. The parameter update rule is then adjusted to equation (5.6)

$$\theta_{t+1} = \theta_t - \eta g^+(\theta_t) \nabla \mathcal{L}(\theta). \tag{5.6}$$

The Quantum Natural Simultaneous Perturbation Stochastic Approximation (QN-SPSA) approximates the gradient using SPSA, and updates the parameters using equation (5.6). It only requires a constant number of circuit evaluations. It uses four more function evaluations to estimate the ansatz fidelity than SPSA, but it converges to a solution more rapidly than SPSA and is thus the optimiser used for VQE [116, 117].

## 5.6 Chapter Summary

Instances of functional but small logic circuits that scale one gate at a time are built programmatically with combinations of errors in the gates that do not comport with the circuit specifications. The file format of the problems that are saved to disk is explained. The specification of the circuits in PMAX-SAT and the format saved to file is also described as well as the process that converts these files into graph optimisation problems. The adaptive Hamiltonians and ansatz that are required for solution enumeration are explained in terms of biasing solutions with weights and initial quantum states respectively. This includes a custom method for reducing the ansatz depth as well as a suggestion that will reduce the ansatz width. The use

of noise models to estimate some of the parameters in the optimisation process is discussed. Throughout different examples relating to the actual problems are provided as illustrations of the complete quantum algorithm solution. Finally, the relative merits of a few ansatz optimisers are discussed and the chosen optimiser QN-SPSA is given a brief explanation.

Chapter 6 presents the results of the various solvers and their comparisons.

# Chapter 6

# Results

This chapter summarises the results from the runs of the various algorithms as graphical comparisons. It presents the data as summaries of the individual runs. The results are given for scaling of the problem encodings, algorithm run times and spatial elements of the ansatz. Lastly, the accuracy of the solvers is summarised.

## 6.1 Overview

Individual solutions are sorted and filtered so that graphs are not overrun by the many unlikely solutions. The solutions are sorted by their probability amplitudes which in these cases will be real numbers. The topmost likely states are then used to graph the bit string sampled probabilities for the individual problems. The information about the individual runs such as the output distributions are all available in the results repository. However chapter 6 deals with summarised results across many runs unless otherwise specified.

## 6.2 Problem Encoding

Figure 6.1 shows how the number of qubits required for a solution increases as the number of gates in the combinational logic circuit does. Since the number of qubits and the number of vertices in the graph problem encoding are identical, the number of vertices and qubits are used interchangeably depending on the context.

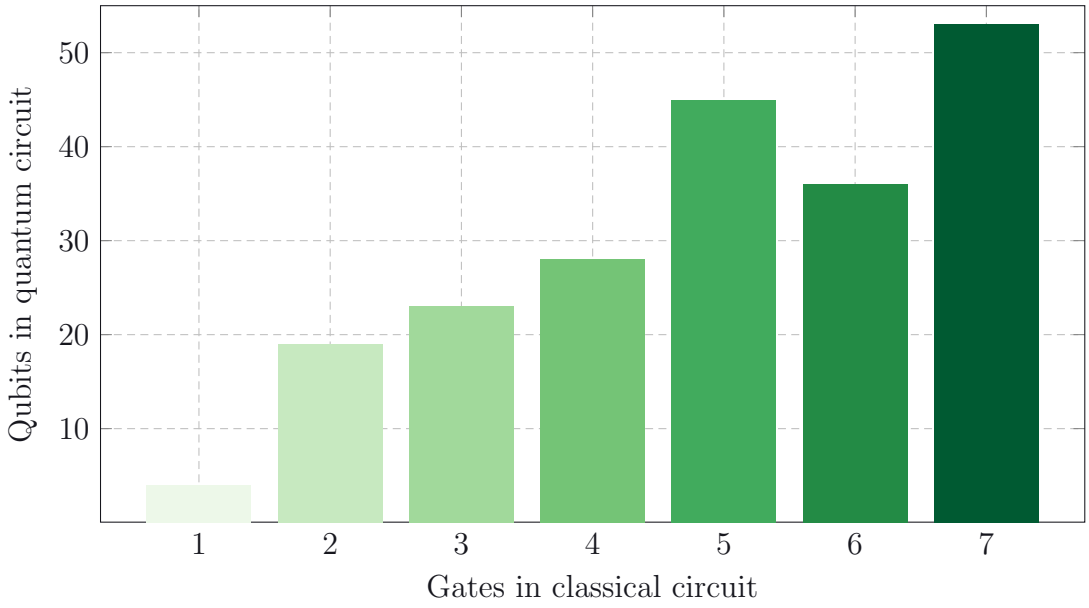


Figure 6.1: The relationship between graph vertices, or the number of qubits, as the number of classical circuit gates increase.

The number of qubits required increases linearly with the number of gates in the circuit. The large increase from one to two gates is due to the two gate circuit containing an `XOR` gate (see figure 5.2) and the one gate circuit only containing a `NOT` gate (see figure 5.1). The corresponding Tseitin transforms (see table 4.1) for the two gates contain the most and least literals respectively. Note that figure 5.6, since it has gates that transform into fewer literals has fewer vertices than figure 5.2. It follows that this problem will execute faster.

## 6.3 Scaling of Solver Solutions

Comparison of the individual solutions is difficult because the enumeration of the solutions between the solvers is not always in the same order so averaging the results, all else being equal, makes the solutions comparable. This is the case because of the stochastic nature of the approximate algorithms and finding a seed that corresponds to the classical enumeration order requires almost valueless post hoc analysis.

### 6.3.1 Classical Solvers

Figure 6.2 shows the scaling of the classical VQA simulations and the exact solver. The solutions have an almost identical shape since they all minimise the same Hamiltonians with the same scaling in Pauli terms. The curves also serve to show the exponential scaling of the exact solver algorithm as well as the inefficiency of classically simulated quantum algorithms.

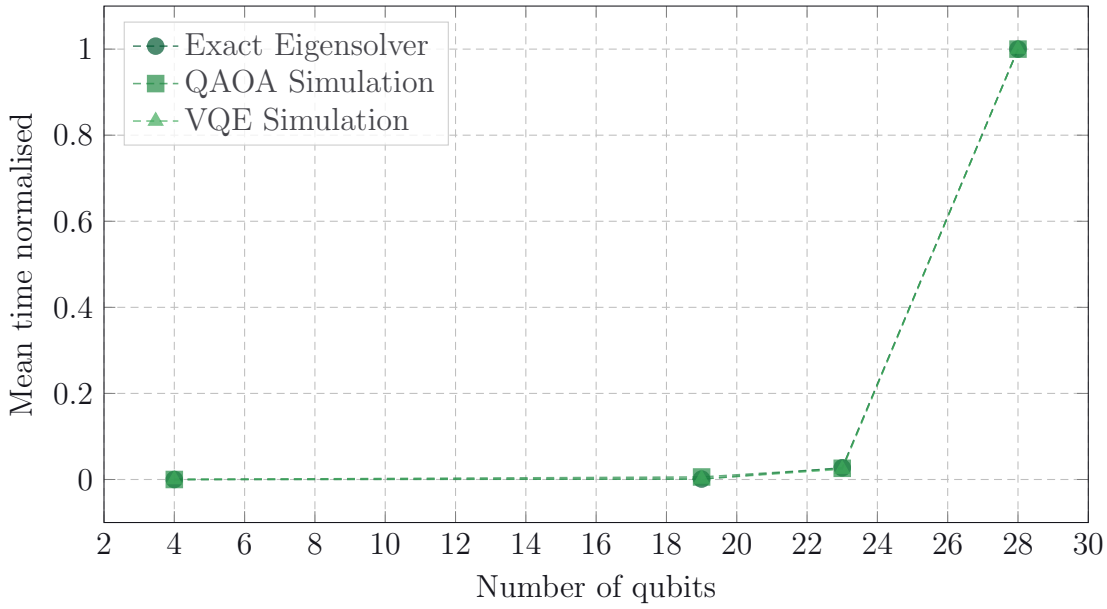


Figure 6.2: Comparison of the averaged runtime of classical simulations of the quantum solvers and the exact classical solver. The time is normalised per solver so the scaling rather than the speed of the solvers are apparent.

### 6.3.2 Hybrid Algorithm Components

A distinction needs to be made between the two timings. The utilisation of the Qiskit Runtime VQE program records both the timestamps of the QPU processing as well as an overall optimiser time. The optimiser time is the entire execution time of the VQA.

However, the optimiser time includes time waiting in the queue for QPU access. It is therefore an unreliable metric to average runs over. Because of this, the many runs are carefully monitored and only those that started immediately or close to immediately were used in the summaries (although the other data are still available). The optimiser and QPU time for each problem is recorded in figure 6.3.

The QPU time is also subject to variance if a run occurs across one of the hourly QPU calibrations. However, this is deemed a part of current quantum computation and so no mitigating measures are taken for this. It is also obvious that the QN-SPSA optimiser which has a set number of evaluations per iteration will be constant in the evaluations and linear in the iterations.

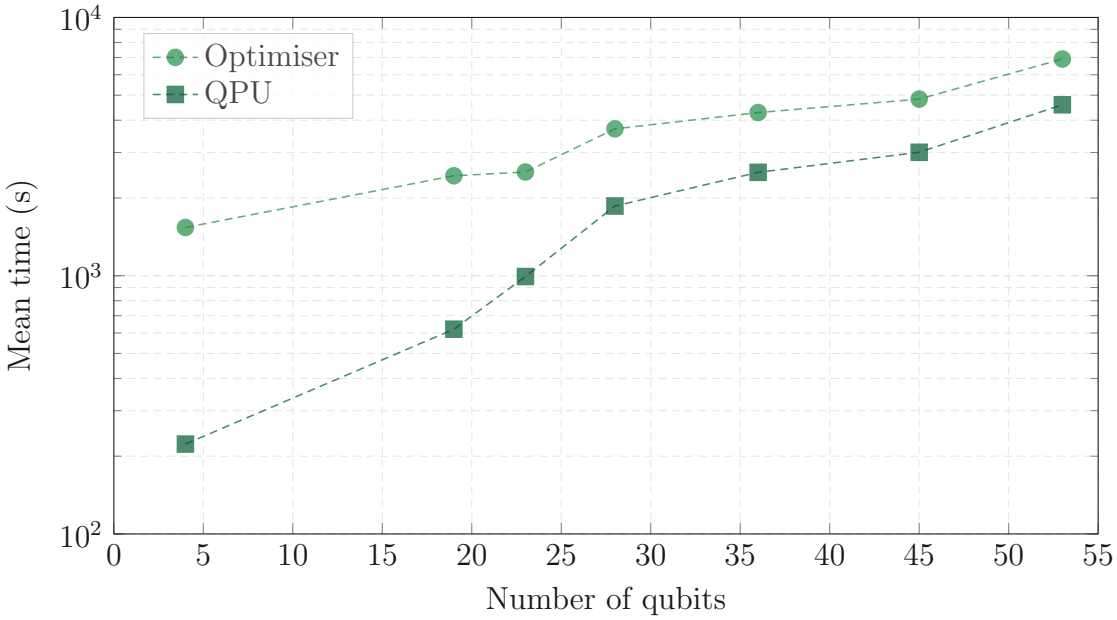


Figure 6.3: A comparison of the average total optimiser time of the VQE solver and the average time on the QPU as the problem size increases.

### 6.3.3 Memory

For exact solvers of the 45 qubit problem 256 TiB of memory is required for storing an array with 3518437208883 rows, for the 53 qubit case 64 PiB for an array with 9007199254740992 rows is required. This is because classically the array manipulations required happen in memory and the arrays are simply too large. This shows that the graph approach is not feasible at all classically. Because of this subsequent results do not have exact classical or classically simulated quantum solver result comparisons for all the data points.

### 6.3.4 Quantum Algorithm Comparison

The exact eigensolver solver is used without the classically simulated quantum algorithms for comparison in figure 6.4 since its curve is so similar. Additionally, the classically simulated quantum algorithms have as their goal being run on quantum

hardware. The eigensolver does not scale as well as the VQE. Rough curve fittings using guesses based on the expected shapes (note the curves are fit then normalised) of the scaling is given. The fits are to get a general idea of the shape of the data. It is clear from the general shape of the plots that the VQE algorithm does scale at least better than the exact classical algorithm.

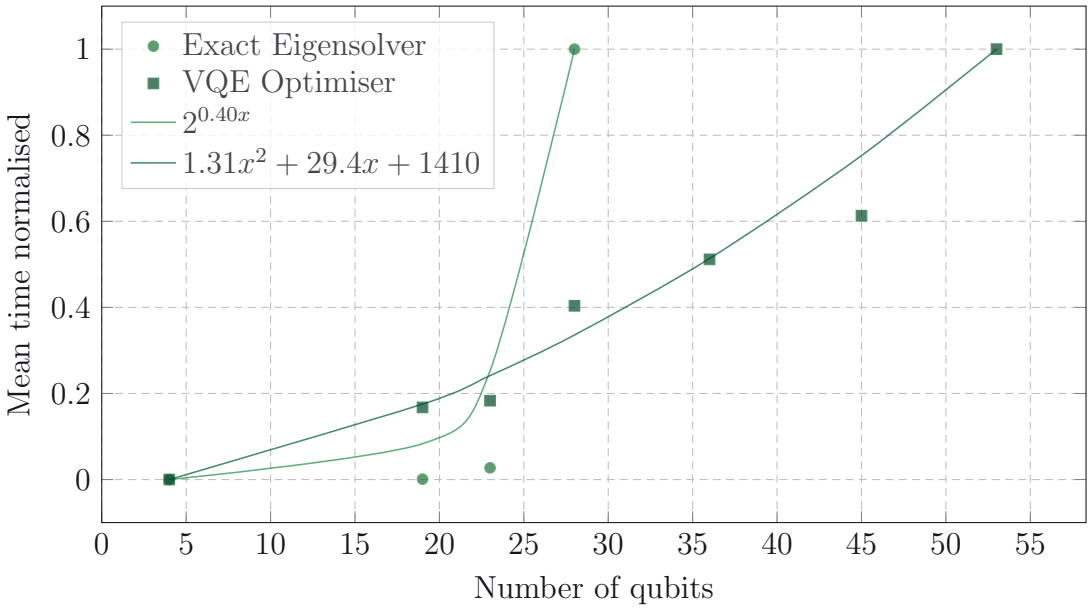


Figure 6.4: A Comparison of the averaged runtime between the exact eigensolver and VQE. The time is normalised per solver so the scaling rather than the speed of the solvers are apparent. The continuous functions are fit based on the shape of the discrete data before normalisation.

In figure 6.5 the curves are fit against the scaling of the gates. This has a significant effect on the exact eigensolver, and it also shows that the approximate classical solver and the VQE scaling is comparable. The solution for the 2-4 binary decoder in figure 5.6 takes less time on the M22 solver even though it has an extra gate as expected. Yet because the number of iterations is increased linearly in the QN-SPSA algorithm this same effect is not as pronounced as on the M22 solver, and the data points between five and six gates are swapped relative to figure 6.4.

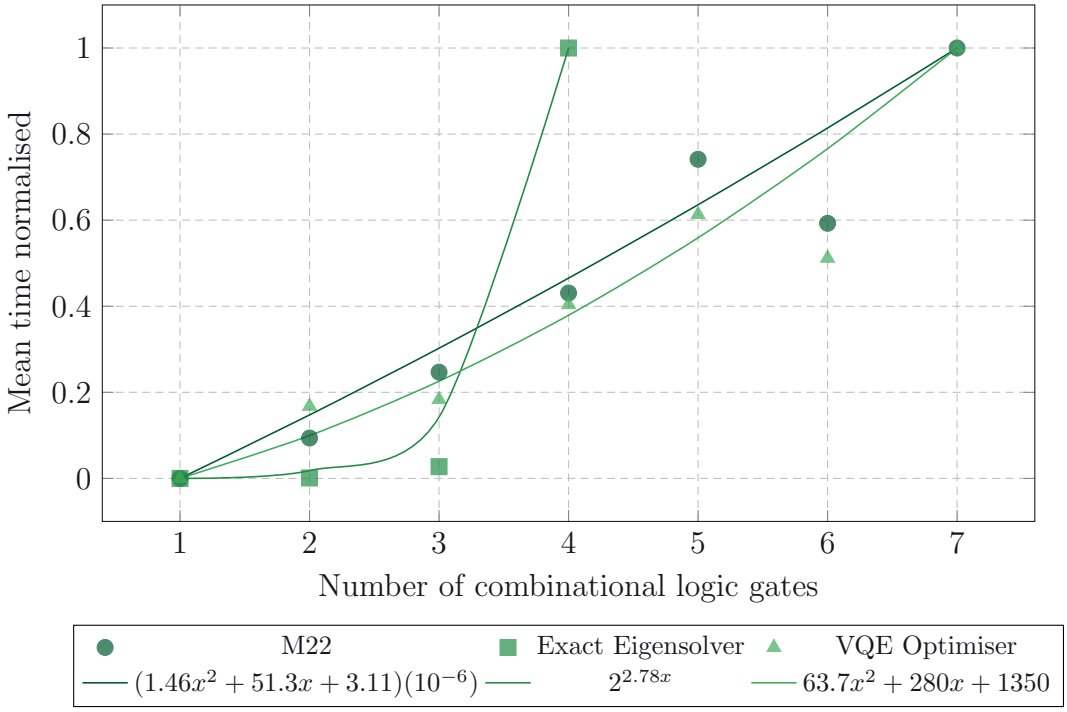


Figure 6.5: A Comparison of the averaged runtime between the exact eigensolver and VQE and the approximate classical solver M22. The time is normalised per solver so the scaling rather than the speed of the solvers are apparent. The continuous functions are fit based on the shape of the discrete data before normalisation.

Figure 6.6 shows the same data but with respect to the number of clauses in the CNF.

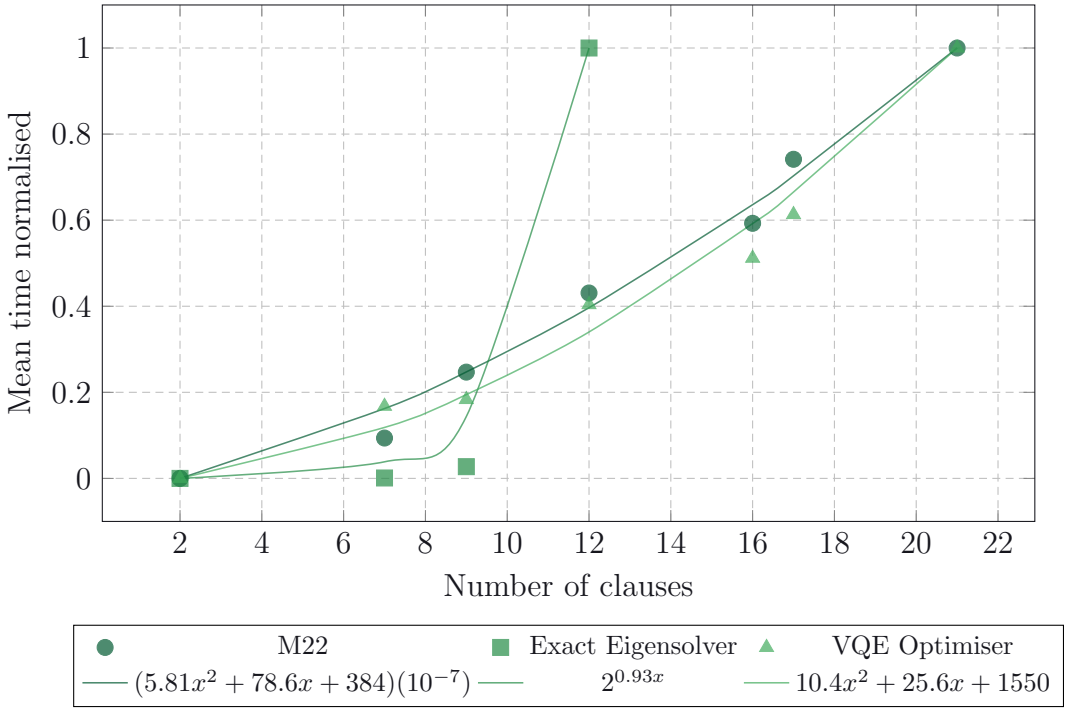


Figure 6.6: A Comparison of the averaged runtime between the exact eigensolver and VQE and the approximate classical solver M22. The time is normalised per solver so the scaling rather than the speed of the solvers are apparent. The continuous functions are fit based on the shape of the discrete data before normalisation.

Figure 6.7 has the graphs time-shifted lower by their first data point and then further down by the smallest point on the graph so log(0) is not plotted. This only marginally decreases the overall absolute accuracy of the data points (because the smallest point is insignificant relative to the total times) and indicates the significant absolute speed advantages of the M22 solver. But more to the investigation, the scaling difference can be more clearly seen on a log axis. This is the clearest indication of the scaling advantage of the quantum solution over exact classical solutions.

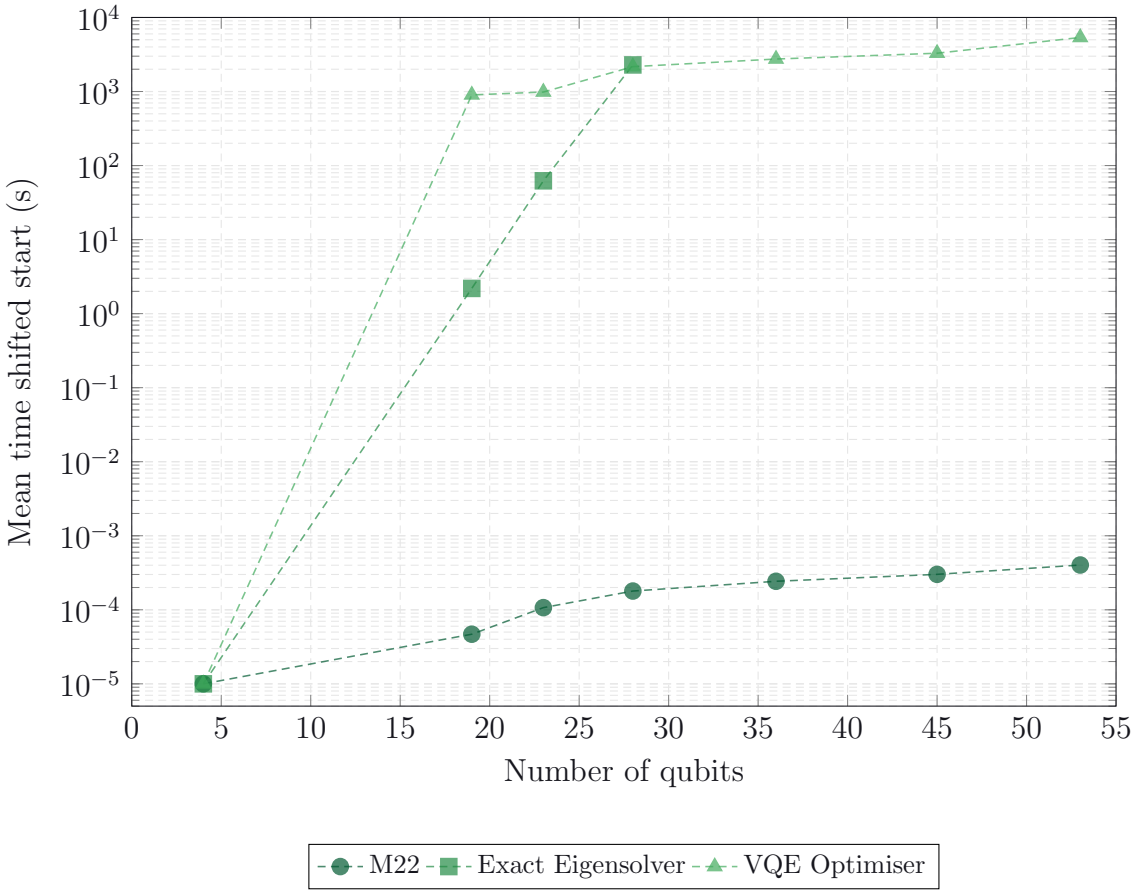


Figure 6.7: A Comparison of the averaged runtime between the exact eigensolver and VQE and the approximate classical solver M22. The speed of the solvers are apparent as well as the scaling since the data are shifted to start at the same smallest runtime point.

## 6.4 Quantum Resources

Figure 6.8 shows how many quantum gates of a particular kind are required in the ansatz for each of the quantum algorithms. The QAOA ansatz which in this case was repeated once ($p = 2$), requires a large number of `CNOT` gates, this is because these gates allow flipping qubits based on other qubits and therefore constrains the solution

to feasible states. This shows it is impractical to run on the current hardware. The custom VQE ansatz however is shallow enough for current hardware. Since CNOT gates are a major source of noise in quantum computation the accuracy of the results increase with decreased CNOT usage and vice versa. The gates in figure 6.8 are the transpiled circuits before they are run on specific backends and therefore are less in quantity than on the QPU. However, the shape of the trend will be similar.

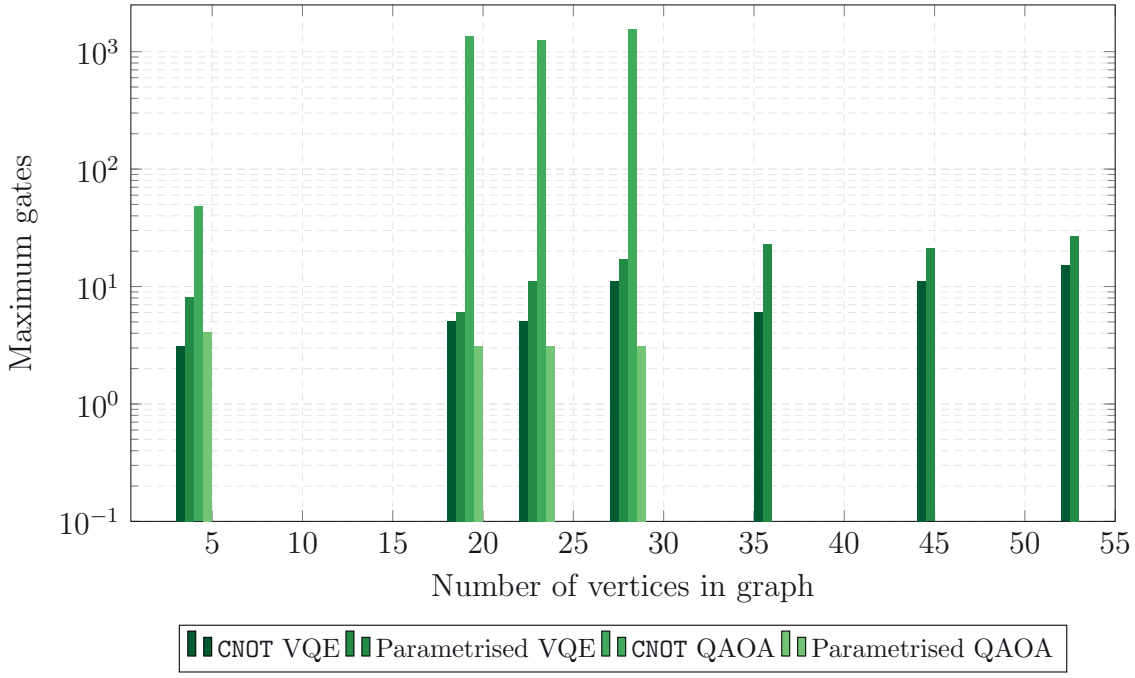


Figure 6.8: The maximum number of CNOT and parametrised quantum gate resources required by each of the custom ansatz as the size of the problem increases.

The overall depth of the respective ansatz in figure 6.9 shows that the ansatz depth for both does not change much as the problem size increases except in the case of the first problem where the custom ansatz is not made use of as it does not have any tunable parameters. The VQE algorithm not increasing in size is very useful for NISQ hardware.

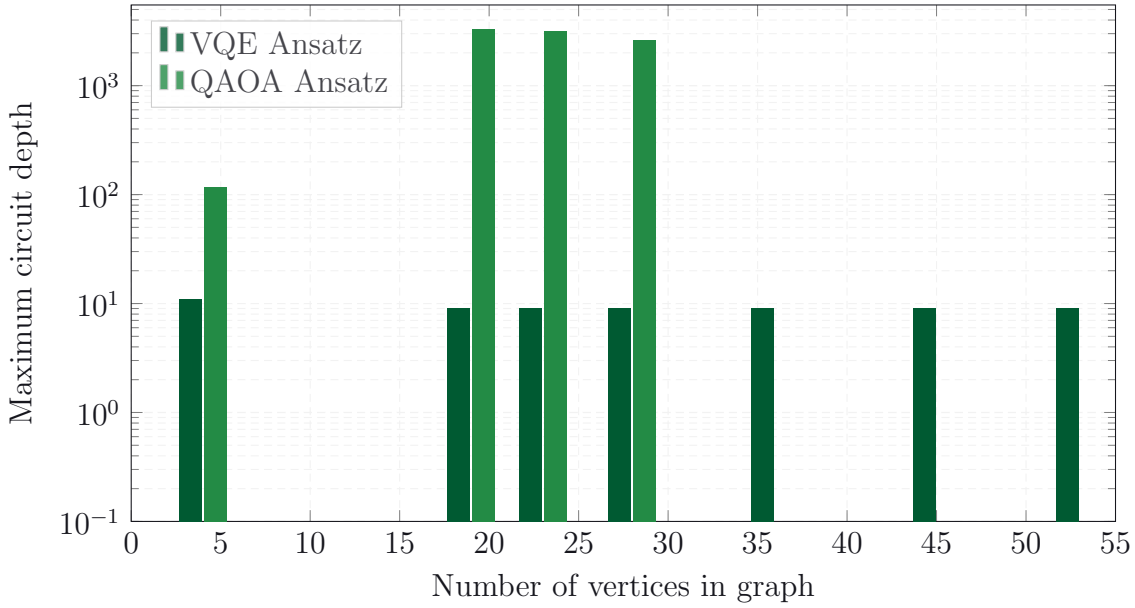


Figure 6.9: Comparison of maximum ansatz depth for the VQAs as the size of the problem increases.

## 6.5 Accuracy

Since the scaling of the VQE algorithm and the M22 solver are comparable (although M22 does maintain a speed advantage) and both solutions are approximate solutions, the accuracy of each solution may be investigated to see if any advantage is conferred by using the quantum algorithm. The M22 finds the MLCSCs and not the MCSCs and is, therefore, an approximate algorithm. It does however find all the MLCSCs and the plots are for the time to enumerate all of them rather than just one whilst all of the other solvers are used to retrieve a MCSC each iteration. Therefore the M22 solver gives exact solutions when compared to the eigensolver so long as the smallest MLCSC is picked out.

The accuracy of the solutions is judged by comparing the energy of the solutions, and whether or not the graph colouring is a MIS. The final accuracy of whether or not

the suspect gates are the same is not calculated, although even with a suboptimal energy a correct MIS may be found, and even with a suboptimal MIS, the correct gates may be found. This means the transforms are robust to deriving good solutions out of suboptimal solver results.

However even as the comparisons of the energy are averages, some of the solutions using the VQA solvers are not found due to either insufficient iterations or to noise in the case of the QPU solutions. The stopping conditions on the classically simulated VQA stop short of providing every possible solution on occasions where there is decreased accuracy. Due to long compute times on the QPU the VQE solution deliberately only attempts a small subset of solutions for each problem.

### 6.5.1 Eigenvalues

Figure 6.10 refers to three energy measurements for each of the two solvers. The optimiser eigenvalue is the energy of the solution at the end of the optimiser iterations. The approximate expectation comes from sampling the expectation of the most likely bit string solution on the problem Hamiltonian. This value is, therefore, lower than the eigenvalue the iterations stop on. The exact solution is the energy value of the exact bit string on the expectation of the specific solver Hamiltonian. This provides a way to have a legitimate comparison even though the cost functions and ansatz may be different and updating during the enumeration. The eigensolver energy is the energy of the exact solver.

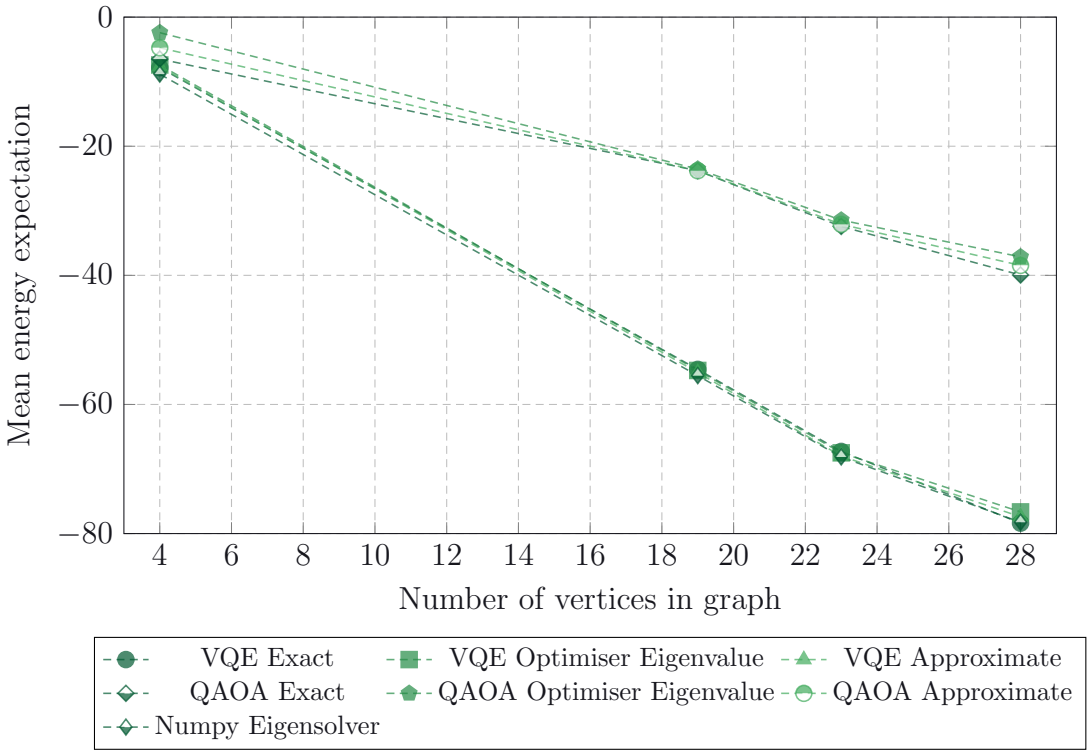


Figure 6.10: Classical simulations of quantum solvers and exact eigensolver average energy expectation values as the problem size increases. The expectation values of the optimiser over the ansatz and the most sampled solution (the approximate value) are compared to the exact and lowest expectation values for the simulations of the quantum solvers.

From figure 6.10 it can be seen that the approximate energy expectations are very close to the ideal exact measurements. When they are not, since these are classical simulations of a quantum system it is to be attributed to finite shot measurements and early stopping of iterations and in cases where it is lower, it is due to the average value of the energies across all the solutions being lower due to omission of exact solutions that do have larger (the energies being negative) than average values.

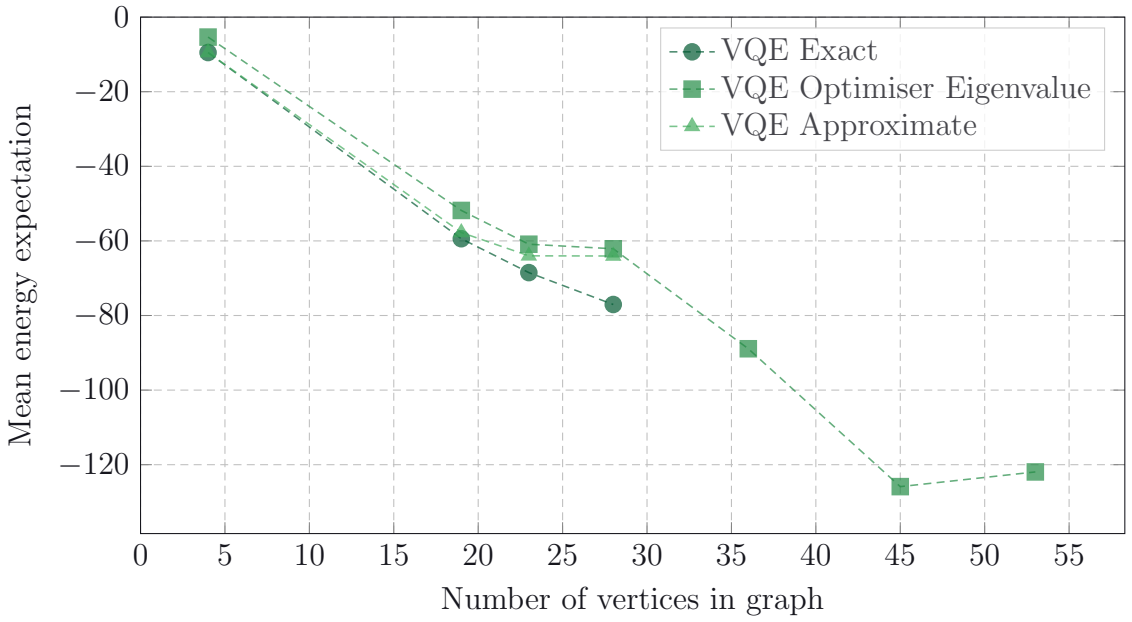


Figure 6.11: A comparison of VQE average energy expectation values as the problem size increases. The expectation values of the optimiser over the ansatz and the most sampled solution (the approximate value) are compared to the exact and lowest expectation values when classical exact solutions exist

The same energy comparisons are plotted in figure 6.11. The approximate solutions are close to the ideal values but start to diverge on the bigger systems as is to be expected on NISQ devices.

### 6.5.2 Error Calculations

Since non-optimal energy of the optimisation may be enough to arrive at a good MIS solution different error or accuracy metrics are introduced to describe the validity of the MIS.

The maximalism of a solution can be found by using the objective function defined in equation (4.10), for each approximate and exact solution in the following equation (6.1). This checks how close the solutions maximisation objective is to the exact solutions

such that a solution that includes more coloured nodes than it ought to is also penalised. It may be the case that this will happen in the case that the overshot objective is an IS but has ignored constraints. It is assumed that (which is the case for all problems in this work) $f^a(x_v)$ is much larger than $f^e(x_v)$ which will make $f_m$ negative.

$$f_m = 1 - \frac{|f^e(x_v) - f^a(x_v)|}{f^e(x_v)}, \qquad |f^e(x_v) - f^a(x_v)| > f^e(x_v). \tag{6.1}$$

Equation (6.2) defines the independence of the set. This factor heavily penalises error, if two adjacent nodes are coloured it is counted as an error on both of the nodes. Take $V(C)$ to be the coloured subset of the graph $V(G)$, a point can be awarded to each uncoloured node since they can never violate the independence constraints. A point can also be awarded for each of the coloured nodes that do not have coloured neighbours. Finally, the normalisation factor divides the awarded points.

$$f_i = \left( \sum_{v \in V(G)} 1 \right)^{-1} \left( \sum_{v \in V(G) \setminus V(C)} 1 \quad + \sum_{v \in V(G) \setminus (N(v) \cap V(C))} 1 \right). \tag{6.2}$$

Equation (6.3) is a combination of equations (6.1) to (6.2) and gives an overall idea of how close the solutions are to a MIS:

$$f = f_m f_i. \tag{6.3}$$

It should be noted that since $f^e(x_v)$ is dependent on using the exact eigensolver cost over the graph there are no values for $f_m$ where the graphs are too large to optimise classically. However, it is not feasible in principle to read the MLCSC result from the M22 solver and find a graph solution corresponding to the MLCSC since knowing which clauses are satisfied does not give information about which literals are true. It is also the case that MLCSC may not be equivalent to the MCSC.

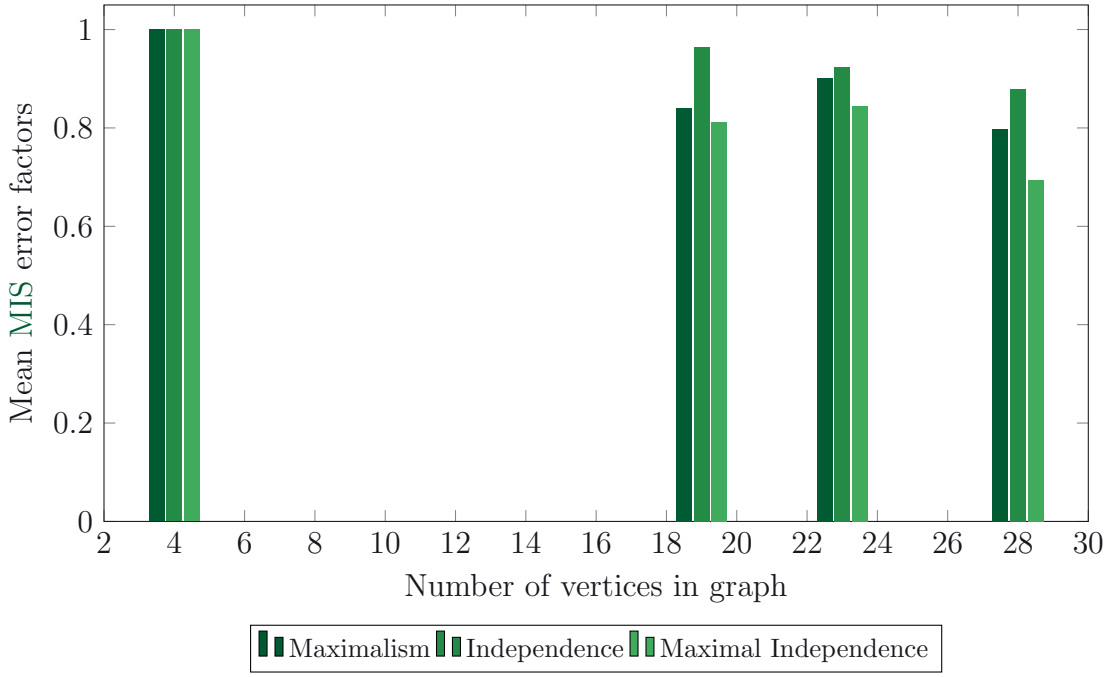


Figure 6.12: The error factors (larger values are better) for the maximum independent set as the as the size of the problem increases for the VQE solver.

Figure 6.12 shows the average results of these metrics for the VQE solver. As mentioned these results fluctuate in conjunction with the number of `CNOT` gates required by the ansatz.

## 6.6 Chapter Summary

The results are aggregated over individual runs of problem instances. The results are presented as graphical comparisons between the solvers. The data is plotted against the scaling of the problem sizes in their various forms (gates, vertices or qubits and clauses). The scaling of the required qubits is considered first and shown to be linear concerning the number of gates in the circuits. The run times of the classical simulations of the quantum algorithms along with the exact classical solver are shown to be infeasible. The spatial scaling of these solutions is also shown to

be infeasible. VQE is shown to scale in P in comparison to the classical cases. The classical approximate solver seems to scale in P and is highly efficient. The custom VQE ansatz is shallow enough for the NISQ devices, but QAOA is far too large. The expectation energy measurements for each of the solvers are compared and methods to make them comparable are used. The two classical algorithms give exact solutions, even though M22 is an approximate solver, whilst the classical simulations of the quantum algorithms are almost exact. The accuracy of VQE lowers as the problem size increases which is most likely caused by entangling gate errors.

Chapter 7 makes further comments on the results of this chapter.

# Chapter 7

# Discussion

The structure of the discussion follows the same structure of chapter 6 and comments on its contents. The focus is on what can be said from chapter 6, due to the nature of the problems there are not many data points and inference to the best explanation of the data are required. Where improvements naturally follow from the results they are mentioned. Distinctions to separate the approximate classical solver and the Variational Quantum Eigensolver (VQE) are made. The temporal and spatial requirements of the algorithms are discussed. The merit of the solvers as current solutions are also discussed, as well an unexplained result from a previous approach to the Quantum Alternating Operator Ansatz (QAOA). Conditions under which the quantum algorithm may obtain an advantage are mentioned.

## 7.1 Problem Encoding

Although the number of qubits scales linearly with the number of gates in the problem more may be done to reduce the number of qubits required. This may be done by reducing the size of the circuits to a formally equivalent circuit. Reductions to the number of edges produced by each of the equivalent circuits are also to be considered as this reduced the required entanglements required for the ansatz. However, doing

this will also introduce additional processing steps to recover the fault in the original circuit.

## 7.2 Scaling of Solver Solutions

It is important to note that any single MLCSC only approximates a MCSC. However enumerating all the MLCSCs and finding the smallest one amongst them is the MCSC. In the M22 case the MLCSCs are enumerated so that even though each solution is an approximation the entire solution can be used for MAX-SAT problems. However it seems to be the case that the MLCSC are MCSC for these small problem instances.

The upper bounds of the number of Maximal Independent Sets (MLISs) which corresponds to the MLCSC in a graph have been calculated in and is given by equation (7.1) [118, 119]. This set of problems have $v \bmod 3 = (1, 1, 2, 1, 0, 0, 2)$.

$$V(G) = \begin{cases} 3^{\frac{v}{3}} & \text{if } v \equiv 0 \pmod 3 \\ 4 \cdot 3^{\frac{v-4}{3}} & \text{if } v \equiv 1 \pmod 3 \\ 2 \cdot 3^{\frac{v-2}{3}} & \text{if } v \equiv 2 \pmod 3. \end{cases} \tag{7.1}$$

The number of MLIS and therefore MLCSC scales exponentially in the solution space. This is problematic because even though enumeration using M22 is very efficient when the problem becomes large enough it will hit a knee point and have to enumerate exponentially many MLCSC to find the smallest of such problems. Benchmarks for the approximate classical solver can be seen in the work "On Computing Minimal Correction Subsets" and "Literal-Based MCS Extraction" which contains the implementation details of the solver and a related algorithm in [120, 121].

This raises the concern about the applicability of the function fitting in the previous section, since a local portion of the plot may be piecewise polynomial but globally exponential.

Since the VQE algorithm only finds the MCSC it may not suffer this same issue as

the problem sizes increase. Additionally only finding one of the MCSC is sufficient for suspect errors. It would then follow that VQE may obtain an advantage on larger scale quantum devices.

An additional two scaling problems concerning problem generation remain, although they are not considered a part of the scope of the research. First, to solve the problem for every row in a truth table make it such that even if the solvers are efficient that the solvers are run exponentially many times in the input space of the problem. Secondly, the number of solutions to the problem create another search problem for the actual error amongst potential errors.

### 7.2.1 Classical Solvers

Although the exact classical solver is not efficient the M22 is highly efficient for small problem sizes and is to be preferred until a comparison can be done on QPUs that are both much larger and more robust to noise.

### 7.2.2 Hybrid Algorithm Components

If possible the entire set of problems without queuing interruptions should be sought. If quantum computing becomes robust enough to require infrequent calibrations this will also have an impact on the results of the longer runs.

### 7.2.3 Memory

Both the VQA and M22 solvers can run on problems too large for the exact classical solver. The exact eigensolver ought not to be used practically except in the cases of benchmarking solution energies or run times.

### 7.2.4 Quantum Algorithm Comparison

Since the preprocessing and postprocessing of the results in the VQA algorithms is not included in the problem scaling as opposed to the M22 solver which does not require

as much pre and post-processing, it may be a concern that the additional time may make the scaling plots incomparable. However, all the additional processing steps are in either linear or polynomial time. The addition of the processing steps, unless there are exponentially many additions, will likewise yield a linear or polynomial time.

VQE has been shown to scale in the neighbourhood of as well as the M22 solver for small problem instances and outperform the exact eigensolver in figure 6.7. Although it suffers from inaccuracies this is due to the NISQ hardware and it may yet outperform M22 in cases where there are exponentially many MLCSC to be enumerated for a solution.

## 7.3 Quantum Resources

Owing to the constrained nature of the QAOA, fewer iterations are required than in non-constrained variational algorithms. Nevertheless, QAOA requires a quantum circuit that is too deep to run on a NISQ device. Even though the number of gates required by QAOA scales linearly as calculated in, and can be seen in figure 6.9 with the problem size, the ansatz is too large for current quantum hardware [85]. Additionally, for greater approximation accuracy the repetitions of the ansatz enlarge the circuit to an infeasible depth.

VQE although usually used for finding ground-states in quantum chemistry and not for combinatorial optimization can use a generic ansatz designed to be small enough to run on the hardware and is, therefore, more appropriate as an immediate solution [105]. The custom VQE ansatz has an almost constant depth and its classical ideal simulations suggest that it is general enough to be accurate. It may be further reduced by removing every wire that already has a value set from the constraints. This applies to the QAOA ansatz too. However, this will require additional mapping transforms and additional changes to the cost Hamiltonians.

Additional depth reductions may be achieved by permuting the order of operators in the ansatz that commute with neighbouring operators. Permuting the order of the

inputs may also at the cost of additional mappings increase the number of commuting operators. This may also lead to reducible operators being placed next to each other.

The ansatz used also have a small number of parameters making the optimisation process search through a smaller space and if other optimisers are used that do scale well with parameter size but have some other advantage it will be more accommodating than more general ansatz.

If the noise problems of NISQ devices are overcome two algorithms become available for use in this domain. This first is QAOA and the second is Grover's algorithm with a quantum oracle built for SAT problems. The QAOA ought to find multiple MIS solutions in a single run, require fewer iterations to optimise, and get arbitrarily close to an exact solution. It is, therefore, a prime choice for this sort of domain, whilst work on Grover's algorithm suggests it may be optimal for unstructured search problems [122].

## 7.4 Accuracy

The states required for accurate solutions are supported by the ansatz used since the classical simulations of the quantum algorithms show results very close to the exact eigensolver energies.

For the current problems since the classical circuits to be solved are not very deep and may have many constraints often the problem in the smallest cases are very close to completely specified. This can be seen by the fact that the energies are close to the ground state values where there are constraints. This means that the classical preprocessing step where preliminary on and off vertices are assigned may already do enough to constitute a solution. However, as the problem size increases the significance of this decreases quickly.

In figure 7.1 the result and optimiser energy is given for a random individual run. It is apparent that even though the initial state of the VQE optimisation process is $|0\rangle^{\otimes n}$ for some $n$ the starting energy is not zero. This is due to the preliminary

classical preprocessing steps.

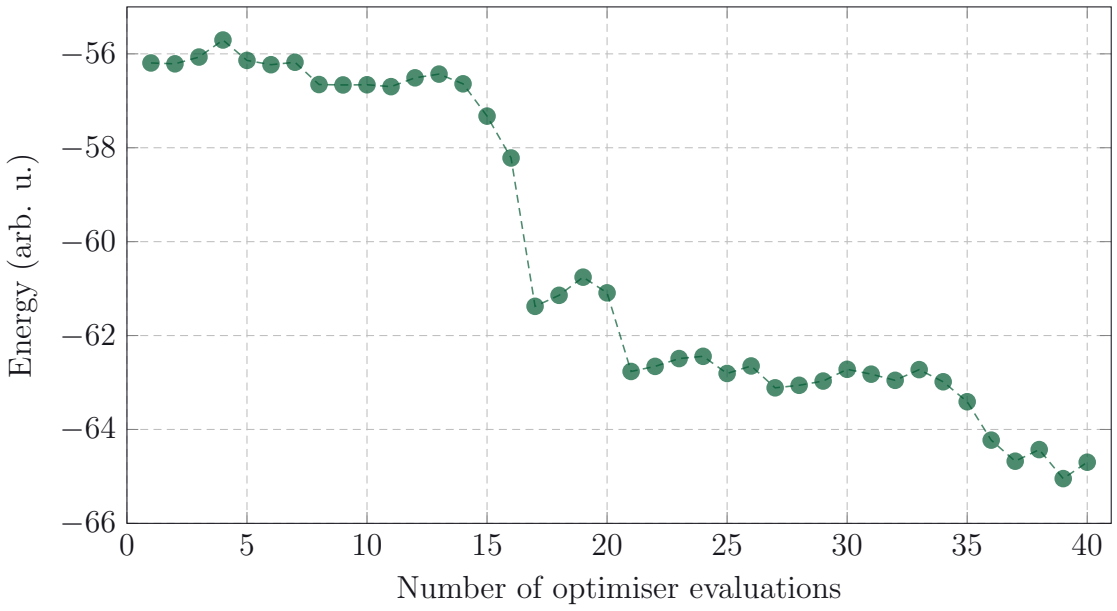


Figure 7.1: The optimiser energy plotted against the optimiser evaluations for the first run on the two-to-one multiplexer with an error in the first `NAND` gate and second row of the truth table as the test vector.

However, it is also apparent that it is not specified from the preprocessing since the energy is still optimised to a lower value.

It is important to note that finding MAX-SAT solutions gives a minimum sized MLCSC. However there may be more than one smallest MLCSC, M22 finds all of the MLCSC and so has much better accuracy in comparison to the quantum algorithms which deteriorate rapidly with size. It is therefore a better solver for accuracy in the NISQ era.

In so far as the usability of any of the solutions, whether they are accurate or not for the debugging problem, there are additional considerations to be made. A potential issue that may occur is in the enumeration of the problems it might be the case that one of the MIS are SAT when later enumerations will yield UNSAT for the same

problem. If this is the case the SAT case will be the maximum case and this will just be a case of the invalid circuit resulting in an equivalent output for the specific test vector. Therefore the extra work needs to be done on carefully picking test vectors that will not give SAT for UNSAT problems, simulation and verification based tools can provide this information [9].

Since the classically simulated quantum algorithms do not show perfect convergence a larger number of iterations may be investigated for the algorithms. Barren plateaus (areas where the optimisation space is flat) can halt finding minimums during optimisation where gradient-based optimisers are used. They are not present in the problem set but mitigating procedures or other optimisers need to be pursued for larger problems. This does not affect any of the classical solvers.

### 7.4.1 Eigenvalues

An inner join on the solutions found for all the problems can alleviate the slight offset in average energies due to partial solving of the full problem set for some of the solvers.

### 7.4.2 Error Calculations

Although the ansatz is hardware efficient they require optimisation for particular backends that could not be performed using the newer Qiskit runtime approach without changes to the cost Hamiltonians and additional mappers. The noise of unoptimised `SWAP` gates is a large concern for NISQ algorithms. This is most likely a significant contributor to noise, as suggested by the error factors inversely following the shape of the `CNOT` gates in the ansatz.

The actual error calculations for independence also penalises errors twice for each error since each error affects more than one node. This should be kept in mind when viewing figure 6.12.

## 7.5 Unexplained Results

The QAOA implementation in Qiskit only produces one of the MIS solutions in the solution space. However previous analysis using Pennylane produces all of the MIS in a single run.

Figure 7.2 shows preliminary results of running the QAOA algorithm on the example problem in figure 4.2 with Pennylane using a Tensorflow GPU backend and Adagrad optimiser at different levels of discretisation [123]. For $p = 3$, it can be seen that the most frequently sampled bit string corresponds to the solution in figure 4.3, whilst the second most sampled solution also presents an MIS. A sweep through the parameter space yields only feasible solutions for both of the implementations.

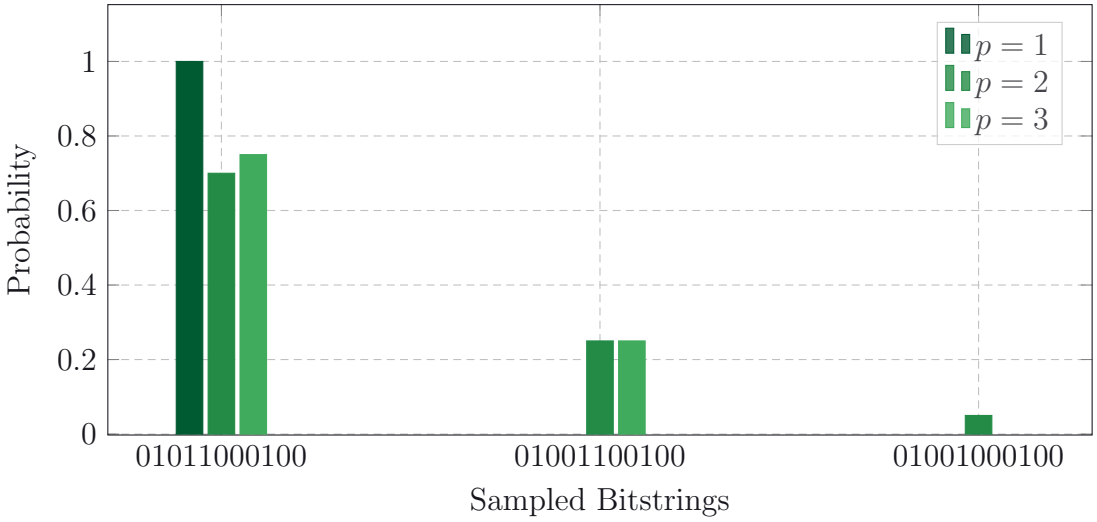


Figure 7.2: The output bit string probability distribution for the Pennylane implementation of QAOA at differenet levels of ansatz repetitions.

## 7.6 Chapter Summary

The scaling of M22 is shown by a mathematical result to be impractical. However, it is recommended approach until much larger and more noise-robust quantum hardware

is available. Ongoing experiments will have to update whether or not the quantum algorithms can surpass solvers like M22 in practicality. The quantum algorithms can currently give results that classical exact solvers cannot due to spatial considerations, however, due to noise the accuracy is degraded as the problem sizes increase and it is not reliable enough as it stands. When the performance of QPU make VQE reliable it may be the case that other methods such as QAOA or Grover's algorithm may be better-suited algorithms as they confer extra benefits over VQE. Nevertheless, a quantum advantage is a strong possibility indicated by the current data and proofs available.

Chapter 8 makes concluding remarks on the research presented in this dissertation.

# Chapter 8

# Conclusion

This research seeks to provide NISQ algorithms that solve a circuit debugging problem more efficiently than available classical algorithms. The research expands into domain-specific experimentation of quantum optimization, providing the tools to make the problem amenable to QAOA and VQE. Although careful considerations were given to reduce circuit depth, width, and noise, the QAOA formulation is not feasible on current NISQ devices. VQE with a modified ansatz for constrained problems and modified Hamiltonians for solution enumeration was investigated for short term small problems on NISQ hardware. Two classical alternatives, one exact and one approximate were used to benchmark the scaling and accuracy of the solutions.

Quantum computing has been used to extend the available methods available for solving a manufacturing problem in the design of VLSI hardware. Multiple gates errors can be found automatically using all the solvers investigated. Since the quantum debugging methods show evidence of scaling better than the classical alternatives, and already outperform the exact algorithm in both temporal and spatial scaling, the solvers runs should be revisited as progress is made on the quantum hardware. M22 or a similar algorithm is recommended as a current approach, both because the accuracy of the quantum approach degenerates with the problem size and because of its slow speed of execution.

Because a solution to any of the NP-Complete problems provide a solution to any of the others this approach could be profitably used to guide new research for other engineering applications using quantum computation.

## 8.1 Contribution of Dissertation

The research area, in general, is new and there is not much on application-specific quantum computation although the field is moving forward quickly and more applications have come to fruition since the inception of this research. However, the following contributions have been made:

1. The application of quantum solvers for combinational logic circuit debugging. Namely VQE and QAOA. Using existing methods to transform the problem into the required structures.

2. Enumeration of solutions using adapting ansatz and Hamiltonians and biasing not previously seen solutions without increasing the size of the ansatz.

3. A method of reducing the size of the ansatz used based on removing quantum gates involving computation on the constraints of the problem whilst preserving sufficient local entanglements for accurate results.

## 8.2 Future Recommendations

Some of the possible future improvements to this work and other research areas are:

1. The most obvious extension of this work is to use this method on sequential logic circuits. This can be done by unrolling the sequential logic circuits to find where and when gate errors occur. Sequential logic along with combinational logic circuits form the basis of all VLSI applications. Without the application of these methods to sequential logic, this will remain an incomplete solution for VLSI debugging.

2. Further optimisation of the ansatz on the quantum hardware will serve to reduce the depth of the quantum circuits and the number of `CNOT` gates used in the problem. This will increase the accuracy of the solutions. Additionally omitting the constraints whose truth-values are already known from the outset will also decrease the width and the depth of the ansatz. This will reduce the required qubits for the problem instance to some sub-graph over which the MIS may be solved.

3. A different representation of the problem circuits may be able to reduce the number of literals in the CNF representation. If there are ways to have fewer total literals because of a more optimal circuit representation or boolean representation then this will allow larger problems to be solved.

4. Application of the methods to other NP-Complete engineering problems may produce notable scaling advantages, this is because not all instances of SAT problems have the same structure and it may be that there are specific instances of SAT where the classical approximate solvers do worse than they do on combinational circuit graphs.

5. Resolution of the single solution QAOA result will reduce the number of times one of the quantum algorithms is required to run to find all instances of the MIS. This will improve the scaling of the algorithm.

# Bibliography


[1] Andreas Veneris, Brian Keng, and Sean Safarpour. "From RTL to silicon: The case for automated debug". In: *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*. ISSN: 2153-697X. Jan. 2011, pp. 306–310. DOI: 10.1109/ASPDAC.2011.5722204 (cit. on p. 1).

[2] R.C. Aitken. "Modeling the unmodelable: algorithmic fault diagnosis". In: *IEEE Design Test of Computers* 14.3 (July 1997), pp. 98–103. ISSN: 1558-1918. DOI: 10.1109/54.606006 (cit. on p. 1).

[3] Ilia Polian et al. "Modeling and Mitigating Transient Errors in Logic Circuits". In: *IEEE Transactions on Dependable and Secure Computing* 8.4 (July 2011), pp. 537–547. ISSN: 1941-0018. DOI: 10.1109/TDSC.2010.26 (cit. on pp. 1, 10).

[4] I. Polian et al. "A Family of Logical Fault Models for Reversible Circuits". In: *14th Asian Test Symposium (ATS'05)*. ISSN: 2377-5386. Dec. 2005, pp. 422–427. DOI: 10.1109/ATS.2005.9 (cit. on p. 1).

[5] Ilia Polian and John P. Hayes. "Advanced modeling of faults in Reversible circuits". In: *2010 East-West Design Test Symposium (EWDTS)*. Sept. 2010, pp. 376–381. DOI: 10.1109/EWDTS.2010.5742135 (cit. on p. 1).

[6] Alexandru Paler, Ilia Polian, and John P. Hayes. "Detection and diagnosis of faulty quantum circuits". In: *17th Asia and South Pacific Design Automation Conference*. ISSN: 2153-697X. Jan. 2012, pp. 181–186. DOI: 10.1109/ASPDAC.2012.6164942 (cit. on p. 1).

[7] Alexandru Paler et al. "Tomographic Testing and Validation of Probabilistic Circuits". In: *2011 Sixteenth IEEE European Test Symposium*. ISSN: 1558-1780. May 2011, pp. 63–68. DOI: 10.1109/ETS.2011.43 (cit. on p. 1).

[8] A. Smith et al. "Fault diagnosis and logic debugging using Boolean satisfiability". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24.10 (Oct. 2005), pp. 1606–1621. ISSN: 1937-4151. DOI: 10.1109/TCAD.2005.852031 (cit. on pp. 2, 10, 28).

[9] Sean Safarpour et al. "Improved Design Debugging Using Maximum Satisfiability". In: *Formal Methods in Computer Aided Design (FMCAD'07)*. Nov. 2007, pp. 13–19. DOI: 10.1109/FAMCAD.2007.26 (cit. on pp. 2, 10, 97).

[10] Yibin Chen et al. "Automated Design Debugging With Maximum Satisfiability". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29.11 (Nov. 2010), pp. 1804–1817. ISSN: 1937-4151. DOI: 10.1109/TCAD.2010.2061270 (cit. on pp. 2, 10).

[11] Richard M Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103 (cit. on pp. 2, 9).

[12] Stephen A. Cook. "The complexity of theorem-proving procedures". In: *Proceedings of the third annual ACM symposium on Theory of computing*. STOC '71. New York, NY, USA: Association for Computing Machinery, May 3, 1971, pp. 151–158. ISBN: 978-1-4503-7464-4. DOI: 10.1145/800157.805047. URL: https://doi.org/10.1145/800157.805047 (visited on 04/07/2022) (cit. on p. 2).

[13] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. Nov. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700 (cit. on p. 2).

[14] David Deutsch and Richard Jozsa. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558 (cit. on p. 2).

[15] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539795293172. URL: http://arxiv.org/abs/quant-ph/9508027 (visited on 04/07/2022) (cit. on p. 2).

[16] Ethan Bernstein and Umesh Vazirani. "Quantum Complexity Theory". In: *SIAM Journal on Computing* 26.5 (Oct. 1997). Publisher: Society for Industrial and Applied Mathematics, pp. 1411–1473. ISSN: 0097-5397. DOI: 10.1137/S0097539796300921. URL: https://epubs.siam.org/doi/10.1137/S0097539796300921 (visited on 04/07/2022) (cit. on p. 2).

[17] R.P. Poplavskii. "Thermodynamic models of information processes". In: *Soviet Physics Uspekhi* 18.3 (1975), p. 222 (cit. on p. 2).

[18] Yuri Manin. "Computable and uncomputable". In: *Sovetskoye Radio, Moscow* 128 (1980) (cit. on p. 2).

[19] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6 (June 1, 1982), pp. 467–488. ISSN: 1572-9575. DOI: 10.1007/BF02650179. URL: https://doi.org/10.1007/BF02650179 (visited on 04/07/2022) (cit. on p. 2).

[20] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 6, 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: http://arxiv.org/abs/1801.00862 (visited on 06/29/2020) (cit. on pp. 2, 19).

[21] Jacob Biamonte et al. "Quantum Machine Learning". In: *Nature* 549.7671 (Sept. 2017), pp. 195–202. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature23474. URL: http://arxiv.org/abs/1611.09347 (visited on 06/24/2020) (cit. on p. 2).

[22] Marcello Benedetti et al. "Parameterized quantum circuits as machine learning models". In: *Quantum Science and Technology* 4.4 (Nov. 13, 2019), p. 043001. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab4eb5. URL: http://arxiv.org/abs/1906.07682 (visited on 07/01/2020) (cit. on p. 2).

[23] Jonas M. Kübler et al. "An Adaptive Optimizer for Measurement-Frugal Variational Algorithms". In: *Quantum* 4 (May 11, 2020), p. 263. ISSN: 2521-327X. DOI: 10.22331/q-2020-05-11-263. URL: http://arxiv.org/abs/1909.09083 (visited on 07/04/2020) (cit. on pp. 2, 19).

[24] Sergey Nurk. "An $O(2^{0.4058m})$ upper bound for Circuit SAT". In: *Steklov Institute of Mathematics at St. Petersburg, Tech. Rep. 10, 2009, PDMI Preprint* (2009) (cit. on p. 9).

[25] Huairui Chu, Mingyu Xiao, and Zhe Zhang. "An Improved Upper Bound for SAT". In: *arXiv:2007.03829 [cs]* (July 7, 2020). URL: http://arxiv.org/abs/2007.03829 (visited on 10/12/2021) (cit. on p. 9).

[26] *Index of /~maksim/benchmarks/iscas85/verilog*. URL: http://www.pld.ttu.ee/~maksim/benchmarks/iscas85/verilog/ (visited on 09/07/2021) (cit. on p. 10).

[27] M.C. Hansen, H. Yalcin, and J.P. Hayes. "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering". In: *IEEE Design Test of Computers* 16.3 (July 1999), pp. 72–80. ISSN: 1558-1918. DOI: 10.1109/54.785838 (cit. on p. 10).

[28] Vinod Maan and Gn Purohit. "A Distributed Approach for Frequency allocation using Graph Coloring in Mobile Networks". In: *International Journal of Computer Applications* 58 (Nov. 2012), pp. 9–13. DOI: 10.5120/9284-3476 (cit. on p. 11).

[29] Xi Li et al. "A Distributed Transmission Scheduling Algorithm for Wireless Networks Based on the Ising Model". In: *2018 IEEE Statistical Signal Processing Workshop (SSP)*. June 2018, pp. 6–10. DOI: 10.1109/SSP.2018.8450715 (cit. on p. 11).

[30] Hao Lu et al. "Algorithms for Balanced Graph Colorings with Applications in Parallel Computing". In: *IEEE Transactions on Parallel and Distributed Systems* 28.5 (May 2017), pp. 1240–1256. ISSN: 1558-2183. DOI: 10.1109/TPDS.2016.2620142 (cit. on p. 11).

[31] Amal Dandashi and Mayez Al-Mouhamed. "Graph Coloring for class scheduling". In: *ACS/IEEE International Conference on Computer Systems and*

*Applications - AICCSA 2010*. ISSN: 2161-5330. May 2010, pp. 1–4. DOI: 10.1109/AICCSA.2010.5586963 (cit. on p. 11).

[32] Omid Ameri Sianaki, Omar Hussain, and Azadeh Rajabian Tabesh. "A Knapsack problem approach for achieving efficient energy consumption in smart grid for endusers' life style". In: *2010 IEEE Conference on Innovative Technologies for an Efficient and Reliable Electricity Supply*. Sept. 2010, pp. 159–164. DOI: 10.1109/CITRES.2010.5619873 (cit. on p. 11).

[33] Yuhong XUE et al. "D2D Resource Allocation and Power Control Algorithms Based on Graph Coloring in 5G IoT". In: *2019 Computing, Communications and IoT Applications (ComComAp)*. Oct. 2019, pp. 17–22. DOI: 10.1109/ComComAp46287.2019.9018806 (cit. on p. 11).

[34] Haibo Wang et al. "Modeling multiple unmanned aerial vehicles placement problem in ad hoc network via quadratic unconstrained binary optimization". In: *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. May 2013, pp. 926–932. DOI: 10.1109/ICUAS.2013.6564778 (cit. on p. 11).

[35] Kotaro Terada et al. "An Ising model mapping to solve rectangle packing problem". In: *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. ISSN: 2472-9124. Apr. 2018, pp. 1–4. DOI: 10.1109/VLSI-DAT.2018.8373233 (cit. on p. 11).

[36] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* 2 (2014). ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005. URL: http://arxiv.org/abs/1302.5843 (visited on 07/04/2020) (cit. on p. 12).

[37] Fred Glover, Gary Kochenberger, and Yu Du. "A Tutorial on Formulating and Using QUBO Models". In: *arXiv:1811.11538 [quant-ph]* (Nov. 4, 2019). URL: http://arxiv.org/abs/1811.11538 (visited on 07/24/2020) (cit. on p. 12).

[38] Ernst Ising. "Beitrag zur Theorie des Ferromagnetismus". In: *Zeitschrift für Physik* 31.1 (Feb. 1, 1925), pp. 253–258. ISSN: 0044-3328. DOI: 10.1007/BF02980577. URL: https://doi.org/10.1007/BF02980577 (visited on 04/07/2022) (cit. on p. 13).

[39] Alexander K. Hartmann and Martin Weigt. *Phase Transitions in Combinatorial Optimization Problems: Basics, Algorithms and Statistical Mechanics*. en.

1st ed. Wiley, July 2005. ISBN: 9783527404735 9783527606733. DOI: 10.1002/3527606734. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/3527606734 (visited on 04/11/2022) (cit. on p. 13).

[40] F Barahona. "On the computational complexity of Ising spin glass models". In: *Journal of Physics A: Mathematical and General* 15.10 (Oct. 1982), pp. 3241–3253. ISSN: 0305-4470, 1361-6447. DOI: 10.1088/0305-4470/15/10/028. URL: https://iopscience.iop.org/article/10.1088/0305-4470/15/10/028 (visited on 04/11/2022) (cit. on p. 13).

[41] Tadashi Kadowaki and Hidetoshi Nishimori. "Quantum annealing in the transverse Ising model". en. In: *Physical Review E* 58.5 (Nov. 1998), pp. 5355–5363. ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.58.5355. URL: https://link.aps.org/doi/10.1103/PhysRevE.58.5355 (visited on 04/11/2022) (cit. on p. 14).

[42] Eleanor G. Rieffel et al. "A case study in programming a quantum annealer for hard operational planning problems". In: *Quantum Information Processing* 14.1 (Jan. 2015), pp. 1–36. ISSN: 1570-0755, 1573-1332. DOI: 10.1007/s11128-014-0892-x. URL: http://arxiv.org/abs/1407.2887 (visited on 07/24/2020) (cit. on p. 17).

[43] Juexiao Su, Tianheng Tu, and Lei He. "A quantum annealing approach for boolean satisfiability problem". In: *Proceedings of the 53rd Annual Design Automation Conference on - DAC '16*. Austin, Texas: ACM Press, 2016, pp. 1–6. ISBN: 978-1-4503-4236-0. DOI: 10.1145/2897937.2897973. URL: http://dl.acm.org/citation.cfm?doid=2897937.2897973 (visited on 07/24/2020) (cit. on p. 17).

[44] Alejandro Perdomo-Ortiz et al. "A Quantum Approach to Diagnosis of Multiple Faults in Electrical Power Systems". In: *2014 IEEE International Conference on Space Mission Challenges for Information Technology*. Sept. 2014, pp. 46–53. DOI: 10.1109/SMC-IT.2014.14 (cit. on p. 17).

[45] Itay Hen and Marcelo S. Sarandy. "Driver Hamiltonians for constrained optimization in quantum annealing". In: *Physical Review A* 93.6 (June 13, 2016), p. 062312. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.93.

062312. URL: http://arxiv.org/abs/1602.07942 (visited on 07/24/2020) (cit. on p. 17).

[46] Nicholas Chancellor et al. "A Direct Mapping of Max k-SAT and High Order Parity Checks to a Chimera Graph". In: *Scientific Reports* 6.1 (Dec. 2016), p. 37107. ISSN: 2045-2322. DOI: 10.1038/srep37107. URL: http://arxiv.org/abs/1604.00651 (visited on 07/23/2020) (cit. on p. 17).

[47] A. Yu Kitaev. "Quantum measurements and the Abelian Stabilizer Problem". In: *arXiv:quant-ph/9511026* (Nov. 20, 1995). URL: http://arxiv.org/abs/quant-ph/9511026 (visited on 04/07/2022) (cit. on p. 18).

[48] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *arXiv:quant-ph/9605043* (Nov. 19, 1996). URL: http://arxiv.org/abs/quant-ph/9605043 (visited on 04/07/2022) (cit. on p. 18).

[49] Charles H. Bennett et al. "Strengths and Weaknesses of Quantum Computing". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1510–1523. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539796300933. URL: http://arxiv.org/abs/quant-ph/9701001 (visited on 04/07/2022) (cit. on p. 18).

[50] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. 10th anniversary ed. Cambridge ; New York: Cambridge University Press, 2010. 676 pp. ISBN: 978-1-107-00217-3 (cit. on pp. 18, 45).

[51] G. Brassard and P. Hoyer. "An exact quantum polynomial-time algorithm for Simon's problem". In: *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*. IEEE Comput. Soc. DOI: 10.1109/istcs.1997.595153. URL: https://doi.org/10.1109%2Fistcs.1997.595153 (cit. on p. 18).

[52] Gilles Brassard et al. "Quantum Amplitude Amplification and Estimation". In: *arXiv:quant-ph/0005055* 305 (2002), pp. 53–74. DOI: 10.1090/conm/305/05215. URL: http://arxiv.org/abs/quant-ph/0005055 (visited on 04/07/2022) (cit. on p. 18).

[53] Andris Ambainis. "Quantum search algorithms". In: *arXiv:quant-ph/0504012* (Apr. 3, 2005). URL: http://arxiv.org/abs/quant-ph/0504012 (visited on 04/07/2022) (cit. on p. 18).

[54] M. Cerezo et al. "Variational Quantum Algorithms". In: *Nature Reviews Physics* 3.9 (Sept. 2021), pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9. URL: http://arxiv.org/abs/2012.09265 (visited on 04/07/2022) (cit. on p. 19).

[55] Kishor Bharti et al. "Noisy intermediate-scale quantum (NISQ) algorithms". In: *Reviews of Modern Physics* 94.1 (Feb. 15, 2022), p. 015004. ISSN: 0034-6861, 1539-0756. DOI: 10.1103/RevModPhys.94.015004. URL: http://arxiv.org/abs/2101.08448 (visited on 04/07/2022) (cit. on p. 19).

[56] Arthur G. Rattew et al. "A Domain-agnostic, Noise-resistant, Hardware-efficient Evolutionary Variational Quantum Eigensolver". In: *arXiv:1910.09694 [quant-ph]* (Jan. 23, 2020). URL: http://arxiv.org/abs/1910.09694 (visited on 07/01/2020) (cit. on p. 19).

[57] Nikolaj Moll et al. "Quantum optimization using variational algorithms on near-term quantum devices". In: *Quantum Science and Technology* 3.3 (July 2018), p. 030503. ISSN: 2058-9565. DOI: 10.1088/2058-9565/aab822. URL: https://iopscience.iop.org/article/10.1088/2058-9565/aab822 (visited on 04/07/2022) (cit. on p. 19).

[58] Harper R. Grimsley et al. "An adaptive variational algorithm for exact molecular simulations on a quantum computer". In: *Nature Communications* 10.1 (Dec. 2019). version: 2, p. 3007. ISSN: 2041-1723. DOI: 10.1038/s41467-019-10988-2. URL: http://arxiv.org/abs/1812.11173 (visited on 06/07/2021) (cit. on p. 19).

[59] Ho Lun Tang et al. "qubit-ADAPT-VQE: An adaptive algorithm for constructing hardware-efficient ansatze on a quantum processor". In: *PRX Quantum* 2.2 (Apr. 28, 2021), p. 020310. ISSN: 2691-3399. DOI: 10.1103/PRXQuantum.2.020310. URL: http://arxiv.org/abs/1911.10205 (visited on 06/07/2021) (cit. on p. 19).

[60] Maria Schuld et al. "Circuit-centric quantum classifiers". In: *Physical Review A* 101.3 (Mar. 6, 2020), p. 032308. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.101.032308. URL: http://arxiv.org/abs/1804.00633 (visited on 07/03/2020) (cit. on p. 19).

[61] Jeremy Adcock et al. "Advances in quantum machine learning". In: *arXiv:1512.02900 [quant-ph]* (Dec. 9, 2015). URL: http://arxiv.org/abs/1512.02900 (visited on 07/01/2020) (cit. on p. 20).

[62] M. Schuld, I. Sinayskiy, and F. Petruccione. "An introduction to quantum machine learning". In: *Contemporary Physics* 56.2 (Apr. 3, 2015), pp. 172–185. ISSN: 0010-7514, 1366-5812. DOI: 10.1080/00107514.2014.964942. URL: http://arxiv.org/abs/1409.3097 (visited on 06/28/2020) (cit. on p. 20).

[63] Farid Ablayev et al. "On quantum methods for machine learning problems part II: Quantum classification algorithms". In: *Big Data Mining and Analytics* 3.1 (Mar. 2020), pp. 56–67. ISSN: 2096-0654. DOI: 10.26599/BDMA.2019.9020018 (cit. on p. 20).

[64] Farid Ablayev et al. "On quantum methods for machine learning problems part I: Quantum tools". In: *Big Data Mining and Analytics* 3.1 (Mar. 2020), pp. 41–55. ISSN: 2096-0654. DOI: 10.26599/BDMA.2019.9020016 (cit. on p. 20).

[65] Istvan Barabasi et al. "Quantum Computing and Deep Learning Working Together to Solve Optimization Problems". In: *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. Dec. 2019, pp. 493–498. DOI: 10.1109/CSCI49370.2019.00095 (cit. on p. 20).

[66] Songtao Shang et al. "A text classification algorithm based on quantum information". In: *2015 11th International Conference on Natural Computation (ICNC)*. ISSN: 2157-9563. Aug. 2015, pp. 381–384. DOI: 10.1109/ICNC.2015.7378020 (cit. on p. 20).

[67] Oscar Galindo et al. "Faster Quantum Alternative to Softmax Selection in Deep Learning and Deep Reinforcement Learning". In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2019, pp. 815–818. DOI: 10.1109/SSCI44817.2019.9003167 (cit. on p. 20).

[68] Nathan Wiebe, Christopher Granade, and David G. Cory. "Quantum Bootstrapping via Compressed Quantum Hamiltonian Learning". In: *New Journal of Physics* 17.2 (Feb. 13, 2015), p. 022005. ISSN: 1367-2630. DOI: 10.1088/1367-

2630/17/2/022005. URL: http://arxiv.org/abs/1409.1524 (visited on 07/01/2020) (cit. on p. 20).

[69] Weng-Long Chang et al. "Quantum speed-up in solving the maximal clique problem". In: *Physical Review A* 97.3 (Mar. 29, 2018), p. 032344. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.97.032344. URL: http://arxiv.org/abs/1803.11356 (visited on 07/30/2020) (cit. on p. 20).

[70] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for solving linear systems of equations". In: *Physical Review Letters* 103.15 (Oct. 7, 2009), p. 150502. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.103.150502. URL: http://arxiv.org/abs/0811.3171 (visited on 06/29/2020) (cit. on p. 21).

[71] Nathan Wiebe, Daniel Braun, and Seth Lloyd. "Quantum Data Fitting". In: *Physical Review Letters* 109.5 (Aug. 2, 2012). version: 2, p. 050505. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.109.050505. URL: http://arxiv.org/abs/1204.5242 (visited on 07/04/2020) (cit. on p. 21).

[72] Xiaosi Xu et al. "Variational algorithms for linear algebra". In: *arXiv:1909.03898 [quant-ph]* (Sept. 9, 2019). URL: http://arxiv.org/abs/1909.03898 (visited on 07/04/2020) (cit. on p. 21).

[73] Hsin-Yuan Huang, Kishor Bharti, and Patrick Rebentrost. "Near-term quantum algorithms for linear systems of equations". In: *arXiv:1909.07344 [quant-ph]* (Dec. 16, 2019). URL: http://arxiv.org/abs/1909.07344 (visited on 07/04/2020) (cit. on p. 21).

[74] Dong An and Lin Lin. "Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm". In: *arXiv:1909.05500 [quant-ph]* (Jan. 9, 2020). URL: http://arxiv.org/abs/1909.05500 (visited on 07/04/2020) (cit. on p. 21).

[75] Carlos Bravo-Prieto et al. "Variational Quantum Linear Solver". In: *arXiv:1909.05820 [quant-ph]* (June 2, 2020). URL: http://arxiv.org/abs/1909.05820 (visited on 07/04/2020) (cit. on p. 21).

[76] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5.1 (July 23, 2014). Number:

1 Publisher: Nature Publishing Group, p. 4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213. URL: https://www.nature.com/articles/ncomms5213 (visited on 04/07/2022) (cit. on p. 21).

[77] Xiao Yuan et al. "Theory of variational quantum simulation". In: *Quantum* 3 (Oct. 7, 2019), p. 191. ISSN: 2521-327X. DOI: 10.22331/q-2019-10-07-191. URL: http://arxiv.org/abs/1812.08767 (visited on 10/12/2021) (cit. on p. 21).

[78] P. Jordan and E. Wigner. "Über das Paulische Äquivalenzverbot". In: *Zeitschrift für Physik* 47.9 (Sept. 1, 1928), pp. 631–651. ISSN: 0044-3328. DOI: 10.1007/BF01331938. URL: https://doi.org/10.1007/BF01331938 (visited on 04/07/2022) (cit. on p. 21).

[79] Sergey B. Bravyi and Alexei Yu. Kitaev. "Fermionic Quantum Computation". In: *Annals of Physics* 298.1 (May 25, 2002), pp. 210–226. ISSN: 0003-4916. DOI: 10.1006/aphy.2002.6254. URL: https://www.sciencedirect.com/science/article/pii/S0003491602962548 (visited on 04/07/2022) (cit. on p. 21).

[80] Sergey Bravyi et al. "Tapering off qubits to simulate fermionic Hamiltonians". In: *arXiv:1701.08213 [quant-ph]* (Jan. 27, 2017). URL: http://arxiv.org/abs/1701.08213 (visited on 04/07/2022) (cit. on p. 21).

[81] Abhinav Kandala et al. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets". In: *Nature* 549.7671 (Sept. 2017). Number: 7671 Publisher: Nature Publishing Group, pp. 242–246. ISSN: 1476-4687. DOI: 10.1038/nature23879. URL: https://www.nature.com/articles/nature23879 (visited on 04/07/2022) (cit. on p. 21).

[82] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A Quantum Approximate Optimization Algorithm". In: *arXiv:1411.4028 [quant-ph]* (Nov. 14, 2014). URL: http://arxiv.org/abs/1411.4028 (visited on 06/29/2020) (cit. on pp. 22, 23, 43).

[83] Boaz Barak et al. "Beating the random assignment on constraint satisfaction problems of bounded degree". In: *arXiv:1505.03424 [cs]* (Aug. 11, 2015). URL: http://arxiv.org/abs/1505.03424 (visited on 04/07/2022) (cit. on p. 22).

[84] Edward Farhi and Aram W. Harrow. "Quantum Supremacy through the Quantum Approximate Optimization Algorithm". In: *arXiv:1602.07674 [quant-ph]* (Oct. 20, 2019). URL: http://arxiv.org/abs/1602.07674 (visited on 04/07/2022) (cit. on p. 22).

[85] Stuart Hadfield et al. "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz". In: *Algorithms* 12.2 (Feb. 12, 2019), p. 34. ISSN: 1999-4893. DOI: 10.3390/a12020034. URL: http://arxiv.org/abs/1709.03489 (visited on 07/23/2020) (cit. on pp. 22, 24, 36, 39, 94).

[86] Stuart Hadfield et al. "Quantum Approximate Optimization with Hard and Soft Constraints". In: *Proceedings of the Second International Workshop on Post Moores Era Supercomputing - PMES'17*. Denver, CO, USA: ACM Press, 2017, pp. 15–21. ISBN: 978-1-4503-5126-3. DOI: 10.1145/3149526.3149530. URL: http://dl.acm.org/citation.cfm?doid=3149526.3149530 (visited on 07/24/2020) (cit. on pp. 22, 24, 39).

[87] Leo Zhou et al. "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices". In: *Physical Review X* 10.2 (June 24, 2020), p. 021067. ISSN: 2160-3308. DOI: 10.1103/PhysRevX.10.021067. URL: http://arxiv.org/abs/1812.01041 (visited on 07/23/2020) (cit. on p. 22).

[88] Zain H. Saleem. "Max Independent Set and Quantum Alternating Operator Ansatz". In: *arXiv:1905.04809 [quant-ph]* (Apr. 9, 2020). URL: http://arxiv.org/abs/1905.04809 (visited on 07/30/2020) (cit. on pp. 22, 39).

[89] Ajinkya Borle, Vincent E. Elfving, and Samuel J. Lomonaco. "Quantum Approximate Optimization for Hard Problems in Linear Algebra". In: *arXiv:2006.15438 [quant-ph]* (June 27, 2020). URL: http://arxiv.org/abs/2006.15438 (visited on 07/04/2020) (cit. on p. 22).

[90] Jeremy Cook, Stephan Eidenbenz, and Andreas Bärtschi. "The Quantum Alternating Operator Ansatz on Max-k Vertex Cover". In: *arXiv:1910.13483 [quant-ph]* (Oct. 29, 2019). URL: http://arxiv.org/abs/1910.13483 (visited on 07/24/2020) (cit. on p. 22).

[91] Ryan Sweke et al. "Stochastic gradient descent for hybrid quantum-classical optimization". In: *arXiv:1910.01155 [quant-ph]* (Oct. 2, 2019). URL: http://arxiv.org/abs/1910.01155 (visited on 07/03/2020) (cit. on p. 23).

[92] Jarrod R. McClean et al. "The theory of variational hybrid quantum-classical algorithms". In: *New Journal of Physics* 18.2 (Feb. 4, 2016), p. 023023. ISSN: 1367-2630. DOI: 10.1088/1367-2630/18/2/023023. URL: http://arxiv.org/abs/1509.04279 (visited on 06/29/2020) (cit. on p. 23).

[93] Gavin E. Crooks. "Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition". In: *arXiv:1905.13311 [quant-ph]* (May 30, 2019). URL: http://arxiv.org/abs/1905.13311 (visited on 07/11/2020) (cit. on p. 23).

[94] James D. Whitfield, Jacob Biamonte, and Alán Aspuru-Guzik. "Simulation of Electronic Structure Hamiltonians Using Quantum Computers". In: *Molecular Physics* 109.5 (Mar. 10, 2011), pp. 735–750. ISSN: 0026-8976, 1362-3028. DOI: 10.1080/00268976.2011.552441. URL: http://arxiv.org/abs/1001.3855 (visited on 07/10/2020) (cit. on p. 23).

[95] Guillaume Verdon, Michael Broughton, and Jacob Biamonte. "A quantum algorithm to train neural networks using low-depth circuits". In: *arXiv:1712.05304 [cond-mat, physics:quant-ph]* (Aug. 10, 2019). URL: http://arxiv.org/abs/1712.05304 (visited on 07/05/2020) (cit. on p. 23).

[96] Joel E. Cohen et al. "Eigenvalue inequalities for products of matrix exponentials". In: *Linear Algebra and its Applications* 45 (June 1, 1982), pp. 55–95. ISSN: 0024-3795. DOI: 10.1016/0024-3795(82)90211-7. URL: https://www.sciencedirect.com/science/article/pii/0024379582902117 (visited on 04/07/2022) (cit. on p. 23).

[97] Naomichi Hatano and Masuo Suzuki. "Finding Exponential Product Formulas of Higher Orders". In: *arXiv:math-ph/0506007* 679 (Nov. 16, 2005), pp. 37–68. DOI: 10.1007/11526216_2. URL: http://arxiv.org/abs/math-ph/0506007 (visited on 04/07/2022) (cit. on p. 23).

[98] Stuart Hadfield. "Quantum Algorithms for Scientific Computing and Approximate Optimization". In: *arXiv:1805.03265 [quant-ph]* (May 8, 2018). URL:

http://arxiv.org/abs/1805.03265 (visited on 07/24/2020) (cit. on pp. 24, 39, 41).

[99] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. "PySAT: A Python Toolkit for Prototyping with SAT Oracles". In: *SAT*. 2018, pp. 428–437. DOI: 10.1007/978-3-319-94144-8_26. URL: https://doi.org/10.1007/978-3-319-94144-8_26 (cit. on p. 31).

[100] Grigori S Tseitin. "On the complexity of derivation in propositional calculus". In: *Automation of reasoning*. Springer, 1983, pp. 466–483 (cit. on p. 35).

[101] Patrick J Hurley. *A concise introduction to logic*. Nelson Education, 2014 (cit. on p. 35).

[102] Dasgupta Sanjoy, Papadimitriou Christos, and Vazirani Umesh. *Algorithms*. McGraw-Hill Boston, 2006 (cit. on p. 36).

[103] *Basic tutorial: qubit rotation — PennyLane*. URL: https://pennylane.ai/qml/demos/tutorial_qubit_rotation.html (visited on 10/12/2021) (cit. on p. 38).

[104] Hannes Pichler et al. "Quantum Optimization for Maximum Independent Set Using Rydberg Atom Arrays". In: *arXiv:1808.10816 [cond-mat, physics:physics, physics:quant-ph]* (Aug. 31, 2018). URL: http://arxiv.org/abs/1808.10816 (visited on 07/23/2020) (cit. on p. 39).

[105] Jaeho Choi, Seunghyeok Oh, and Joongheon Kim. "The Useful Quantum Computing Techniques for Artificial Intelligence Engineers". In: *2020 International Conference on Information Networking (ICOIN)*. ISSN: 1976-7684. Jan. 2020, pp. 1–3. DOI: 10.1109/ICOIN48656.2020.9016555 (cit. on pp. 39, 94).

[106] Jaeho Choi and Joongheon Kim. "A Tutorial on Quantum Approximate Optimization Algorithm (QAOA): Fundamentals and Applications". In: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. ISSN: 2162-1233. Oct. 2019, pp. 138–142. DOI: 10.1109/ICTC46691.2019.8939749 (cit. on pp. 39, 41).

[107] J. D. Whitfield, M. Faccin, and J. D. Biamonte. "Ground State Spin Logic". In: *EPL (Europhysics Letters)* 99.5 (Sept. 1, 2012), p. 57004. ISSN: 0295-5075,

1286-4854. DOI: 10.1209/0295-5075/99/57004. URL: http://arxiv.org/abs/1205.1742 (visited on 07/30/2020) (cit. on p. 41).

[108] Stuart Hadfield. "On the representation of Boolean and real functions as Hamiltonians for quantum computing". In: *arXiv:1804.09130 [quant-ph]* (Apr. 24, 2018). URL: http://arxiv.org/abs/1804.09130 (visited on 07/24/2020) (cit. on p. 41).

[109] Sukin Sim, Peter D. Johnson, and Alan Aspuru-Guzik. "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms". In: *Advanced Quantum Technologies* 2.12 (Dec. 2019), p. 1900070. ISSN: 2511-9044, 2511-9044. DOI: 10.1002/qute.201900070. URL: http://arxiv.org/abs/1905.10876 (visited on 07/19/2021) (cit. on p. 47).

[110] Rebekah Herrman et al. "Impact of Graph Structures for QAOA on MaxCut". In: *arXiv:2102.05997 [quant-ph]* (Feb. 11, 2021). URL: http://arxiv.org/abs/2102.05997 (visited on 10/12/2021) (cit. on p. 51).

[111] Bochen Tan and Jason Cong. "Optimal Layout Synthesis for Quantum Computing". In: *arXiv:2007.15671 [quant-ph]* (July 30, 2020). URL: http://arxiv.org/abs/2007.15671 (visited on 10/12/2021) (cit. on p. 65).

[112] Jarrod R. McClean et al. "Barren plateaus in quantum neural network training landscapes". In: *Nature Communications* 9.1 (Nov. 16, 2018). Number: 1 Publisher: Nature Publishing Group, p. 4812. ISSN: 2041-1723. DOI: 10.1038/s41467-018-07090-4. URL: https://www.nature.com/articles/s41467-018-07090-4 (visited on 04/07/2022) (cit. on p. 71).

[113] J.C. Spall. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". In: *IEEE Transactions on Automatic Control* 37.3 (Mar. 1992), pp. 332–341. ISSN: 1558-2523. DOI: 10.1109/9.119632 (cit. on p. 71).

[114] Kishor Bharti et al. "Noisy intermediate-scale quantum (NISQ) algorithms". In: *arXiv:2101.08448 [cond-mat, physics:quant-ph]* (Oct. 6, 2021). URL: http://arxiv.org/abs/2101.08448 (visited on 10/12/2021) (cit. on p. 71).

[115] James Stokes et al. "Quantum Natural Gradient". In: *Quantum* 4 (May 25, 2020), p. 269. ISSN: 2521-327X. DOI: 10.22331/q-2020-05-25-269. URL: http://arxiv.org/abs/1909.02108 (visited on 07/04/2020) (cit. on p. 72).

[116] Naoki Yamamoto. "On the natural gradient for variational quantum eigensolver". In: *arXiv:1909.05074 [quant-ph]* (Sept. 11, 2019). URL: http://arxiv.org/abs/1909.05074 (visited on 07/04/2020) (cit. on p. 72).

[117] Julien Gacon et al. "Simultaneous Perturbation Stochastic Approximation of the Quantum Fisher Information". In: *arXiv:2103.09232 [quant-ph]* (Mar. 15, 2021). URL: http://arxiv.org/abs/2103.09232 (visited on 09/20/2021) (cit. on p. 72).

[118] John W Moon and Leo Moser. "On cliques in graphs". In: *Israel journal of Mathematics* 3.1 (1965), pp. 23–28 (cit. on p. 92).

[119] David R. Wood. "On the number of maximal independent sets in a graph". In: *arXiv:1104.1243 [cs, math]* (Apr. 14, 2011). URL: http://arxiv.org/abs/1104.1243 (visited on 09/30/2021) (cit. on p. 92).

[120] Joao Marques-Silva et al. "On computing minimal correction subsets". In: *Twenty-Third International Joint Conference on Artificial Intelligence*. Citeseer. 2013 (cit. on p. 92).

[121] Carlos Mencía, Alessandro Previti, and Joao Marques-Silva. "Literal-based MCS extraction". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015 (cit. on p. 92).

[122] Christof Zalka. "Grover's quantum searching algorithm is optimal". In: *Physical Review A* 60.4 (Oct. 1, 1999). version: 2, pp. 2746–2751. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.60.2746. URL: http://arxiv.org/abs/quant-ph/9711070 (visited on 10/12/2021) (cit. on p. 95).

[123] Ville Bergholm et al. "PennyLane: Automatic differentiation of hybrid quantum-classical computations". In: *arXiv:1811.04968 [physics, physics:quant-ph]* (Feb. 13, 2020). URL: http://arxiv.org/abs/1811.04968 (visited on 07/01/2020) (cit. on p. 98).

# Appendix A

# Source Code

This appendix contains the source code of the modules developed for implementation of the various algorithms. The Python modules are not specific to this work and may be used to implement the quantum algorithms for many Nondeterministic Polynomial Time Complete (NP-Complete) problems and with a little adjustment all of them. It also includes the modules for the circuit debugging problem that may be used with any classical solvers and is used to set up and transform the problem for the quantum solutions. The scripts that were used to run the specific trials of the algorithms in this dissertation can be found on the project git pages: qcircsat and circsat.

## A.1 Programming Approach

The following appendices A.1.1 to A.1.8 include the key code along with explanations from the modules created for the methods discussed in chapter 4 and application discussed in chapter 5.

### A.1.1 Problem Generation

Problems are generated using the Circuit class in the circsat module. The problems are saved as files in the WDIMACS format. The appropriate circuit specification is given by a truth table of IO values of the circuit. The circuit is built using gates from the Circuit class. However multiple Circuits are built with combinations of all the possible gate errors. Not every possible replacement is made and the replacements of the expected gate with the improper gate is always made with the inverse of the gate. That means that a `not` gate would be replaced by a `buffer` and a `nand` gate by an `and` gate and vice versa. Extension to every type of incorrect replacement can be made in future.

It is recommended that the problem generation be a class or module in future iterations of the software. The IO of the truth table along with the circuits themselves are used to create a circuit object. The properties of the circuit are then used to set the appropriate constraints and weights (including the top weight) for the WCNF file that will be named according to the error and which line of the truth table the constraints were generated from.

### A.1.2 Cirsat Module

Takes as an argument the problem folder. It enumerates through the MCSC or MLCSC for each of the problems in the problem folder. After enumeration, the clauses are converted back to the gates they represent. This occurs whilst useful information such as the oracle time is recorded to file.

### A.1.3 Preprocessing

Listing 3 contains the source code to be discussed. It is worth noting the two classes discussed below are named according to what type of structure they process. They are therefore not the structures they are called by but operators acting on the objects they are named after. In retrospect, this is probably a bit confusing but the classes act functionally and the *some object to some other object* names are unwieldy.

The Circuit classes primary function is to transform a netlist of gates along with their Input-Output (IO) labels into an object that can be used by the PySAT module to create a WCNF file describing the PMAX-SAT problem. As such it accepts Input-Output (IO) labels for the circuit along with a dictionary of gates that are statically created by the Circuit class itself.

The Tseitin class transforms the WCNF into a graph with vertices and edges such that some optimisation problem can be solved on the resultant structure. There is a crucial assumption that needs to be followed in the transformation: the CNF is labelled with numbers that represent a variable or literal in a Boolean representation. Written by hand one might appropriately use letters to represent variables with subscripts to label and distinguish them such as $(\neg i_0^0 \vee \neg o_0^5) \wedge (i_0^1 \vee o_0^6)$ and symbols to represent negation and conjunction and other Boolean operations. In the code this is represented by $[[-1, -2], [1, 2]]$.

However, the vertices are labelled with whole numbers. The vertex list is therefore created by $\{-1, -2, 1, 2\} \mapsto \{0, 1, 2, 3\}$. Note that if a literal symbol is repeated somewhere in the CNF a new vertex label represents that literal and not the same vertex label. This creates a one to two mapping from CNF to $G(V, E)$ and creates a challenge for the eventual inverse transforms that are to be computed to find a solution in the original encoding later in the problem resolution.

Notice because vertices are just encoded by whole numbers they are always positive and therefore positive numbers represent negations of literals. This means that even though a node that is coloured means that the node is "on" in the graph encoding it may mean that the literal is "off" for the circuit. This means that the constraints are encoded in the graph by colouring IO whether or not they are negations. A brief setting of all neighbours of on nodes to off can be done as a last classical preprocessing step in this transformation.

Listing 3: Implementation of the Python preprocess module. This module includes the Circuit and Tseitin classes. This implements the transformations discussed in section 4.2.

```
from functools import reduce
from itertools import count

class Circuit:
    '''
    Transforms a netlist of gates along with their IO labels into
    an object that can be used by the pysat module to create a WCNF
    file describing the PMAX-SAT problem. As such it accepts IO
    labels for the circuit along with a dictionary of gates that
    are statically created by the Circuit class itself.
    '''

    def __init__(self, inputs, outputs, **kwargs):
        self.inputs = inputs
        self.outputs = outputs
        self.cct_cnf = reduce(lambda g1,g2: g1+g2, kwargs.values())

    @staticmethod
    def NOT(i, o):
        return [[-i, -o], [i, o]]

    @staticmethod
    def BUFFER(i, o):
        return [[-i, o], [i, -o]]

    @staticmethod
    def AND(i0, i1, o):
        return [[-i0, -i1, o], [i0, -o], [i1, -o]]

    @staticmethod
    def NAND(i0, i1, o):
        return [[-i0, -i1, -o], [i0, o], [i1, o]]

    @staticmethod
    def OR(i0, i1, o):
        return [[i0, i1, -o], [-i0, o], [-i1, o]]
```

```
37
38      @staticmethod
39      def NOR(i0, i1, o):
40          return [[i0, i1, o], [-i0, -o], [-i1, -o]]
41
42      @staticmethod
43      def XOR(i0, i1, o):
44          return [[-i0, -i1, -o], [i0, i1, -o],
45                  [i0, -i1, o], [-i0, i1, o]]
46
47      @staticmethod
48      def XNOR(i0, i1, o):
49          return [[-i0, -i1, o], [i0, i1, o],
50                  [i0, -i1, -o], [-i0, i1, -o]]
51
52      @property
53      def constraints(self):
54          return [[l] for l in self.inputs + self.outputs]
55
56      @property
57      def weights(self):
58          return [1 for _ in range(len(self.cct_cnf))]
59
60      @property
61      def top_weight(self):
62          return 1 + len(self.cct_cnf)
63
64  from itertools import combinations, chain
65  import networkx as nx
66
67  class Tseitin():
68      '''
69      Transforms WCNF into a graph with vertices and edges
70      such that some optimisation problem can be solved on
71      the resultant structure.
72      '''
```

```
73      def __init__(self, cct_cnf):
74          self.constraints = cct_cnf.hard
75          self.cct_cnf = cct_cnf.soft
76          self.cs = int(cct_cnf.comments[0][-1])
77          self.inputs = self.flatten_iter(self.constraints[:self.cs])
78          self.outputs = self.flatten_iter(self.constraints[self.cs:])
79          self.edges = self.edge_combinations + self.consistent_assignments
80          self.graph = nx.Graph(self.edges)
81
82      def __label_vertices(self):
83          label = 0
84          lv = list()
85          for clause in self.cct_cnf:
86              clauses = list()
87              lv.append(clauses)
88              for literal in clause:
89                  clauses.append(label)
90                  label += 1
91          return lv
92
93      def flatten_iter(self, itr):
94          return list(chain(*itr))
95
96      def remove_duplicates(self, itr):
97          return tuple(set(self.flatten_iter(itr)))
98
99      @property
100     def vertex_enc(self):
101         return {v:c for v,c in enumerate(
102             self.flatten_iter(self.cct_cnf))}
103
104     def constraint_enc(self, cnf_io):
105         graph_io = list()
106         for l in cnf_io:
107             for k,v in self.vertex_enc.items():
108                 if l == v:
```

```
109                     graph_io.append(k)
110         return graph_io
111
112     def neighbours(self, nodes):
113         return [list(self.graph.neighbors(n)) for n in nodes]
114
115     @property
116     def graph_on_off(self):
117         cct_constraints = self.inputs + self.outputs
118         on = self.constraint_enc(cct_constraints)
119         off_dup = self.neighbours(on)
120         off = self.remove_duplicates(off_dup)
121         overlap = set(on).intersection(set(off))
122         on = list(set(on) - overlap)
123         off = list(set(off) - overlap)
124         overlap = list(overlap)
125         return on, off, overlap
126
127     @property
128     def edge_combinations(self):
129         '''
130         Edges from adjacent vertices forces at least one
131         literal in each clause to be true
132         '''
133         return self.flatten_iter([list(combinations(clause, 2))
134                 for clause in self.__label_vertices()])
135
136     @property
137     def consistent_assignments(self):
138         '''
139         Connecting edges between all literals and their
140         negations forces consistent truth assignments
141         '''
142         negations = list()
143         for ki, ko in combinations(self.vertex_enc, 2):
144             if self.vertex_enc[ki] == -1*self.vertex_enc[ko]:
```

```
145                     negations.append((ki, ko))
146             return negations
```

### A.1.4 Postprocessing

The postprocessing in listing 4 is required to convert both the MCSCs back to MCSGs errors and to convert graph solutions back to MCSCs and finally MCSGs errors. Similarly to the previously discussed modules the classes are named after objects that they take as input in their transformations.

The class Clauses allows calculation of MCSGs from MCSCs and a CNF Circuit. It simply checks whether a clause appears in a certain gate. If the clause does appear in the gate then the gate is a candidate for an erroneous implementation. Because multiple clauses may have remained unsatisfied it is common for a gate to have more than one unsatisfied clause. This means that a gate may be listed as an error candidate more than once. It is not clear if this does make that specific gate more likely to be a source of error than the other gates that may be part of the solution but it is thought that it would increase the probability since every clause violated corresponds to a part in the truth table of the gate that is violated.

The Graph class converts a graph solution into MCSCs and MCSGs solutions given a graph solution, CNF Circuit and a vertex to clause mapping. The vertex to clause mapping is obtained from the Tseitin class. This class inherits from the Clauses class to allow it to convert clause solutions into gate solutions. This is used for problems that output graph solutions instead of clause based solutions directly like general SAT solver libraries.

Listing 4: Implementation of the Python postprocess module. This module includes the Clauses and Graph classes. Section 4.4 contains discussion on the methods used in this implementation.

```
class Clauses:
    '''
    Allows calculation of MCGS from MCSC
    a CNF Circuit.
    '''

    def __init__(self, cct, clauses):
        self.cct = cct
        self.clauses = clauses

    def minimal_corrected_gates(self):
        mcsgs = list()
        for k,v in self.cct.items():
            for mcsc in self.clauses:
                if mcsc in v:
                    mcsgs.append(k)
        return mcsgs

class Graph(Clauses):
    '''
    Converts a Graph solution into MCSC and MCGS
    solutions given a graph solution, CNF Circuit
    and a vertex to clause mapping.
    '''

    def __init__(self, graph_sols, cct_cnf,
            cct, vertex_enc):
        self.g_sols = graph_sols
        self.cnf = cct_cnf
        self.cct = cct
        self.ve = vertex_enc
        super().__init__(self.cct,
                self.literals_to_mcscs)

    @property
    def vertex_to_literals(self):
```

```
37          lit_sols = list()
38          for g_sol in self.g_sols:
39              lit_sol = list()
40              lit_sols.append(lit_sol)
41              for lit in g_sol:
42                  lit_sol.append(self.ve[lit])
43          return lit_sols
44
45      @property
46      def literals_to_mcscs(self):
47          mcscs = list()
48          for lit_sols in self.vertex_to_literals:
49              mcsc = self.cnf.copy()
50              for lit_sol in lit_sols:
51                  for clause in mcsc:
52                      if lit_sol in clause:
53                          mcsc.remove(clause)
54              mcscs += mcsc
55          return mcscs
```

### A.1.5 Qcircsat Module

This is the main module that aims to solve the circuit debugging problem using the methodology of converting the problem into a graph structure and using quantum or brute force classical algorithms to optimise for the correct MIS solution.

The general approach is to load the circuits and WDIMACS files for a given problem specified as a command-line argument. For each of these problems, a graph is constructed along with its on and off nodes. The relevant Hamiltonians and Ansatz are also constructed over which the Ansatz is optimised or in the case of the brute force solution an exact eigensolver that uses the Hamiltonians instead. After the solution is extracted there is classical postprocessing that gets relevant data from the solution to compare accuracy and time taken for the solution, including saving data

that describes the set-up and construction of the solvers.

Since there may be multiple MIS solutions for each problem this process is repeated until there are no more MIS left. This is done by recreating the Hamiltonians to bias terms not previously seen in the solution. Once there are no more terms left to bias or the solutions produced are no longer MIS or the solver is stuck on the same solution (this usually happens in the case that the solver is not sufficiently accurate for the problem, but it is required since for larger problems this may happen frequently and the solver ought to continue in case it fairs better for the remaining runs). Finally, the data is saved to file or if an image is required to tex source code.

### A.1.6 Qtools

The source code to be discussed is contained in listing 5. The class Qtools defines common mathematical structures used for quantum circuits and for Hamiltonians such as specifying a Pauli operator on a specific wire.

The class Postprocess handles the plotting and processing of data returned by the various solutions to make the results of the runs clearer. Additionally, it writes common data and specified extended data to file.

The class Error calculates the various expectations given solutions over Hamiltonians as well as factors indicating the quality of the MIS solution in terms of the result.

Listing 5: Implementation of the Python qtools module. This module includes the Qtools, Postprocess and Error classes. This contains many helper functions to the creation of Hamiltonians and writing data to file. The Error class implementation follows section 6.5.2.

```
1 import datetime
2 from functools import reduce
3 from qiskit.opflow import I
4
5 class Qtools:
6     '''
```

```
7       Defines common mathematical structures used
8       for quantum circuits and for Hamiltonians.
9       '''
10
11      def __init__(self):
12          self.counts = list()
13          self.means = list()
14          self.stds = list()
15          self.params = list()
16
17      def cb(self, eval_count, parameters, mean, std):
18          print(f'{datetime.datetime.now()}: Circuit evalution {eval_count} '
19                  f'with parameters {parameters} has mean energy {mean} '
20                  f'with deviation {std}')
21          self.counts.append(eval_count)
22          self.means.append(mean)
23          self.stds.append(std)
24          self.params.append(parameters.tolist())
25
26      @staticmethod
27      def tensor_iter(iter_ops):
28          return reduce(lambda A,B: A^B, iter_ops)
29
30      @staticmethod
31      def compose_iter(iter_ops):
32          return reduce(lambda A,B: A@B, iter_ops)
33
34      @classmethod
35      def pauli_at(cls, op, at, nodes):
36          return cls.tensor_iter((I if at != n else op for n in sorted(nodes)))
37
38      @staticmethod
39      def hamiltonian(coeffs, ops):
40          return reduce(lambda i,j: i + j, (c*o
41              for c,o in zip(coeffs, ops)))
42
```

```
43 import networkx as nx
44 import numpy as np
45 import json
46 import matplotlib.pyplot as plt
47
48 class Postprocess:
49     '''
50     Handles the plotting and processing of data
51     returned by the various solutions to make
52     the results of the runs clearer. Additionally
53     writes common data and specified extended data
54     to file.
55     '''
56
57     def __init__(self, result, graph):
58         if isinstance(result, dict):
59             self.result = {
60                     'optimal_point': result['optimal_point'].tolist(),
61                     'eigenvalue': result['eigenvalue'].real,
62                     'optimiser_evals': result['optimizer_evals'],
63                     'cost_function_evals': result['cost_function_evals'],
64                     'optimiser_time': result['optimizer_time'],
65                     'eigenstate': result['eigenstate']
66             }
67         else: # note that this breaks the numpy solver since it doesn't have an
          ↪  optimal point
68             # require seperate classes that inherit from a base class is the way
               ↪  to go
69             self.result = {
70                     'optimal_point': result.optimal_point.tolist(),
71                     'eigenvalue': result.eigenvalue.real,
72                     'optimiser_evals': result.optimizer_evals,
73                     'cost_function_evals': result.cost_function_evals,
74                     'optimiser_time': result.optimizer_time,
75                     'eigenstate': result.eigenstate
76             }
```

```
77          self.graph = graph
78
79      def write_result(self, filename, *argv):
80          for arg in argv:
81              self.result.update(arg)
82          with open(f'{filename}.json', 'w') as _:
83              json.dump(self.result, _, indent=4)
84
85      def write_graph(self, filename):
86          nx.write_edgelist(self.graph, f'{filename}.txt')
87
88      @property
89      def run_time(self):
90          return self.result['optimiser_time']
91
92      @property
93      def energy(self):
94          return self.result['eigenvalue']
95
96      @property
97      def likely_states(self):
98          d = self.result['eigenstate']
99          d_sorted = {k: v for k, v in sorted(d.items(), key=lambda i: i[1])}
100         return {k: d_sorted[k] for k in [*d_sorted][-9:]}
101
102     @property
103     def most_likely(self):
104         d = self.result['eigenstate']
105         if not isinstance(d, dict):
106             return [*d.sample(1)][0]
107         else:
108             return max(d, key=d.get)
109
110     @staticmethod
111     def bitstrings(graph, coloured_node_hist):
112         zero_state = '0'*len(graph)
```

```
113         bitstring_hist = list()
114         for sol in coloured_node_hist:
115             bitstring = ''
116             for i, s in enumerate(zero_state):
117                 if i in sol: bitstring += '1'
118                 else: bitstring += '0'
119             bitstring_hist.append(bitstring)
120         return bitstring_hist
121 
122     @staticmethod
123     def coloured_nodes(bitstring):
124         return [i for i,b in enumerate([int(_) for _ in bitstring]) if b]
125 
126     def colour_map(self, bitstring):
127         return ['g' if node in self.coloured_nodes(bitstring)
128                     else 'c' for node in self.graph.nodes]
129 
130     def __draw_graph(self, cmap, ax):
131         nx.draw_networkx(self.graph, node_color=cmap, node_size=200,
132                 alpha=.8, ax=ax, pos=nx.circular_layout(self.graph))
133 
134     def draw_graph(self):
135         k = self.most_likely
136         cmap = self.colour_map(k)
137         nx.draw_networkx(self.graph, node_color=cmap, node_size=200,
138                 alpha=.8, pos=nx.circular_layout(self.graph))
139 
140     def draw_graphs(self):
141         size = int(np.ceil(np.sqrt(len(self.likely_states))))
142         fig, axs = plt.subplots(size, size, figsize=(18, size*4), sharex=True,
        ↪  sharey=True)
143         if isinstance(axs, np.ndarray):
144             for ax, k in zip(axs.flatten(), self.likely_states.keys()):
145                 cmap = self.colour_map(k)
146                 ax.set_title(f'Probability:
                ↪  {np.abs(self.likely_states[k])**2}')
```

```
147                 self.__draw_graph(cmap, ax)
148         else: self.draw_graph()
149 
150 from qiskit.opflow.state_fns import StateFn, CircuitStateFn
151 from qiskit.opflow.expectations import AerPauliExpectation
152 from qiskit.opflow.converters import CircuitSampler
153 from qiskit.utils import QuantumInstance
154 from qiskit import QuantumCircuit, Aer
155 
156 class Error():
157     '''
158     Calculates the various expectations given solutions
159     over Hamiltonians as well as factors indicating the
160     quality of the MIS solution in terms of the result.
161     '''
162 
163     @staticmethod
164     def expectation_sampler(hamiltonian, bitstring):
165         '''
166         The following routine for evaluating expectations of operators in qiskit
    ↪  is taken from:
167         https://quantumcomputing.stackexchange.com/questions/12080
168         '''
169         psi = QuantumCircuit(len(bitstring))
170         for i,_ in enumerate(reversed(bitstring)):
171             if int(_): psi.x(i)
172 
173         psi = CircuitStateFn(psi)
174 
175         backend = Aer.get_backend('aer_simulator')
176         instance = QuantumInstance(backend, shots=1)
177 
178         measurable = StateFn(hamiltonian, is_measurement=True)@psi
179         expectation = AerPauliExpectation().convert(measurable)
180         sampler = CircuitSampler(backend).convert(expectation)
181         return sampler.eval().real
```

```
182
183     @staticmethod
184     def mis_obj(bitstring):
185         return sum([int(v) for v in bitstring])
186
187     @staticmethod
188     def max_factor(exact_obj, approx_obj):
189         return 1 - abs(exact_obj
190                 - approx_obj)/exact_obj
191
192     @staticmethod
193     def ind_factor(graph, coloured_nodes):
194         adherence = 0
195         num_nodes = len(graph.nodes)
196         for v in graph.nodes:
197             n = set(graph.neighbors(v))
198             s = set(coloured_nodes)
199             if v not in coloured_nodes:
200                 adherence += 1
201             elif not n.intersection(s):
202                 adherence += 1
203         return adherence/num_nodes
204
205     @staticmethod
206     def mis_factor(ind_f, max_f):
207         return ind_f*max_f
```

### A.1.7 Drivers

The drivers class adapted from Pennylane and given in listing 6 is used to return Qiskit compatible Hamiltonians. It creates cost drivers and allows specification whether coloured or uncoloured nodes in a graph are assigned higher or lower energies. It also allows the specification of whether adjacent nodes that are coloured or uncoloured

should be assigned higher or lower energies. These Hamiltonians are usually used with others to fully specify an optimisation problem and are versatile enough to cover all the major graph optimisation problems commonly solved in Quantum Computing.

Listing 6: Implementation of the Python drivers module. This module contains the Drivers class. This implements the Hamiltonian drivers from section 4.3.2 and section 4.3.4.

```
# Copyright 2018-2020 Xanadu Quantum Technologies Inc.
# Modification Copyright 2020-2021 Peter Demetriou
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
    # http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import networkx as nx
from qiskit.opflow import I, X, Z
from qcircsat.qtools import Qtools

class Drivers:
    '''
    Adapted from Pennylane to return Qiskit compatible Hamiltonians.
    Creates cost drivers, allows specification whether coloured or
    uncoloured nodes in a graph are assigned higher or lower energies.
    Also allows specification whether adjacent nodes that are coloured
    or uncoloured should be assigned higher or lower energies. These
    Hamiltonians are usually used with others in order to fully specify
    an optimisation problem and are versatile enough to cover all the
    major graph optimisation problems commonly solved in Quantum Computing.
```

```
30      '''
31
32      @staticmethod
33      def bit_driver(graph, b):
34          nodes = graph.nodes
35          coeffs = (1 if b == 1 else -1 for _ in nodes)
36          ops = (Qtools.pauli_at(Z, at, nodes) for at in graph)
37          return Qtools.hamiltonian(coeffs, ops)
38
39      @staticmethod
40      def edge_driver(graph, reward):
41          nodes = graph.nodes
42          edges = graph.edges
43          coeffs = list()
44          ops = list()
45          if len(reward) == 0 or len(reward) == 4:
46              coeffs = [1 for _ in nodes]
47              ops = [I for v in nodes]
48          else:
49              reward = list(set(reward) - {'01'})
50              sign = -1
51              if len(reward) == 2:
52                  reward = list({'00', '10', '11'} - set(reward))
53                  sign = 1
54              reward = reward[0]
55              if reward == '00':
56                  for e in edges:
57                      coeffs.extend([0.25 * sign, 0.25 * sign, 0.25 * sign])
58                      ops.extend([Qtools.pauli_at(Z, e[0],
                        ↪ nodes)@Qtools.pauli_at(Z, e[1], nodes),
59                          Qtools.pauli_at(Z, e[0], nodes), Qtools.pauli_at(Z, e[1],
                            ↪ nodes)])
60              if reward == '10':
61                  for e in edges:
62                      coeffs.append(-0.5 * sign)
63                      ops.append(Qtools.pauli_at(Z, e[0], nodes)@Qtools.pauli_at(Z,
                        ↪ e[1], nodes))
```

```
64              if reward == '11':
65                  for e in edges:
66                      coeffs.extend([0.25 * sign, -0.25 * sign, -0.25 * sign])
67                      ops.extend([Qtools.pauli_at(Z, e[0],
                        ↪  nodes)@Qtools.pauli_at(Z, e[1], nodes),
68                          Qtools.pauli_at(Z, e[0], nodes), Qtools.pauli_at(Z, e[1],
                            ↪  nodes)])
69          return Qtools.hamiltonian(coeffs, ops)
```

### A.1.8 Mixers

The source code listing 7 is adapted from Pennylane to return Qiskit compatible Hamiltonians. These Hamiltonians are usually used with others to fully specify an optimisation problem and are versatile enough to cover all the major graph optimisation problems commonly solved in Quantum Computing. With some small extensions, it can cover all the major graph optimisation problems.

Listing 7: Implementation of the Python mixers module. This module contains the Mixers class which is an implementation of the Hamiltonians in sections 4.3.5 to 4.3.6.

```
1  # Copyright 2018-2020 Xanadu Quantum Technologies Inc.
2  # Modification Copyright 2020-2021 Peter Demetriou
3  #
4  # Licensed under the Apache License, Version 2.0 (the "License");
5  # you may not use this file except in compliance with the License.
6  # You may obtain a copy of the License at
7  #
8      # http://www.apache.org/licenses/LICENSE-2.0
9  #
10 # Unless required by applicable law or agreed to in writing, software
11 # distributed under the License is distributed on an "AS IS" BASIS,
12 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 # See the License for the specific language governing permissions and
14 # limitations under the License.
```

```
15
16  import networkx as nx
17  from itertools import product
18  from functools import reduce
19  from qiskit.opflow import I, X, Z
20  from qcircsat.qtools import Qtools
21
22  class Mixers:
23      '''
24      Adapted from Pennylane to return Qiskit compatible Hamiltonians.
25      Creates mixers, allows specification whether coloured or
26      uncoloured nodes in a graph are assigned higher or lower energies.
27      These Hamiltonians are usually used with others in order to full specify
28      an optimisation problem and are versatile enough to cover most of the
29      major graph optimisation problems commonly solved in Quantum Computing.
30      '''
31
32      @staticmethod
33      def x_mixer(graph):
34          nodes = graph.nodes
35          coeffs = (1 for _ in nodes)
36          ops = (Qtools.pauli_at(X, at, nodes) for at in nodes)
37          return Qtools.hamiltonian(coeffs, ops)
38
39      @staticmethod
40      def bit_flip_mixer(graph, b):
41          sign = 1 if b == 0 else -1
42          coeffs = []
43          ops = []
44          nodes = graph.nodes
45          for i in nodes:
46              neighbours = list(graph.neighbors(i))
47              degree = len(neighbours)
48              n_ops = [[Qtools.pauli_at(X, i, nodes)]] + [[Qtools.pauli_at(I, n,
                ↪  nodes),
49                  Qtools.pauli_at(Z, n, nodes)] for n in neighbours]
```

```
            n_coeffs = [[1, sign] for n in neighbours]
            final_ops = (Qtools.compose_iter(m) for m in product(*n_ops))
            final_coeffs = ((0.5**degree)*reduce(lambda i,j: i*j, m)
                            for m in product(*n_coeffs))
            coeffs.extend(final_coeffs)
            ops.extend(final_ops)
        return Qtools.hamiltonian(coeffs, ops)

```

### A.1.9 Constraints

The Constraints class penalises or rewards certain nodes in cost Hamiltonians constructed from graphs. This is required to weight the constraints of the problem in the cost functions. The source code is provided in listing 8.

Listing 8: Implementation of the Python constraints module. This module contains the Constraints class and is used to reward or penalise terms in a Hamiltonian as shown in section 4.3.3.

```
import networkx as nx
from qiskit.opflow import Z
from qcircsat.qtools import Qtools

class Constraints:
    '''
    Penalises or rewards certain nodes in cost
    Hamiltonians constructed from graphs.
    '''

    @staticmethod
    def reward(graph, at):
        return 0.5*Qtools.pauli_at(Z, at, graph.nodes)

    @classmethod
```

```
16      def penalise(cls, graph, at):
17          return -cls.reward(graph, at)
18
```

### A.1.10 Problems

Listing 9 provides the Hamiltonians for MIS in use in VQE optimisation and the Hamiltonians required for MIS in the constrained and unconstrained QAOA optimisation case. It additionally allows switching nodes on and off and biasing other nodes. This class can be extended to cover the other major graph optimisation problems.

Listing 9: Implementation of the Python problems module. This module constains the Problems class. This contains the implementation of sections 2.7.6 to 2.7.8 and sections 4.3.7 to 4.3.9.

```
1   from qcircsat.drivers import Drivers
2   from qcircsat.mixers import Mixers
3   from qcircsat.constraints import Constraints
4
5   class Problems:
6       '''
7       Provides the Hamiltonians for MIS in use
8       in VQE optimisation and the Hamiltonians
9       required for MIS in the constrained and
10      unconstrained QAOA optimisation case.
11      Additionally allows switching nodes on
12      and off and biasing other nodes.
13      '''
14
15      @staticmethod
16      def vqe_max_independent_set(G, on=None, off=None, bias=None):
17          cost = 3*Drivers.edge_driver(G, ['10', '01', '00']) \
18                  + Drivers.bit_driver(G, 1)
19          if on is not None:
```

```
20              for t in on:
21                  cost += 5*Constraints.reward(G, t)
22          if off is not None:
23              for f in off:
24                  cost += 5*Constraints.penalise(G, f)
25          if bias is not None:
26              cost += 3*Constraints.reward(G, bias)
27          return cost.reduce()
28
29      @staticmethod
30      def qaoa_max_independent_set(G, constrained=True, on=None, off=None,
        ↪  bias=None):
31          if constrained:
32              phase = Drivers.bit_driver(G, 1)
33              mixer = Mixers.bit_flip_mixer(G, 0).reduce()
34          else:
35              phase = 3*Drivers.edge_driver(G, ['10', '01', '00']) \
36                      + Drivers.bit_driver(G, 1)
37              mixer = Mixers.x_mixer(G).reduce()
38          if on is not None:
39              for t in on:
40                  phase += 5*Constraints.reward(G, t)
41          if off is not None:
42              for f in off:
43                  phase += 5*Constraints.penalise(G, f)
44          if bias is not None:
45              phase += 5*Constraints.reward(G, bias)
46          return phase.reduce(), mixer
47
```

### A.1.11 Placement of Vertices in Graph Drawing

The circular placement of the nodes in the graph drawings comes from minimally adapted source code in the Networkx module.

# Appendix B

# Conference Paper

This conference paper is published by the 2021 IEEE Computer Society Annual Symposium on VLSI. It represents a snapshot of the dissertation at an earlier stage where QAOA was the focus.

2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) | 978-1-6654-3946-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISVLSI51109.2021.00083

# A Quantum Variational Approach to Debugging Combinational Logic Circuits

Peter Demetriou*, Conrad J. Haupt†, Ken J. Nixon‡
*School of Electrical & Information Engineering, University of the Witwatersrand, Johannesburg, South Africa*
*ORCID: 0000-0001-8223-9813, †ORCID: 0000-0003-3128-4985, ‡ORCID: 0000-0001-5391-8147

***Abstract*—The application of a quantum approach to combinational logic circuit debugging for future use in Very Large-Scale Integration system design is presented. Although previous work in this area has sped up debugging solutions through the use of satisfiability solvers, the solutions remain classically intractable for large systems. Additionally, whilst many quantum algorithms have shown promise in solving classically intractable problems, there have been few domain-specific implementations, particularly where hard and soft constraints exist and multiple solutions are required. A Quantum Alternating Operator Ansatz is applied to a reduction of the satisfiability problem into a maximum independent set problem. While the Quantum Alternating Operator Ansatz approach constrains the optimization to feasible solutions, the resulting ansatz is both too large for current Noisy Intermediate-Scale Quantum devices and to simulate using a Hamiltonian formulation with gradient-based optimisers. Therefore, a circuit driven simulation is preferred. A small example of a debugging problem is simulated using the aforementioned approach.**

***Keywords*-Combinational logic circuit debugging; Noisy Intermediate-Scale Quantum devices; Optimization; Quantum Alternating Operator Ansatz; Satisfiability**

## I. Introduction

Very Large-Scale Integration (VLSI) uses hardware description languages before manufacturing to verify the design of the integrated circuits. Estimates report that verification and the subsequent debugging task take up to 70% of the VLSI design process and that 60% of ASICs fail due to functional errors [1]. An automated debugging solution that scales can therefore, decrease the manufacturing cost and time of quality products. VLSI uses millions of logic gates and the systems continue to grow even larger, research is therefore conducted on finding powerful debugging solutions.

There are many fault diagnosis models employed in VLSI design such as the stuck-at, bridging, open or break, fault models [2]. Furthermore, advanced modeling of faults have been used for reversible and quantum circuits [3]–[7]. The work presented is a model free logic debugging approach to finding errors in the early stages of the VLSI design cycle before physical implementation. Given a circuit specification along with expected input-output combinations this approach finds gate implementation errors along with their locations. These sorts of errors are often caused by specification changes, bugs in automated software and human error [8].

Automated debugging solutions for multiple fault detection encoded as satisfiability (SAT) problems have been proposed, and extended to partial maximum satisfiability (MAX-SAT) problems for input-output constraints [8]–[10]. However, the solution search space for these problems increases exponentially as the circuit size increases since SAT is in the complexity class NP-Hard. This makes SAT classically intractable for large problems.

There are quantum computing algorithms that provide an exponential speed-up over the best known classical alternatives and cannot be efficiently simulated on a classical device, suggesting that it may be advantageous for certain problems. Although quantum computing currently has limitations concerning software and hardware, there is fruitful research to be performed on the available Noisy Intermediate-Scale Quantum (NISQ) devices [11], [12]. These devices are prone to error, but classical-quantum hybrid algorithms called variational algorithms, that implement a cost function dependent on quantum gate parameters that are classically optimized, do show immediate promise [13], [14].

## II. Quantum Computing and Combinatorial Optimization

Combinatorial optimization problems have many practical applications in engineering. The practical applications provide methods to cast these problems in familiar mathematical form. They also build a conceptual set of engineering problems in which these forms arise. Helpfully, classical Ising models of these applications are as ubiquitous as Ising models are in quantum computing and therefore form a bridge between the engineering problems and quantum computing [15], [16].

The SAT formulation of the debugging problem for VLSI and associated heuristics to reduce memory use and decrease the run time has been considered [8], as well as a MAX-SAT and approximate MAX-SAT two-step analyses as a debugging formulation for SoC design [9]. Further work has presented a MAX-SAT formulation of the debugging problem for VLSI with improvements to decrease the run time compared to conventional SAT solvers [10].

The preceding SAT approaches to debugging combinational logical circuits form the classical basis of the problem for which this work seeks to leverage quantum computation. It also shows the formal logical representations required for SAT debugging of the logic circuits (subsection IV-A showcases how the transformation is achieved) as well as providing a discussion of the significance of SAT for debugging applications.

978-1-6654-3946-6/21/$31.00 ©2021 IEEE
DOI 10.1109/ISVLSI51109.2021.00083

Variational algorithms that involve optimization of parametrised quantum gates have shown immediate promise. Examples include where an evolutionary algorithm was used to generate an ansatz for a Variational Quantum Eigensolver (VQE) with volume and noise resilience benefits [17], as well as a study where two versions of a novel adaptive optimizer required fewer measurements [14]. The noise resilience and comparatively shallow parameterised unitary gate variational circuits help negate many of the problems encountered on NISQ devices as well as allowing for solutions to problems where the exact circuit implementation is not known.

Farhi et al. are the progenitors of the Quantum Approximate Optimization Algorithm which approximates adiabatic quantum computing and can encode classical combinatorial problems using Ising Models [18]. The algorithm is most commonly used for combinatorial optimization. A generalization of the aforementioned algorithm that does not require the problem to be encoded as an Ising Model has been presented [19]. The generalization uses the same abbreviation, "QAOA", which henceforth will refer to the Quantum Alternating Operator Ansatz. This allows for constraining the space of possible solutions to only feasible solutions. Design patterns and considerations for many constrained combinatorial optimization problems expanding on previous work exist [20]. Zhou et al. suggest that QAOA is not prone to the same vanishing spectral gaps as adiabatic quantum computing [21]. The QAOA maximum independent set (MIS) solution is dependent on the choices of initial states and to produce non-symmetric output distributions in contrast to the maximum cut solution [22]. Borle et al. use QAOA for binary linear least squares, and although they achieve promising simulation results, they were unable to replicate the results on a NISQ device [23].

## III. Background

### A. Adiabatic Quantum Computing

There are different types of quantum computing, although this research focuses on gate-model quantum computing, adiabatic computing is useful for understanding the quantum algorithm to be used. In an adiabatic process, the time evolution of the system is such that there is no heat transfer concerning the system. In a quantum system with energy states described by an initial Hamiltonian $H_i$, in its lowest energy state, evolving into some final Hamiltonian $H_f$, the system will stay in its energy configuration in $H_f$ subject to the speed of reconfiguration and the gaps between the energy spectra. This is the adiabatic theorem. This may be utilized by encoding the solution of a classical problem as a ground state of a quantum many-body system and annealing (repeatedly re-configuring) a simple Hamiltonian, such as an equal superposition, into the encoded system. At absolute zero the ground state may be read out directly, at other temperatures the corresponding Boltzmann distribution may be sampled from to determine the ground state as described in equations (1) and (2)

$$P(i) = \frac{e^{\frac{-H_i}{kT}}}{Z} \tag{1}$$

$$Z = \sum_{j=1}^{M} e^{\frac{-H_j}{kT}} \tag{2}$$

The time evolution of a quantum system in (3), is governed by the Schrödinger equation:

$$i\hbar \frac{d}{dt} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle \tag{3}$$

whose solution in (4), is the unitary operator:

$$\hat{U}(t) = e^{\frac{-it}{\hbar}\hat{H}} \tag{4}$$

that provides the gate-model state time evolution in (5) for $t > t_0$

$$|\Psi(t)\rangle = \hat{U}(t) |\Psi(t_0)\rangle . \tag{5}$$

### B. The Quantum Alternating Operator Ansatz

QAOA is a variational algorithm for gate-model quantum computers that use the intuition of quantum annealing [18]. The adiabatic path evolution is split into $p$ steps using the Trotterisation technique in (6). Whitfield et al. provide a brief discussion of Trotterisation [24] which has been extended to show the conceptual analogy for comparing Trotterisation to quantum annealing [25].

$$U(t_i) = \prod_{i=0}^{p} e^{-iH_m\beta_i} e^{-iH_p\gamma_i} \tag{6}$$

The $p$ steps influence the approximation accuracy. Each discrete time step $i$ has parameters $\beta_i$ and $\gamma_i$ that operate as one of the angles or time steps for the general unitary transformations performed on the Bloch sphere and given by (7):

$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -e^{i\lambda}\sin(\frac{\theta}{2}) \\ e^{i\phi}\sin(\frac{\theta}{2}) & e^{i\lambda+i\phi}\cos(\frac{\theta}{2}) \end{pmatrix} \tag{7}$$

These parameters are classically optimized such that the energy of the system given by the objective function is optimal. The Hamiltonian's $H_m$ and $H_p$ are the mixer and phase Hamiltonian's, each encoding part of the combinatorial problem. QAOA includes more operators than only those corresponding to the time evolution of a fixed local Hamiltonian. It, therefore, can support ansatz where the evolution stays entirely within the feasible solutions using the mixer to constrain the problem rather than requiring additional terms to be added to an Ising model to penalise infeasible solutions [19], [20], [26].

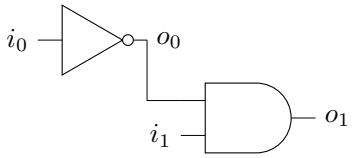


Figure 1. Unsatisfiable Combinational Logic Circuit

## IV. Methodology

As the debugging problem is NP-hard, a quantum algorithm is an appropriate candidate to seek a scaling advantage. However, even though the number of gates required by QAOA scales linearly with the problem size, the ansatz is nevertheless too large for current quantum hardware. Additionally, for greater approximation accuracy the repetitions of the ansatz enlarge the circuit to an infeasible depth.

Since SAT solutions have been useful for this type of debugging problem, the circuits first need to be reduced to a formal logical form to solve their associated SAT problem. Suppose the following circuit in Fig. 1, with inputs and outputs all true, is to be debugged. Trivially, either the `not` gate or `and` gate are unsatisfiable.

The SAT problem requires a Boolean representation that will evaluate to true if, and only if, the representation conforms to possible input-output assignments of the represented gate. For combinational logic gates, the Boolean representations need chained conjunctive clauses each containing a disjunction of literals, known as conjunctive normal form (CNF). Furthermore, each clause may contain a maximum of three literals (3-CNF) which is a canonical form for solving the SAT problem. This requires the Tseytin transform [27]. Section IV-A provides a derivation of the transform for the `and` gate using propositional logic since the usual citation does not contain this information [28].

### A. Tseytin Transforms

1) $o \Rightarrow i_o \wedge i_1$
2) $\neg o \Rightarrow \neg(i_o \wedge i_1)$
3) $\neg o \Rightarrow \neg i_o \vee \neg i_1$ — 2 DeM
4) $\neg\neg o \vee (\neg i_o \vee \neg i_1)$ — 3 Imp
5) $o \vee (\neg i_o \vee \neg i_1)$ — 4 DN
6) $o \vee \neg i_o \vee \neg i_1$ — 5 Assoc
7) $\neg o \vee (i_o \wedge i_1)$ — 1 Imp
8) $(\neg o \vee i_o) \wedge (\neg o \vee i_1)$ — 7 Dist
9) $(o \vee \neg i_o \vee \neg i_1) \wedge (\neg o \vee i_o) \wedge (\neg o \vee i_1)$ — 6, 8 Conj

Note that the transform introduces a constant factor of literals per gate and is therefore, $O(n)$ in the number of gates. Note the two premises in 1 and 2 exhaustively describe states of an `and` gate, and the conclusion in premise 9 is the 3-CNF representation.

The `not` gate in the circuit in Fig. 1 may be represented by (8) (the superscripts are explained in section (IV-B)):

$$(\neg i_0^0 \vee \neg o_0^5) \wedge (i_0^1 \vee o_0^6), \tag{8}$$

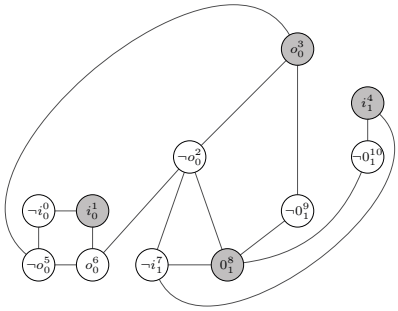


Figure 2. Graph with a colored Maximum Independent Set

and the `and` gate as (9):

$$(\neg o_0^2 \vee \neg i_1^7 \vee o_1^8) \wedge (o_0^3 \vee \neg o_1^9) \wedge (i_1^4 \vee \neg o_1^{10}). \tag{9}$$

### B. Independent Set Problem

There is a subclass of NP problems known as NP-complete problems. MAX-3SAT is such a problem and can be reduced to several other NP-Complete problems such as IS, maximum clique, and vertex cover [29]. It is these NP-complete representations of the problem for which there are available variational formulations. Therefore a reduction from MAX-3SAT to one of these forms is a requirement. Although each of these reductions only costs polynomial time and their implementations do not confer any resource advantages over each other, they can require an extra step. For example, inverting the IS graph for the maximum clique representation [19]. Next, the MAX-3SAT problem is reduced to the MIS problem. In Fig. 2 the 3-CNF formula is represented as a graph and one of the MIS is colored in, the superscripts are shared with equations (8) and (9), and are the bit string indices in the problem encoding.

An IS is a set $S$ in a graph $G = (V, E)$ where for any two vertices in $S$ there are no edges between them. The largest of such sets is at least one of the MIS. The graph may be constructed by creating adjacent vertices for each clause (that is with edges between them) such that at least one of the literals in the disjunction is forced to be true. However, since the truth assignments are to be consistent, a connecting edge between all literals and their negations is required.

There are no colored vertices in the first `not` gate clause depicted by the Fig. 2 solution, identifying it as a suspect gate error. There may be multiple graph solutions that enumerate gate errors of differing cardinality.

### C. The Quantum Alternating Operator Ansatz for Maximum Independent Set

The design parameters for QAOA are the initial state, the phase operators, and the mixing operators. A derivation for the design parameters is given below [19], [20], [22], [26], [30]–[32].

For a graph $G$ with $V(G)$ as its vertices, $E(G)$ its edges and $N(v)$, $d(v)$ the neighbors and degree of vertex $v$ respectively, let $x_v$ denote an n-bit string $\{0,1\}^n$ corresponding to a graph coloring. The MIS objective function is the sum of the bits:

$$f(x_v) = \sum_{v \in V(G)}^{n} x_v. \tag{10}$$

In order to implement the hard constraints of the MIS problem for a colored vertex, (10) is constrained by (11):

$$\bigwedge_{w \in N(v)} \neg x_w = 1. \tag{11}$$

The Boolean formula needs to be encoded as a Hamiltonian operator. Transformations are provided in [26], [32]–[34]. Using (10) to find the bit driver Hamiltonian (12).

$$H_p = \frac{1}{2} \sum_{v \in V(G)} \left( \mathbb{I} - \sigma_v^Z \right). \tag{12}$$

The bit-flip mixer Hamiltonian is described in (13):

$$H_m = \sum_{v \in V(G)} \frac{\sigma_v^X}{2^{d(v)}} \prod_{w \in N(v)} \left( \mathbb{I} + \sigma_w^Z \right) \tag{13}$$

The expansions of the Hamiltonian's assume that multiplication between the Pauli operator is tensor multiplication such that $\sigma_v^X \sigma_w^Z$ is short-hand for $\sigma_v^X \otimes \sigma_w^Z$.

Using Trotterisation from (6) the alternating ansatz can be prepared. The initial state $|s\rangle$ must be a valid IS since the solution is constrained only to feasible states during the optimization. It is therefore recommended to use $|0\rangle^{\otimes n}$ since this is always a valid IS.

However, because this application contains soft constraints concerning the circuit input and output values, the initial state is set to make the inputs and outputs true with a single $\sigma^X$ gate. Additionally, since these values are predetermined and by implication so are their negations, which in this case would be set to false, the initial state used is, $|01001000100\rangle$. Note, the bit indexing is from left to right following Fig. 2.

Equation (12) transforms the eigenvalues of the Pauli $\sigma^Z$ operator to 0 and 1 and is implemented by $R_z$ gates. Equation (13) may be implemented by a multiply open-controlled (switches on zero, not one) Toffoli gate with a bit flip as a target and a full uncomputation of the Toffoli gate after the target. Even though the graph is undirected every vertex needs to be treated as both a target and a neighbor and therefore the index iteration order in (13) is important. Since such a gate is not available in the software used it is required to decompose it into available gates.

Fig. 3 shows the two equivalent circuits side by side. It should be noted that in general, the ansatz will require the maximum degree of the graph $\Delta(G)$, minus one extra ancilla qubit. The open controls may be implemented by a $X$ (equivalent to $\sigma^X$) gate to flip the bit in front of the graph and an immediate additional $X$ gate to uncompute thereafter. The multiply controlled Toffoli is then implemented by chaining the outputs and inputs of ordinary Toffoli gates [35]. The mixer, therefore, corresponds to the neighbors of each vertex controlling whether it gets flipped, based on whether or not they have been flipped. Lastly, note that the two inner $X$ may be dropped as shown in the full ansatz in Fig. 4.

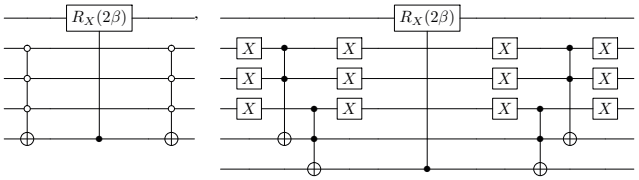


Figure 3. Mixing Operator Equivalent Circuits

In Fig. 4 the ansatz has barriers for the initial state, phase operator and each of the sequential mixing operators. The sequential mixing operators may contain adjacent $X$ ($\sigma^X$) gates that can be optimized out depending on the ordering. The order of the mixing operators also affects results in the case that there is more than one MIS since the first sequential mixer will flip a specific qubit that will therefore constrain the rest of the possible flips. This effect appears to diminish as $p$ increases.

## V. Parameter Optimization

Gradient descent methods are generally not used for two reasons: the number of circuit evaluations performed for optimizing the parameters is expensive and there are often flat regions in the cost functions parameter space called barren plateaus, which do not allow for gradient-based optimization [36]. However, since this particular problem is small enough for barren plateaus to be unlikely and because gradient descent finds all the MIS solutions whilst other optimisers tend to find only one such solution. The problem is simulated with software that uses gradient-based optimization on a GPU backend to decrease the optimization time.

### A. Measurement

Fig. 5 shows preliminary results of running the QAOA algorithm with Pennylane using a Tensorflow GPU backend and Adagrad optimizer at different levels of discretisation [37]. For $p = 3$, it can be seen that the most frequently sampled bit string corresponds to the solution shown in Fig. 2, whilst the second most sampled solution also presents an MIS. It should be noted that a sweep through the parameter space yields only feasible solutions which indicates the validity of the circuit construction since it is much more likely to pick out an infeasible IS solution among a solution space of $2^n$ than not. The specific solutions are further verified using a classical approach.

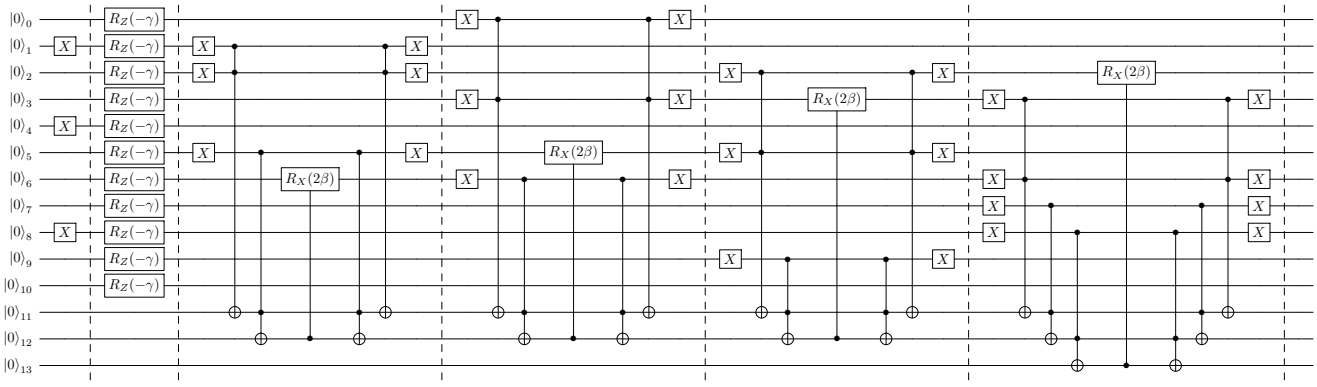

Figure 4. Ansatz for $p = 1$

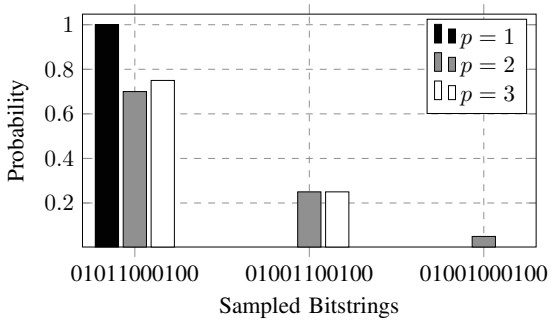


Figure 5. The Quantum Alternating Operator Ansatz Sampled Bitstrings

## VI. Discussion

Variational algorithms may require many measurements to optimize the objective function, and can, therefore, take a long time to find approximate solutions [14]. However, for larger problems, they show potential to scale better than classical alternatives. Owing to the constrained nature of the QAOA, fewer iterations are required in non-constrained variational algorithms. Nevertheless, QAOA requires a quantum circuit that is too deep to run on a NISQ device.

VQE, another variational algorithm, finds the smallest eigenvalue of a system described by a Hamiltonian. Since the eigenvalues of a Hamiltonian are the energy spectrum of the quantum system, VQE is used for finding ground-states in quantum chemistry and is less frequently applied to combinatorial optimization [31]. However, VQE can use generic ansatz designed to be small enough to run on the hardware and may be more appropriate as an immediate solution. Although for large enough problems, a larger ansatz will be required for a more comprehensive search of the solution space, leading to the same difficulties. Therefore, a quantum advantage for debugging is only expected when much larger quantum machines are available, even though a fully fault-tolerant system will not be required.

## VII. Conclusion

This research seeks to provide a NISQ variational method that solves a circuit debugging problem. The research expands into domain-specific experimentation of quantum optimization, providing the tools to make the problem amenable to QAOA and VQE. Although careful considerations were given to reduce circuit depth, width, and noise, the QAOA formulation is not feasible on current NISQ devices. VQE should be investigated for short term small problems on NISQ hardware using the measurement frugal optimisers and should be run on quantum hardware to see if the *noisiness* of the quantum devices can be overcome with the ultimate aim of gaining a quantum advantage over currently known algorithms for an NP-hard combinatorial optimization required to debug a combinational logic circuit.

## References


[1] A. Veneris, B. Keng, and S. Safarpour, "From RTL to silicon: The case for automated debug," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, Jan. 2011, pp. 306–310, iSSN: 2153-697X.

[2] R. Aitken, "Modeling the unmodelable: algorithmic fault diagnosis," *IEEE Design Test of Computers*, vol. 14, no. 3, pp. 98–103, Jul. 1997.

[3] I. Polian, J. P. Hayes, S. M. Reddy, and B. Becker, "Modeling and Mitigating Transient Errors in Logic Circuits," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 4, pp. 537–547, Jul. 2011, conference Name: IEEE Transactions on Dependable and Secure Computing.

[4] I. Polian, T. Fiehn, B. Becker, and J. Hayes, "A Family of Logical Fault Models for Reversible Circuits," in *14th Asian Test Symposium (ATS'05)*, Dec. 2005, pp. 422–427, iSSN: 2377-5386.

[5] I. Polian and J. P. Hayes, "Advanced modeling of faults in Reversible circuits," in *2010 East-West Design Test Symposium (EWDTS)*, Sep. 2010, pp. 376–381.

[6] A. Paler, I. Polian, and J. P. Hayes, "Detection and diagnosis of faulty quantum circuits," in *17th Asia and South Pacific Design Automation Conference*, Jan. 2012, pp. 181–186, iSSN: 2153-697X.

[7] A. Paler, A. Alaghi, I. Polian, and J. P. Hayes, "Tomographic Testing and Validation of Probabilistic Circuits," in *2011 Sixteenth IEEE European Test Symposium*, May 2011, pp. 63–68, iSSN: 1558-1780.

[8] A. Smith, A. Veneris, M. Ali, and A. Viglas, "Fault diagnosis and logic debugging using Boolean satisfiability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 10, pp. 1606–1621, Oct. 2005.

[9] S. Safarpour, H. Mangassarian, A. Veneris, M. H. Liffiton, and K. A. Sakallah, "Improved Design Debugging Using Maximum Satisfiability," in *Formal Methods in Computer Aided Design (FMCAD'07)*, Nov. 2007, pp. 13–19.

[10] Y. Chen, S. Safarpour, J. Marques-Silva, and A. Veneris, "Automated Design Debugging With Maximum Satisfiability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 11, pp. 1804–1817, Nov. 2010.

[11] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018. [Online]. Available: http://arxiv.org/abs/1801.00862

[12] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum Machine Learning," *Nature*, vol. 549, no. 7671, pp. 195–202, Sep. 2017. [Online]. Available: http://arxiv.org/abs/1611.09347

[13] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, Nov. 2019. [Online]. Available: http://arxiv.org/abs/1906.07682

[14] J. M. Kübler, A. Arrasmith, L. Cincio, and P. J. Coles, "An Adaptive Optimizer for Measurement-Frugal Variational Algorithms," *Quantum*, vol. 4, p. 263, May 2020. [Online]. Available: http://arxiv.org/abs/1909.09083

[15] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, 2014. [Online]. Available: http://arxiv.org/abs/1302.5843

[16] F. Glover, G. Kochenberger, and Y. Du, "A Tutorial on Formulating and Using QUBO Models," *arXiv:1811.11538 [quant-ph]*, Nov. 2019. [Online]. Available: http://arxiv.org/abs/1811.11538

[17] A. G. Rattew, S. Hu, M. Pistoia, R. Chen, and S. Wood, "A Domain-agnostic, Noise-resistant, Hardware-efficient Evolutionary Variational Quantum Eigensolver," *arXiv:1910.09694 [quant-ph]*, Jan. 2020. [Online]. Available: http://arxiv.org/abs/1910.09694

[18] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm," *arXiv:1411.4028 [quant-ph]*, Nov. 2014. [Online]. Available: http://arxiv.org/abs/1411.4028

[19] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz," *Algorithms*, vol. 12, no. 2, p. 34, Feb. 2019. [Online]. Available: http://arxiv.org/abs/1709.03489

[20] S. Hadfield, Z. Wang, E. G. Rieffel, B. O'Gorman, D. Venturelli, and R. Biswas, "Quantum Approximate Optimization with Hard and Soft Constraints," in *Proceedings of the Second International Workshop on Post Moores Era Supercomputing - PMES'17*. Denver, CO, USA: ACM Press, 2017, pp. 15–21. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3149526.3149530

[21] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices," *Physical Review X*, vol. 10, no. 2, p. 021067, Jun. 2020. [Online]. Available: http://arxiv.org/abs/1812.01041

[22] Z. H. Saleem, "Max Independent Set and Quantum Alternating Operator Ansatz," *arXiv:1905.04809 [quant-ph]*, Apr. 2020. [Online]. Available: http://arxiv.org/abs/1905.04809

[23] A. Borle, V. E. Elfving, and S. J. Lomonaco, "Quantum Approximate Optimization for Hard Problems in Linear Algebra," *arXiv:2006.15438 [quant-ph]*, Jun. 2020. [Online]. Available: http://arxiv.org/abs/2006.15438

[24] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, "Simulation of Electronic Structure Hamiltonians Using Quantum Computers," *Molecular Physics*, vol. 109, no. 5, pp. 735–750, Mar. 2011. [Online]. Available: http://arxiv.org/abs/1001.3855

[25] G. Verdon, M. Broughton, and J. Biamonte, "A quantum algorithm to train neural networks using low-depth circuits," *arXiv:1712.05304 [cond-mat, physics:quant-ph]*, Aug. 2019. [Online]. Available: http://arxiv.org/abs/1712.05304

[26] S. Hadfield, "Quantum Algorithms for Scientific Computing and Approximate Optimization," *arXiv:1805.03265 [quant-ph]*, May 2018. [Online]. Available: http://arxiv.org/abs/1805.03265

[27] G. Tseitin, "On the complexity of proofs in propositional logics," in *Seminars in Mathematics*, vol. 8, 1970, pp. 466–483.

[28] P. J. Hurley, *A concise introduction to logic*. Nelson Education, 2014.

[29] D. Sanjoy, P. Christos, and V. Umesh, *Algorithms*. McGraw-Hill Boston, 2006.

[30] H. Pichler, S.-T. Wang, L. Zhou, S. Choi, and M. D. Lukin, "Quantum Optimization for Maximum Independent Set Using Rydberg Atom Arrays," *arXiv:1808.10816 [cond-mat, physics:physics, physics:quant-ph]*, Aug. 2018. [Online]. Available: http://arxiv.org/abs/1808.10816

[31] J. Choi, S. Oh, and J. Kim, "The Useful Quantum Computing Techniques for Artificial Intelligence Engineers," in *2020 International Conference on Information Networking (ICOIN)*, Jan. 2020, pp. 1–3, iSSN: 1976-7684.

[32] J. Choi and J. Kim, "A Tutorial on Quantum Approximate Optimization Algorithm (QAOA): Fundamentals and Applications," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct. 2019, pp. 138–142, iSSN: 2162-1233.

[33] J. D. Whitfield, M. Faccin, and J. D. Biamonte, "Ground State Spin Logic," *EPL (Europhysics Letters)*, vol. 99, no. 5, p. 57004, Sep. 2012. [Online]. Available: http://arxiv.org/abs/1205.1742

[34] S. Hadfield, "On the representation of Boolean and real functions as Hamiltonians for quantum computing," *arXiv:1804.09130 [quant-ph]*, Apr. 2018. [Online]. Available: http://arxiv.org/abs/1804.09130

[35] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, 10th ed. Cambridge ; New York: Cambridge University Press, 2010.

[36] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature Communications*, vol. 9, no. 1, p. 4812, Dec. 2018. [Online]. Available: http://arxiv.org/abs/1803.11173

[37] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Száva, and N. Killoran, "PennyLane: Automatic differentiation of hybrid quantum-classical computations," *arXiv:1811.04968 [physics, physics:quant-ph]*, Feb. 2020. [Online]. Available: http://arxiv.org/abs/1811.04968

# Appendix C

# The Manhattan QPU

This appendix contains information on the specific QPU used. It contains the topology of the hardware as well as the calibration data taken before the runs.

## C.1 IBM Manhattan QPU

labelsec:IBM Manhattan QPU Figure C.1.1 shows the topology of the hardware used to run the VQE algorithm on. There are much fewer edge connections between the physical qubits than what is required for the full adder in figure 5.9. This indicates that a large amount of `SWAP` gates will be necessary for the circuitry since the problem topology is not embeddable on the hardware.

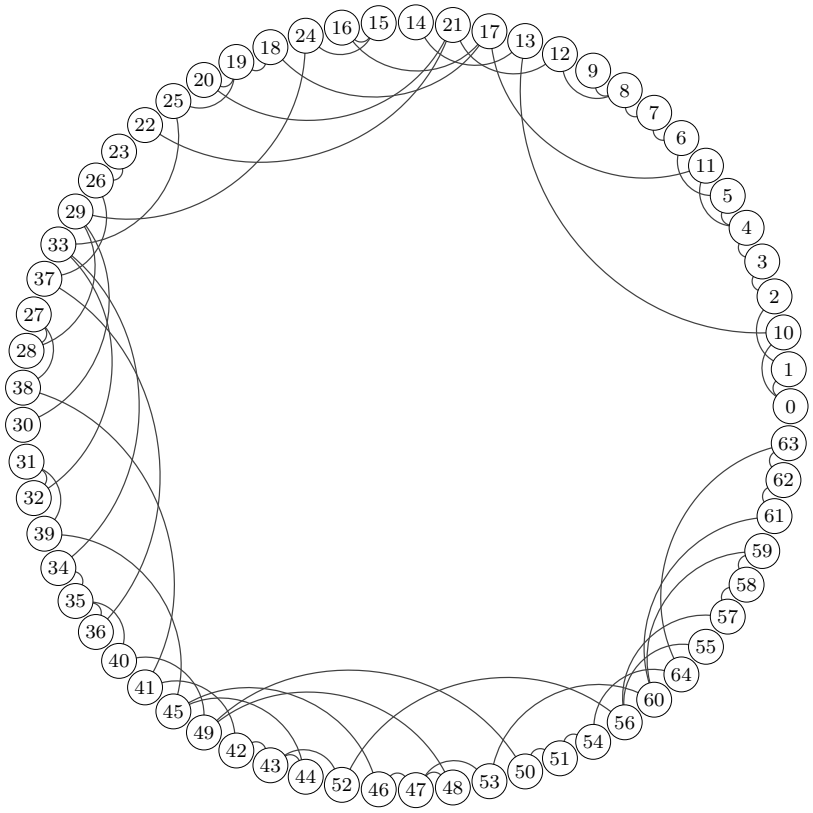


Figure C.1.1: The Manhattan QPU topology represented as a graph.

## C.2 Error Calibration Data

Table C.1 shows some of the calibration data at the time of the runs for the Manhattan hardware. Some error rates on this calibration in the table C.1 are highlighted as values that are apt to cause calculation errors. However it is the case that calibration was done during the runs and these numbers do fluctuate slightly, therefore this is just to get an idea of which qubits, in particular, might have affected the results.

Table C.1: Error calibration data for IBM Manhattan.

| Q | T1 (us) | T2 (us) | Readout error | $I$ error | $X$ error | `CNOT` error |
|---|---|---|---|---|---|---|
| 1 | 66.89 | 80.48 | $7.610 \times 10^{-2}$ | $4.308 \times 10^{-4}$ | $4.308 \times 10^{-4}$ | (0, 10): $1.617 \times 10^{-2}$ |
| | | | | | | (0, 1): $1.598 \times 10^{-2}$ |
| 2 | 77.61 | 89.3 | $2.570 \times 10^{-2}$ | $4.159 \times 10^{-4}$ | $4.159 \times 10^{-4}$ | (1, 0): $1.598 \times 10^{-2}$ |
| | | | | | | (1, 2): $1.256 \times 10^{-2}$ |
| 3 | 64.61 | 78.01 | $2.820 \times 10^{-2}$ | $3.578 \times 10^{-4}$ | $3.578 \times 10^{-4}$ | (2, 1): $1.256 \times 10^{-2}$ |
| | | | | | | (2, 3): $1.553 \times 10^{-2}$ |
| 4 | 57.4 | 78.86 | $1.710 \times 10^{-2}$ | $7.135 \times 10^{-4}$ | $7.135 \times 10^{-4}$ | (3, 4): $1.557 \times 10^{-2}$ |
| | | | | | | (3, 2): $1.553 \times 10^{-2}$ |
| 5 | 51.82 | 82.96 | $2.800 \times 10^{-2}$ | $7.776 \times 10^{-4}$ | $7.776 \times 10^{-4}$ | (4, 3): $1.557 \times 10^{-2}$ |
| | | | | | | (4, 11): $1.459 \times 10^{-2}$ |
| | | | | | | (4, 5): $1.334 \times 10^{-2}$ |
| 6 | 70.07 | 50.84 | $4.680 \times 10^{-2}$ | $5.116 \times 10^{-4}$ | $5.116 \times 10^{-4}$ | (5, 6): $1.559 \times 10^{-2}$ |
| | | | | | | (5, 4): $1.334 \times 10^{-2}$ |
| 7 | 66.98 | 78.71 | $2.270 \times 10^{-2}$ | $4.808 \times 10^{-4}$ | $4.808 \times 10^{-4}$ | (6, 5): $1.559 \times 10^{-2}$ |
| | | | | | | (6, 7): $1.062 \times 10^{-2}$ |
| 8 | 92.07 | 110.06 | $1.820 \times 10^{-2}$ | $3.356 \times 10^{-4}$ | $3.356 \times 10^{-4}$ | (7, 6): $1.062 \times 10^{-2}$ |
| | | | | | | (7, 8): $1.793 \times 10^{-2}$ |
| 9 | 51.08 | 54.32 | $4.410 \times 10^{-2}$ | $2.145 \times 10^{-3}$ | $2.145 \times 10^{-3}$ | (8, 9): $3.200 \times 10^{-2}$ |
| | | | | | | (8, 12): $2.312 \times 10^{-2}$ |
| | | | | | | (8, 7): $1.793 \times 10^{-2}$ |
| 10 | 57.53 | 73.54 | $1.020 \times 10^{-2}$ | $3.564 \times 10^{-4}$ | $3.564 \times 10^{-4}$ | (9, 8): $3.200 \times 10^{-2}$ |
| 11 | 78.46 | 73.35 | $3.400 \times 10^{-2}$ | $4.651 \times 10^{-4}$ | $4.651 \times 10^{-4}$ | (10, 0): $1.617 \times 10^{-2}$ |
| | | | | | | (10, 13): $1.127 \times 10^{-2}$ |
| 12 | 55.3 | 109.69 | $2.830 \times 10^{-1}$ | $6.993 \times 10^{-4}$ | $6.993 \times 10^{-4}$ | (11, 17): $1.071 \times 10^{-2}$ |
| | | | | | | (11, 4): $1.459 \times 10^{-2}$ |
| 13 | 42.12 | 76.85 | $2.040 \times 10^{-2}$ | $4.122 \times 10^{-4}$ | $4.122 \times 10^{-4}$ | (12, 21): $2.433 \times 10^{-2}$ |
| | | | | | | (12, 8): $2.312 \times 10^{-2}$ |
| 14 | 60.31 | 66.9 | $5.660 \times 10^{-2}$ | $3.597 \times 10^{-4}$ | $3.597 \times 10^{-4}$ | (13, 10): $1.127 \times 10^{-2}$ |
| | | | | | | (13, 14): $1.222 \times 10^{-2}$ |
| 15 | 72.46 | 70.12 | $7.940 \times 10^{-2}$ | $5.567 \times 10^{-4}$ | $5.567 \times 10^{-4}$ | (14, 15): $1.842 \times 10^{-2}$ |
| | | | | | | (14, 13): $1.222 \times 10^{-2}$ |

**Table C.1 continued from previous page**

| Q | T1 (us) | T2 (us) | Readout error | $I$ error | $X$ error | CNOT error |
|---|---|---|---|---|---|---|
| 16 | 92.06 | 109.42 | $2.380 \times 10^{-2}$ | $3.188 \times 10^{-4}$ | $3.188 \times 10^{-4}$ | (15, 14): $1.842 \times 10^{-2}$ |
| | | | | | | (15, 24): $7.823 \times 10^{-3}$ |
| | | | | | | (15, 16): $9.464 \times 10^{-3}$ |
| 17 | 98.45 | 96.62 | $2.420 \times 10^{-2}$ | $2.915 \times 10^{-4}$ | $2.915 \times 10^{-4}$ | (16, 17): $1.061 \times 10^{-2}$ |
| | | | | | | (16, 15): $9.464 \times 10^{-3}$ |
| 18 | 59.27 | 75.21 | $4.200 \times 10^{-2}$ | $4.813 \times 10^{-4}$ | $4.813 \times 10^{-4}$ | (17, 11): $1.071 \times 10^{-2}$ |
| | | | | | | (17, 16): $1.061 \times 10^{-2}$ |
| | | | | | | (17, 18): $1.566 \times 10^{-2}$ |
| 19 | 63.22 | 84.25 | $1.570 \times 10^{-2}$ | $7.577 \times 10^{-4}$ | $7.577 \times 10^{-4}$ | (18, 19): $1.473 \times 10^{-2}$ |
| | | | | | | (18, 17): $1.566 \times 10^{-2}$ |
| 20 | 63.09 | 80.7 | $1.420 \times 10^{-2}$ | $4.846 \times 10^{-4}$ | $4.846 \times 10^{-4}$ | (19, 20): $2.408 \times 10^{-2}$ |
| | | | | | | (19, 18): $1.473 \times 10^{-2}$ |
| | | | | | | (19, 25): $2.422 \times 10^{-2}$ |
| 21 | 43.33 | 29.07 | $2.260 \times 10^{-2}$ | $5.194 \times 10^{-4}$ | $5.194 \times 10^{-4}$ | (20, 19): $2.408 \times 10^{-2}$ |
| | | | | | | (20, 21): $1.157 \times 10^{-2}$ |
| 22 | 59.23 | 91.41 | $2.190 \times 10^{-2}$ | $4.333 \times 10^{-4}$ | $4.333 \times 10^{-4}$ | (21, 12): $2.433 \times 10^{-2}$ |
| | | | | | | (21, 22): $1.716 \times 10^{-2}$ |
| | | | | | | (21, 20): $1.157 \times 10^{-2}$ |
| 23 | 62.37 | 95.03 | $2.350 \times 10^{-2}$ | $5.019 \times 10^{-4}$ | $5.019 \times 10^{-4}$ | (22, 21): $1.716 \times 10^{-2}$ |
| | | | | | | (22, 23): $1.496 \times 10^{-2}$ |
| 24 | 28.77 | 45.84 | $4.620 \times 10^{-2}$ | $6.152 \times 10^{-4}$ | $6.152 \times 10^{-4}$ | (23, 26): $1.505 \times 10^{-2}$ |
| | | | | | | (23, 22): $1.496 \times 10^{-2}$ |
| 25 | 65.72 | 70.75 | $1.440 \times 10^{-2}$ | $2.995 \times 10^{-4}$ | $2.995 \times 10^{-4}$ | (24, 15): $7.823 \times 10^{-3}$ |
| | | | | | | (24, 29): $7.870 \times 10^{-3}$ |
| 26 | 47.9 | 51.12 | $2.010 \times 10^{-2}$ | $4.201 \times 10^{-4}$ | $4.201 \times 10^{-4}$ | (25, 33): $1.130 \times 10^{-2}$ |
| | | | | | | (25, 19): $2.422 \times 10^{-2}$ |
| 27 | 84.23 | 111.21 | $1.830 \times 10^{-2}$ | $2.697 \times 10^{-4}$ | $2.697 \times 10^{-4}$ | (26, 23): $1.505 \times 10^{-2}$ |
| | | | | | | (26, 37): $9.074 \times 10^{-3}$ |
| 28 | 69.86 | 96.65 | $3.380 \times 10^{-2}$ | $3.575 \times 10^{-4}$ | $3.575 \times 10^{-4}$ | (27, 28): $1.897 \times 10^{-2}$ |
| | | | | | | (27, 38): $1.289 \times 10^{-2}$ |
| 29 | 28.72 | 36.91 | $1.610 \times 10^{-2}$ | $7.541 \times 10^{-4}$ | $7.541 \times 10^{-4}$ | (28, 29): $1.414 \times 10^{-2}$ |
| | | | | | | (28, 27): $1.897 \times 10^{-2}$ |

**Table C.1 continued from previous page**

| Q | T1 (us) | T2 (us) | Readout error | $I$ error | $X$ error | `CNOT` error |
|---|---|---|---|---|---|---|
| 30 | 90.38 | 86.8 | $8.800 \times 10^{-3}$ | $2.651 \times 10^{-4}$ | $2.651 \times 10^{-4}$ | (29, 28): $1.414 \times 10^{-2}$ |
| | | | | | | (29, 24): $7.870 \times 10^{-3}$ |
| | | | | | | (29, 30): $7.934 \times 10^{-3}$ |
| 31 | 74.54 | 38.67 | $1.010 \times 10^{-2}$ | $4.353 \times 10^{-4}$ | $4.353 \times 10^{-4}$ | (30, 31): $1.663 \times 10^{-2}$ |
| | | | | | | (30, 29): $7.934 \times 10^{-3}$ |
| 32 | 39.82 | 65.25 | $4.410 \times 10^{-2}$ | $7.601 \times 10^{-4}$ | $7.601 \times 10^{-4}$ | (31, 30): $1.663 \times 10^{-2}$ |
| | | | | | | (31, 32): $1.346 \times 10^{-2}$ |
| | | | | | | (31, 39): $1.211 \times 10^{-2}$ |
| 33 | 62.82 | 131.15 | $1.720 \times 10^{-2}$ | $3.681 \times 10^{-4}$ | $3.681 \times 10^{-4}$ | (32, 31): $1.346 \times 10^{-2}$ |
| | | | | | | (32, 33): $1.342 \times 10^{-2}$ |
| 34 | 65.65 | 90.73 | $2.010 \times 10^{-2}$ | $2.606 \times 10^{-4}$ | $2.606 \times 10^{-4}$ | (33, 34): $7.228 \times 10^{-3}$ |
| | | | | | | (33, 25): $1.130 \times 10^{-2}$ |
| | | | | | | (33, 32): $1.342 \times 10^{-2}$ |
| 35 | 66.1 | 118.8 | $1.260 \times 10^{-2}$ | $2.826 \times 10^{-4}$ | $2.826 \times 10^{-4}$ | (34, 33): $7.228 \times 10^{-3}$ |
| | | | | | | (34, 35): $1.188 \times 10^{-2}$ |
| 36 | 82.44 | 82.09 | $3.240 \times 10^{-2}$ | $4.598 \times 10^{-4}$ | $4.598 \times 10^{-4}$ | (35, 40): $1.319 \times 10^{-2}$ |
| | | | | | | (35, 34): $1.188 \times 10^{-2}$ |
| | | | | | | (35, 36): $1.232 \times 10^{-2}$ |
| 37 | 61.2 | 113.21 | $1.150 \times 10^{-2}$ | $3.407 \times 10^{-4}$ | $3.407 \times 10^{-4}$ | (36, 35): $1.232 \times 10^{-2}$ |
| | | | | | | (36, 37): $1.271 \times 10^{-2}$ |
| 38 | 69.63 | 63.5 | $1.830 \times 10^{-2}$ | $3.608 \times 10^{-4}$ | $3.608 \times 10^{-4}$ | (37, 26): $9.074 \times 10^{-3}$ |
| | | | | | | (37, 36): $1.271 \times 10^{-2}$ |
| 39 | 48.87 | 63.06 | $1.163 \times 10^{-1}$ | $4.280 \times 10^{-4}$ | $4.280 \times 10^{-4}$ | (38, 27): $1.289 \times 10^{-2}$ |
| | | | | | | (38, 41): 1.000 |
| 40 | 72.38 | 33.96 | $1.000 \times 10^{-2}$ | $3.066 \times 10^{-4}$ | $3.066 \times 10^{-4}$ | (39, 31): $1.211 \times 10^{-2}$ |
| | | | | | | (39, 45): $1.395 \times 10^{-2}$ |
| 41 | 85.69 | 117.18 | $2.920 \times 10^{-2}$ | $2.477 \times 10^{-4}$ | $2.477 \times 10^{-4}$ | (40, 35): $1.319 \times 10^{-2}$ |
| | | | | | | (40, 49): $1.271 \times 10^{-2}$ |
| 42 | 76.77 | 27.11 | $8.940 \times 10^{-2}$ | $5.858 \times 10^{-4}$ | $5.858 \times 10^{-4}$ | (41, 42): $2.480 \times 10^{-2}$ |
| | | | | | | (41, 38): 1.000 |
| 43 | 61.85 | 78.33 | $3.040 \times 10^{-2}$ | $3.601 \times 10^{-4}$ | $3.601 \times 10^{-4}$ | (42, 41): $2.480 \times 10^{-2}$ |
| | | | | | | (42, 43): $8.944 \times 10^{-2}$ |

**Table C.1 continued from previous page**

| Q | T1 (us) | T2 (us) | Readout error | $I$ error | $X$ error | `CNOT` error |
|---|---|---|---|---|---|---|
| 44 | 48.55 | 16.74 | $8.130 \times 10^{-2}$ | $7.160 \times 10^{-2}$ | $7.160 \times 10^{-2}$ | (43, 42): $8.944 \times 10^{-2}$ |
| | | | | | | (43, 44): $9.811 \times 10^{-2}$ |
| | | | | | | (43, 52): 1.000 |
| 45 | 90.22 | 82.82 | $4.450 \times 10^{-2}$ | $3.732 \times 10^{-4}$ | $3.732 \times 10^{-4}$ | (44, 45): $9.950 \times 10^{-3}$ |
| | | | | | | (44, 43): $9.811 \times 10^{-2}$ |
| 46 | 83.95 | 91.64 | $2.710 \times 10^{-2}$ | $2.638 \times 10^{-4}$ | $2.638 \times 10^{-4}$ | (45, 44): $9.950 \times 10^{-3}$ |
| | | | | | | (45, 39): $1.395 \times 10^{-2}$ |
| | | | | | | (45, 46): $1.099 \times 10^{-2}$ |
| 47 | 60.28 | 86.98 | $6.280 \times 10^{-2}$ | $3.271 \times 10^{-4}$ | $3.271 \times 10^{-4}$ | (46, 47): $1.054 \times 10^{-2}$ |
| | | | | | | (46, 45): $1.099 \times 10^{-2}$ |
| 48 | 81.16 | 92.16 | $1.450 \times 10^{-2}$ | $3.125 \times 10^{-4}$ | $3.125 \times 10^{-4}$ | (47, 48): $8.111 \times 10^{-3}$ |
| | | | | | | (47, 53): $1.097 \times 10^{-2}$ |
| | | | | | | (47, 46): $1.054 \times 10^{-2}$ |
| 49 | 64.99 | 94.84 | $2.540 \times 10^{-2}$ | $3.840 \times 10^{-4}$ | $3.840 \times 10^{-4}$ | (48, 47): $8.111 \times 10^{-3}$ |
| | | | | | | (48, 49): $9.873 \times 10^{-3}$ |
| 50 | 50.78 | 77.52 | $1.540 \times 10^{-2}$ | $3.866 \times 10^{-4}$ | $3.866 \times 10^{-4}$ | (49, 40): $1.271 \times 10^{-2}$ |
| | | | | | | (49, 50): $1.225 \times 10^{-2}$ |
| | | | | | | (49, 48): $9.873 \times 10^{-3}$ |
| 51 | 63.17 | 77.71 | $2.030 \times 10^{-2}$ | $3.772 \times 10^{-4}$ | $3.772 \times 10^{-4}$ | (50, 51): $7.700 \times 10^{-3}$ |
| | | | | | | (50, 49): $1.225 \times 10^{-2}$ |
| 52 | 64.04 | 84.32 | $1.790 \times 10^{-2}$ | $3.370 \times 10^{-4}$ | $3.370 \times 10^{-4}$ | (51, 50): $7.700 \times 10^{-3}$ |
| | | | | | | (51, 54): $9.585 \times 10^{-3}$ |
| 53 | 63.48 | 56.32 | $3.870 \times 10^{-2}$ | $2.815 \times 10^{-4}$ | $2.815 \times 10^{-4}$ | (52, 56): $2.300 \times 10^{-2}$ |
| | | | | | | (52, 43): 1.000 |
| 54 | 59.29 | 127.72 | $7.600 \times 10^{-3}$ | $3.913 \times 10^{-4}$ | $3.913 \times 10^{-4}$ | (53, 47): $1.097 \times 10^{-2}$ |
| | | | | | | (53, 60): $1.072 \times 10^{-2}$ |
| 55 | 91.25 | 149.31 | $4.720 \times 10^{-2}$ | $3.663 \times 10^{-4}$ | $3.663 \times 10^{-4}$ | (54, 64): $3.441 \times 10^{-2}$ |
| | | | | | | (54, 51): $9.585 \times 10^{-3}$ |
| 56 | 68.43 | 66.34 | $3.920 \times 10^{-2}$ | $1.358 \times 10^{-3}$ | $1.358 \times 10^{-3}$ | (55, 56): $1.884 \times 10^{-2}$ |
| 57 | 81.14 | 16.85 | $1.277 \times 10^{-1}$ | $5.609 \times 10^{-4}$ | $5.609 \times 10^{-4}$ | (56, 55): $1.884 \times 10^{-2}$ |
| | | | | | | (56, 52): $2.300 \times 10^{-2}$ |
| | | | | | | (56, 57): $5.479 \times 10^{-2}$ |
| 58 | 73.34 | 48.86 | $1.028 \times 10^{-1}$ | $7.688 \times 10^{-4}$ | $7.688 \times 10^{-4}$ | (57, 58): $2.336 \times 10^{-2}$ |
| | | | | | | (57, 56): $5.479 \times 10^{-2}$ |

**Table C.1 continued from previous page**

| Q | T1 (us) | T2 (us) | Readout error | $I$ error | $X$ error | `CNOT` error |
|---|---|---|---|---|---|---|
| 59 | 49.11 | 73.57 | $1.150 \times 10^{-2}$ | $3.384 \times 10^{-4}$ | $3.384 \times 10^{-4}$ | (58, 57): $2.336 \times 10^{-2}$ |
| | | | | | | (58, 59): $2.347 \times 10^{-2}$ |
| 60 | 43.26 | 47.66 | $2.980 \times 10^{-2}$ | $2.025 \times 10^{-3}$ | $2.025 \times 10^{-3}$ | (59, 60): $2.470 \times 10^{-2}$ |
| | | | | | | (59, 58): $2.347 \times 10^{-2}$ |
| 61 | 59.66 | 47.73 | $1.290 \times 10^{-2}$ | $4.864 \times 10^{-4}$ | $4.864 \times 10^{-4}$ | (60, 53): $1.072 \times 10^{-2}$ |
| | | | | | | (60, 59): $2.470 \times 10^{-2}$ |
| | | | | | | (60, 61): $1.637 \times 10^{-2}$ |
| 62 | 71.07 | 41.07 | $1.760 \times 10^{-2}$ | $5.540 \times 10^{-4}$ | $5.540 \times 10^{-4}$ | (61, 60): $1.637 \times 10^{-2}$ |
| | | | | | | (61, 62): $2.331 \times 10^{-2}$ |
| 63 | 55.48 | 52.86 | $3.870 \times 10^{-2}$ | $4.784 \times 10^{-4}$ | $4.784 \times 10^{-4}$ | (62, 61): $2.331 \times 10^{-2}$ |
| | | | | | | (62, 63): $1.685 \times 10^{-2}$ |
| 64 | 11.66 | 22.91 | $4.690 \times 10^{-2}$ | $1.122 \times 10^{-3}$ | $1.122 \times 10^{-3}$ | (63, 62): $1.685 \times 10^{-2}$ |
| | | | | | | (63, 64): $3.774 \times 10^{-2}$ |
| 65 | 74.7 | 55.07 | $2.177 \times 10^{-1}$ | $1.601 \times 10^{-3}$ | $1.601 \times 10^{-3}$ | (64, 54): $3.441 \times 10^{-2}$ |
| | | | | | | (64, 63): $3.774 \times 10^{-2}$ |