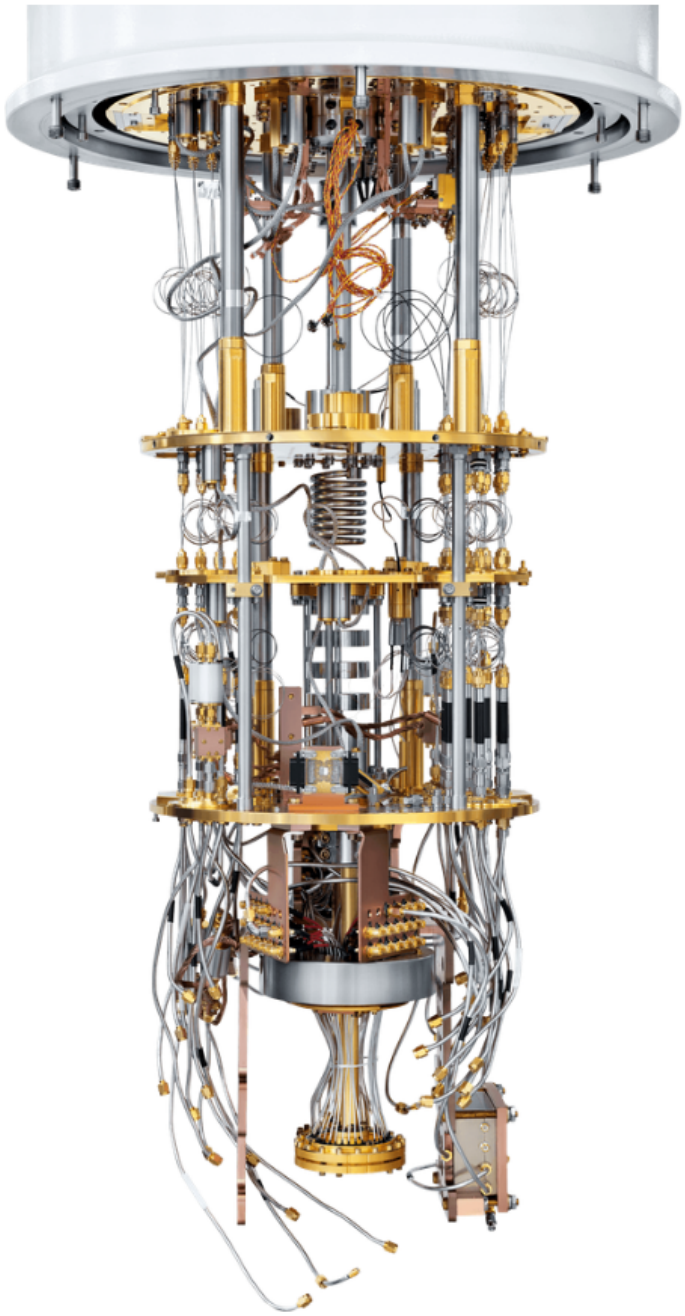
A practical quantum algorithm for solving structural optimization problems: a proof-of-concept!

MSc thesis

J. de Zoete

Technische Universiteit Delft



TU Delft Delft University of Technology

Challenge the future

A PRACTICAL QUANTUM ALGORITHM FOR SOLVING STRUCTURAL OPTIMIZATION PROBLEMS: A PROOF-OF-CONCEPT!

MSc THESIS

by

J. de Zoete

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology,
to be defended publicly on Friday September 10th, 2021 at 09:00 AM.

Supervisors: Dr. B. Chen TU Delft, AE
Dr. M. Möller TU Delft, EEMCS

Cover picture: Rigetti Computing. "Quantum computer." Retrieved from:
https://www.rigetti.com/.

TU Delft Delft University of Technology

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. The cooperation with the Oak Ridge Leadership Computing Facility is gratefully acknowledged.

Delft University of Technology
Faculty of Aerospace Engineering
Department of Aerospace Structures and Materials

GRADUATION COMMITTEE

Dated: September 10, 2021

Chair holder:

Prof. Dr. ir. S. van der Zwaag

Committee members:

Dr. B. Chen

Dr. M. Möller

Dr. ir. M.I. Gerritsma

ABSTRACT

The aerospace engineering industry is continuously striving for faster methods to solve and optimize problems with a higher degree of accuracy. It is therefore of relevance to investigate possibilities to further increase the computational speed at which engineering and research problems are solved. Quantum computing is expected to further increase the computational speed. Therefore, the research objective of this MSc thesis is to investigate the feasibility of optimizing a 2D determinate truss structure with a quantum algorithm run on a Gate-Based Quantum Computer (GBQC).

An extensive literature review concerning several quantum computing algorithms was performed. After carefully trading-off all the examined algorithms, it became apparent that the Quantum Approximate Optimization Algorithm (QAOA), capable of approximating combinatorial optimization problems, is currently one of the more suitable near-term quantum algorithm candidate for this problem, because of its simplicity and robustness.

The most important take-away from this thesis is that it is possible to map a truss structure to QAOA format so that it can be discretely optimized on a GBQC. The program proves to be working on quantum virtual machines. Currently, however it is not possible to obtain correct results when running on real quantum hardware. The reason is that the total noise levels of the real quantum hardware is currently too high for the required gate volume. Hopefully in the future the noise levels for quantum machines will decrease, which will enable people to run more interesting programs.

Contents

1

INTRODUCTION

1.1. MOTIVATION

With quantum computers being in an early stage of development, they exhibit enormous potential. It is expected that quantum computers eventually will solve real-life engineering and research problems that classical computers cannot solve, this is named quantum supremacy *or quantum advantage* [1]. The Noisy Intermediate-Scale Quantum (NISQ) era, that started in 2017, when the first 50-qubit quantum computers became available, could be considered as the stepping stone for the future, where quantum computers (QC) may be able to have a significant effect on society. Quantum computing might become an extra tool in the toolbox of engineers, one day. However, it is uncertain how many years it will take, maybe it will take decades [2] or perhaps it will never have a significant effect on society [3]. In quantum computing nothing is taken for granted which makes it a risky field of research, however it provides also opportunities worth the effort to explore, especially because of the enormous expected potential.

One of the priorities of the aerospace industry is the reduction of aircraft weight to reduce the usage of kerosene and thus costs, whilst complying with the regulations of aviation authorities. This weight reduction can be achieved by different methods, such as exploring different materials and optimizing the design of a structural part (e.g. a wingbox structure).

1.2. RESEARCH QUESTION, AIM/OBJECTIVES

In this thesis work, 2D determinate truss structures are optimized, where the objective is to minimize weight, and the design variables are the discrete cross-sectional area choices, of which the best combination will result in the optimal configuration. The optimization problem is subject to a strength constraint, where the stress in the truss should equal or at least be extremely close to the maximum allowed stress. The truss structures optimized in this work are simple structures, of which the solutions can be easily obtained by classical methods, where various reference solutions can be found in Appendix D. By no means this thesis work tries to show a quantum advantage or quantum supremacy. The objective is rather a proof-of-concept to investigate the feasibility of optimizing a 2D truss structure with a Gate-Based Quantum Computer (GBQC). Wils [4] has investigated the feasibility of optimizing indeterminate structural 2D structures with a Quantum Annealing (QA) computer. QA computers are not considered in this thesis work, since it is yet unknown if they are scalable [2]. It would, however be interesting to see if the work of Wils [4] in its QA format can be mapped to a GBQC format.

Quantum computing may fundamentally change the way scientists, engineers and others solve or optimize problems, as discussed later in the literature review in Chapter 2. This research is conducted as part of the effort to connect the fields of aerospace structural engineering and quantum computing. A 2D truss structure will be optimized. Therefore, the main research objective of this thesis is formulated as:

Investigate the feasibility of minimizing the weight of a 2D truss structure with a quantum algorithm on a GBQC, with discrete cross-sectional areas as design variables subject to a strength constraint.

In order to achieve the research objective, the research question of this thesis is formulated as:

Is it feasible to minimize the weight of a 2D truss structure by varying the cross-sectional areas, subjected to a strength constraint with a quantum algorithm on a GBQC?

The following sub-questions and sub-sub-questions are asked:

1. How can GBQC be utilized to optimize truss structures?
   (a) How to cast the truss problem to GBQC quantum-suitable format?
   (b) How to map a QA problem truss optimization problem from Wils [4] to a GBQC quantum-suitable format?
2. What is the influence on the results when varying some of the parameters of a quantum computer?
   (a) What will be the effect on quantum performance when the noise in quantum computers and quantum gates will be decreased?
   (b) What will be the effect on quantum performance when the number of qubits in quantum computers will be increased?
3. How can the results of the quantum computer best be interpreted?
   (a) How does the probabilistic nature of quantum computers influence the results?
   (b) How can the results from the quantum computer be verified?

1.3. THESIS OUTLINE

This thesis work is composed in five chapters. In this chapter a motivation is given as well as the research objectives and research questions. In Chapter 2 a summary of the literature review can be found which describes various reviewed quantum algorithms. In addition to that, Chapter 2 explains in more detail the two algorithms used in this thesis, the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA). In Chapter 3 the methodology of describing a truss structure discretely is shown. The results of this thesis are presented and discussed in Chapter 4. Finally, in Chapter 5 conclusions and recommendations for future work are given.

2

LITERATURE REVIEW

In Section 2.1 an introduction to quantum terms and symbols are described. A general introduction on quantum computing is given in Section 2.2. A short summary of all the quantum algorithms reviewed in the literature review is shown in Section 2.3. In this thesis the Variation Quantum Eigensolver (VQE) is used as a subroutine of the Quantum Approximate Optimization Algorithm (QAOA) and for this reason both algorithms are explained in more detail in Section 2.4 and Section 2.5 respectively.

2.1. A BRIEF EXPLANATION ON QUANTUM TERMS

A short introduction on the notation used to compare algorithms is shown in the paragraph below. Furthermore, commonly used terms in this literature review are shown in Table 2.1, it can be used as an extremely short quantum dictionary. For more information about quantum computation and quantum information, the reader is referred to Nielsen and Chuang [5].

SYMBOLS AND LETTERS USED TO COMPARE ALGORITHMS

In order to be able to evaluate the performance of different algorithms, several parameters have to be compared. The letters and symbols N, s, κ, λ and ϵ described below are used in order to compare runtimes of algorithms. The parameter N defines the number of entries in an arbitrary matrix $\mathbf{A}$ (N x N entries), where λ represents the eigenvalues of the matrix $\mathbf{A}$, s is the maximum number of non-zero entries in a row in a N x N matrix. If a matrix is symmetric and positive-definite, the condition number κ is then defined by $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$ and describes the conditioning of the matrix, (i.e. a large κ is associated with an ill-conditioned matrix) and finally ϵ is the desired precision [6].

Table 2.1: Quantum words with explanation

quantum word	**explanation**
ancillary *or ancilla* quantum bit	Auxiliary *or extra* quantum bit to facilitate the computation, however the output of this ancilla quantum bit is not required for the rest of the circuits computation.
ansatz	Educated guess *or initial estimate*.
circuit depth	The maximum length of time steps *or gates* of a single qubits path, when considering all qubits, from its circuits input to output.
circuit size	Total number of quantum gates, excluding the identity gates (for more information about gates refer to Table A.1), in a quantum circuit [7].
circuit width	The number of quantum bits used in a circuit.

complex conjugate	Has equal real part as the original function and the imaginary part is equal in magnitude, however opposite in sign. For example, the complex conjugate of 3 + 2i is 3 - 2i.
Hilbert space	Finite dimensional complex vector space & inner product space.
eigenstate	Eigenvector in a quantum state in Hilbert space.
expectation value	Average value. In literature the expectation value is referred to as the average value [5, 8]. It is not the most expected or probable value.
Hadamard gate	Quantum gate that puts qubits in superposition, named after Jacques Hadamard, refer to Appendix A for more information.
Hamiltonian	Energy of the system, an operator consisting of kinetic and potential energies.
Hermitian matrix[1]	A complex square matrix equivalent to its conjugate transpose $A^\dagger = (A^T)^*$. $A = \begin{bmatrix} a & b+1i. \\ b-1i. & c \end{bmatrix} = A^\dagger = \begin{bmatrix} a & b+1i. \\ b-1i. & c \end{bmatrix}$
hybrid algorithm	Algorithm that makes use of both classical and quantum computing. The words variational and hybrid are used interchangeably.
Pauli gates	Quantum gates that consist of a unitary matrix originally discovered by Wolfgang Pauli. For more information, refer to Appendix A.
Decision problem	Problem that can either result in a no or yes answer. It can be represented alternatively as a boolean problem that can either be zero or one.
determinism	Certainty, no randomness involved. A deterministic algorithm will for a given initial input always produce the same output.
nondeterminism	Uncertainty, randomness involved. A nondeterministic algorithm can for a given initial input produce different outputs.
P problems	Problems that are solvable in polynomial time.
NP problems	Decision (boolean) problems that are solvable in nondeterministic polynomial time, where the solution yes or one is verifiable in polynomial time.
NP complete problems	NP complete problems are problems that are intersecting the hardest problem(s) in the NP set and the easiest problem(s) in the NP-hard set, so: X ∈ NP and X ∈ NP-hard.
NP hard problems	Problems that are at least as hard as the hardest problem(s) in NP, where every problem Y ∈ NP reduces to X. If a problem Y reduces to problem X, it means that X is at least as hard as Y. So, if X ∈ NP then Y ∈ NP and if X ∈ P then Y ∈ P. Alternatively, one could say that if an algorithm for solving X can solve any NP-problem Y then X is NP-hard.
quantum bit or *qubit*	n qubits can simultaneously encode 2^n states of information
quantum gates	Gates used in quantum circuits.
quantum circuit	The major difference between an electrical circuit and a quantum circuit is, that with a logical circuit it is often impossible to retrieve the inputs after applying the gates. It is possible to reverse the quantum circuit and retrieve the initial states of the qubits, if no measurement has taken place, because the information is still encoded in the quantum state.
unitary matrix[1]	A matrix is unitary if and only if $B^\dagger$B = I and $B^{-1}B$ = I, meaning that the determinant has to be equal to ±1. For example: $B = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = B^\dagger = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$, $B^{-1} = \frac{1}{-1}\begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix}$ $B^\dagger B$ = I and $B^{-1}B = \frac{1}{-1}\begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ = -1 $\begin{bmatrix} i^2 & 0 \\ 0 & i^2 \end{bmatrix}$ = I

2.2. A GENERAL INTRODUCTION TO QUANTUM COMPUTING

In this section the quantum computing topic is introduced in a concise manner. If the reader would like to learn more about quantum computing and quantum information in general, the reader is

[1] For more information review: https://www.math.purdue.edu/~eremenko/dvi/lect3.26.pdf.

referred to Nielsen and Chuang [5].

With quantum computers being in an early stage of development, they exhibit a lot of potential. It is expected that quantum computers eventually will solve real-life engineering and research problems that classical computers cannot solve, which is called quantum supremacy [1]. Google researchers claimed to have achieved quantum supremacy [9]. This claim was immediately opposed by IBM researchers [10] and others [11], whom stated it should be named quantum advantage instead. Nevertheless, to be able to use quantum computers for real-life applications, controllability issues and decoherence of qubits [3] should be solved. Controllability issues occur by increasing the number of qubits *or degrees of freedom (DoF)* in a quantum computer. It can be compared to controllability issues experienced when increasing the number of wheels of a bicycle from 2 to 2^{1000} [3]. Quantum decoherence is, as the name implies, the decoherence *or collapse* of the superposition of the qubit state in quantum computers (QC), incurred by interaction with the environment, into its 0 or 1 state [12].

To counteract the effect of quantum decoherence, Shor [13] and Steane [14] independently came up with a scheme for quantum error correction. Thereafter, Preskill [15] developed the concept for a fault-tolerant quantum computer, that proves that if quantum decoherence is weak, it can be corrected. Eventually, it is expected that quantum systems and many-qubit QC are able to protect qubits from decoherence [16] by quantum error correction, however this requires a significant amount of additional physical qubits [2].

In theory, quantum annealers (QA) can exhibit equivalent properties as a Gate-Based Quantum Computer (GBQC), when additional physical qubits are considered to enforce robustness by diminishing noise [17]. Currently, however research does not provide good arguments to expect that quantum annealers are scalable, opposite to GBQC [2]. Furthermore, it is unknown if QA are able to function properly as the problem size increases [2]. For these reasons, quantum annealing computers are not considered in this thesis. It would, however be interesting to investigate the possibility of mapping Wils [4] his QA work to GBQC format.

The Noisy Intermediate-Scale Quantum (NISQ) era, which is the era where GBQC consist of fifty to a few hundred qubits [2]. GBQC are quantum computers that run quantum circuits, consisting of quantum gates. Currently, 50-qubit QC and more cannot be simulated by classical super computers, because the computational cost doubles with every added qubit and thus increases exponentially [1]. This minimum of 50-qubit QC is not a set in stone theoretical limit, it also depends on: the depth of the circuit; number of qubits; new algorithms and the increase in classical computational power in the coming years [1, 18, 19]. Classical computers cannot simulate the dynamics of many-qubit quantum computers, therefore quantum computers might unveil a new field of research [2].

2.3. REVIEW OF QUANTUM ALGORITHMS

In this section a summary of the reviewed quantum algorithms of the literature review are shown.

The Quantum Fourier Transform (QFT) is used to transform data from one domain to the frequency domain. Quantum Phase Estimation (QPE) is used to calculate eigenvalues of a certain matrix $\mathbf{A}$ as explained in Nielsen and Chuang [5]. A quantum algorithm that is used to determine the lowest eigenvalue of an arbitrary matrix $\mathbf{A}$ is named Variational Quantum Eigensolver (VQE) [20]. The Quantum Approximate Optimization Algorithm (QAOA) is a quantum algorithm similar to VQE, only applied to a different type of problem, namely an optimization problem [21]. QAOA is a shal-

low quantum algorithm, meaning that it has low circuit depth and therefore will exhibit less noise. Alpha-VQE [22] is an algorithm used to operate in-between the regimes of QPE and VQE to take the best parts of both methods preventing unmanageable circuit depth when using QPE and preventing the potential of unmanageable runtimes when using VQE. A quantum algorithm named after Harrow, Hassidim and Lloyd (HHL) [23], could in theory exponentially increase the speed at which linear systems of equations are 'solved'. A versatile quantum algorithm, similar to HHL, able to 'solve' linear systems of equations on NISQ computers is named Variational Quantum Linear Solver (VQLS)[24]. Both VQLS and HHL have a major caveat, which is that the algorithm could only solve one entry of the solution vector x, meaning that if N entries are required to be solved, the algorithm has to be rerun N times. Adiabatic QC evolves in time described by the Schrödinger equation used in quantum annealing computers Farhi *et al.* [25]. The evolution randomization algorithms proposed by Subaşl *et al.* [26] are simple variants of adiabatic quantum computing. They have an almost linear time complexity in κ and low circuit depth, meaning that it is an efficient algorithm. Both these algorithms proposed by Subaşl *et al.* [26] can be implemented on a GBQC via Hamiltonian simulation. Quantum Semi-Definite Programming (qSDP) [27] is an optimization algorithm that generalizes linear and quadratic programming problems and combinatorial optimization problems by optimizing objective functions. A quantum algorithm capable of analyzing the principal components (eigenvalues) of matrices quantum mechanically, enabling the potential of solving bigger matrices is named Quantum Principal Component Analysis (qPCA) [28, 29]. Classifying data in data sets can be achieved by Quantum Support Vector Machines (qSVM) [30]. Quantum Neural Networks (QNN) are able to minimize cost functions and will be most powerful when sufficient training data is available and utilized, further explained in Murphy [31]. Quantum Natural Gradient Descent (QNGD) minimizes a cost function and progresses along direction of steepest descent or with Newton's method, for more information see Rebentrost *et al.* [32].

In order to optimize the 2D truss structure it should be expressed in an objective function so that it can be minimized. Therefore, the algorithms: QFT; QPE; α-VQE; qPCA and qSVM are dropped from the list of possible quantum algorithms to solve this truss problem, since these algorithms are utilized for different purposes. HHL cannot be run without error correction on NISQ computers, therefore it has been dropped as well. Considering the fact that VQLS (and HHL) have a major caveat which is that both algorithms could only solve one entry, thereby it has been dropped as a feasible algorithm. Due to the lack of quantum resources to encode sufficient training data for various truss structures the potential of QNN's cannot be fully utilized and therefore QNN is dropped as a feasible algorithm. This leaves QAOA, the evolution randomization algorithms, qSDP and QNGD as possible algorithms. These four quantum algorithms have been carefully considered. The simplest and most robust (least noisy) algorithm was chosen, namely QAOA, to optimize the truss structure, where VQE was used as a subroutine to calculate the expectation.

2.4. VARIATIONAL QUANTUM EIGENSOLVER (VQE)

In this section the Variational Quantum Eigensolver (VQE) algorithm is explained in more detail, since it has been used as a subroutine of QAOA in this thesis.

The quantum algorithm able to determine the lowest eigenvalue with a small quantum circuit was found in 2013 by Peruzzo *et al.* [20] is named Variational Quantum Eigensolver (VQE). A short summary is described in this section. It is particularly useful in chemistry, material science, pharmaceutical industry and for internet search engines. It is an algorithm that uses the variational method *or variational principle* [8] to solve eigenvalues. It consists of two algorithms first ran on a quantum processing unit, calculating the expectation value of a Hamiltonian ($\mathscr{H}$) and then ran on a classical processing unit with an optimization algorithm to compute the eigenvalues and eigenvectors of $\mathscr{H}$.

The first part of the algorithm ran on the QC is named Quantum Expectation Estimation (QEE). It starts with a given Hamiltonian $\mathscr{H}$, polynomially scaling with the size of the system, for an ansatz input state $|\psi(\theta)\rangle$, where θ is a changing parameter. $\mathscr{H}$ can be written for real h constants, where the superscripts describe the subspace spanned by the eigenvectors on which the Pauli operators act and the subscripts describe the different Pauli operators used:

$$\mathscr{H} = \sum_{i\alpha} h^i_\alpha \sigma^i_\alpha + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \sigma^i_\alpha \sigma^j_\beta + \dots \tag{2.1}$$

Subsequently, by using the linearity of quantum observables it can be shown that[20]:

$$\langle \mathscr{H} \rangle = \sum_{i\alpha} h^i_\alpha \langle \sigma^i_\alpha \rangle + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \langle \sigma^i_\alpha \sigma^j_\beta \rangle + \dots \tag{2.2}$$

where the $\langle\ \rangle$ stands for the expectation value. Rewriting Equation (2.2) results in the ground state wave function [8]:

$$E_{ground} \leq \langle \psi(\theta)|\mathscr{H}|\psi(\theta)\rangle = \sum_{i\alpha} h^i_\alpha \langle \psi(\theta)|\sigma^i_\alpha|\psi(\theta)\rangle + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \langle \psi(\theta)|\sigma^i_\alpha \sigma^j_\beta|\psi(\theta)\rangle + \dots \tag{2.3}$$

In the first part of the algorithm the ground state wave function is efficiently well approximated [20]. The second part of the algorithm, is a variational method that exists to compute the ground state eigenvalue. This is achieved by using the Rayleigh-Ritz quotient to normalize the ground state wave function, known as the variational principle [8]:

$$E_0 \leq E_\theta = \frac{\langle \psi(\theta)|\mathscr{H}|\psi(\theta)\rangle}{\langle \psi(\theta)\ |\ \psi(\theta)\rangle} \tag{2.4}$$

by varying θ to iteratively optimize for E_θ, which minimizes the right-hand side of the equation with the following condition $E_0 \leq E_\theta$, where E_0 is the smallest eigenvalue of $\mathscr{H}$.

VQE provides an exponential speedup compared to classical methods [20]. Classical methods have a restricted polynomially scaled ansatz, due to memory limitations, whereas quantum circuits can (theoretically) consist of an exponential number of (entangled) states [33]. QPE uses a number of iterations of $\mathscr{O}(1)$ and a circuit depth of $\mathscr{O}(\frac{1}{\epsilon})$. VQE uses a number of iterations of $\mathscr{O}(\frac{1}{\epsilon^2})$ with ϵ being the precision parameter and a circuit depth of $\mathscr{O}(1)$ experimentally first demonstrated by O'Malley *et al.* [33] and later by Kandala *et al.* [34] for a larger molecule up to BeH_2. VQE is an iterative algorithm able to compute minimal eigenvalues of larger molecules [34], than what is currently experimentally demonstrated with QPE [35–38]. Furthermore, VQE is robust against systematic device errors, because VQE requires only short state preparation and measurement sequences [33]. A potential downside for VQE is that a large number of samples is required by the quantum expectation estimation algorithm subroutine of the VQE algorithm, which may lead to unmanageable runtimes [22]. A similar method to VQE using the same approach however, using different Hamiltonians and applied to a different type of problem [2, 39], is the Quantum Approximate Optimization Algorithm (QAOA), explained in the next section.

2.5. QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM (QAOA)

A quantum algorithm for combinatorial optimization problems to find the extremum, that starts with an ansatz, is named Quantum Approximate Optimization Algorithm (QAOA) and was initially proposed by Farhi *et al.* [21]. The QAOA is essentially an algorithm to approximate the solution of a combinatorial optimization problem.

The QAOA is seen as a leading heuristic candidate [40] in quantum computing, because of its simplicity and robustness[2]. In 2018 there was also criticism by Guerreschi and Matsuura [41] that quantum speedups will not be achieved for a real-life combinatorial problem with less than a few hundred of qubits. Contrary to this criticism by Guerreschi and Matsuura [41] the regime to challenge the best classical algorithm was met by Zhou *et al.* [42] in the year 2019. Also, Zhou *et al.* [42] proposed two heuristic strategies, INTERP and FOURIER, to let the algorithm choose improved initial guesses for varying parameters. The QAOA is already implemented as an algorithm applied to a simplified real world aircraft assignment problem known as the tail assignment problem [43]. The QAOA is also explored for other (NP-)hard linear algebra problems [44]. An advantage of the QAOA is that it is a shallow quantum algorithm, meaning that the circuit depth is small compared to other quantum algorithms. For this reason QAOA is more robust against noise and decoherence. Furthermore, the QAOA is robust against certain systematic errors [41].

It is mentioned by Verdon *et al.* [45] that the QAOA can be a discretized alternative of quantum annealing process, however it is not a requirement. The QAOA is able to find the approximate solution in a shorter time/depth than annealing [44]. Another interesting development is a proposed generalization of the work of Farhi *et al.* [21] by Hadfield *et al.* [40], renaming it to Quantum Alternating Operator Ansatz (QAOA) with the same acronym. In Bravo-Prieto *et al.* [24] it is suggested that the QAOA can be utilized as an ansatz of the Variational Quantum Linear Solver (VQLS) algorithm.

After reading these articles it became apparent that almost all authors propose for more research to be done in the research field of the QAOA to solve combinatorial optimization problems. The question remains if the QAOA will provide quantum advantage over classical algorithms. The answer is currently unknown.

A short summary of the QAOA in its original form by Farhi *et al.* [21] is described below:
The algorithm consists of two Hamiltonians C and B with 2p parameters $\gamma_1,\ldots,\gamma_p,\beta_1,\ldots,\beta_p$ acting as time operators, where p $\in \mathbb{N}$.

The objective or cost function C(z), defined with n bit strings and m clauses, also often written as cost Hamiltonian H_C, is shown here:

$$C(z) = \sum_{\alpha=1}^{m} C_\alpha(z), \tag{2.5}$$

where z = $z_1 z_2 ... z_{n-1} z_n$ and $C_\alpha(z) = \begin{cases} 1, & \text{if bit string z satisfies clause } \alpha \\ 0, & \text{otherwise.} \end{cases}$

Alternatively C(z) can be represented as:

$$C(z) = \begin{bmatrix} C_\alpha(0,\ldots,0) & & \\ & \ddots & \\ & & C_\alpha(1,\ldots,1) \end{bmatrix} \tag{2.6}$$

It is combined with angle $\gamma \in \mathbb{R}$, restricted to the domain $[0, 2\pi]$, because of integer eigenvalues. All terms commute $[C_\alpha(0,\ldots,0), C_\alpha(1,\ldots,1)] = 0$, meaning that there is independence of order, because of diagonality in the computational $|0\rangle$ and $|1\rangle$ basis. Cost function C is written as a unitary operator U(C, γ):

$$U(C,\gamma) = e^{-i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}. \tag{2.7}$$

[2] Peter Shor, Quantum Approximate Optimization Algorithms, ISCA 2018 Tutorial: Grand Challenges and Research Tools for Quantum Computing, University of Chicago, https://www.epiqc.cs.uchicago.edu/s/QAOA-talk.pdf

The operator B *or often referred to as mixer Hamiltonian* H_B is shown here:

$$B = \sum_{j=1}^{n} \sigma_j^x, \tag{2.8}$$

where σ_j^x is the Pauli X operator acting on the j'th qubit. Combined with angle $\beta \in \mathbb{R}$, restricted to the domain $[0, \pi]$, operator B is written as a unitary operator U(B, β):

$$U(B, \beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta\sigma_j^x} \tag{2.9}$$

The initial state $|s\rangle$ is a uniform superposition of the computational basis states:

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle \tag{2.10}$$

A quantum state is made and parameterized by alternating β_p and γ_p for 1 to p rounds, obtainable with quantum circuit with maximum depth of mp + p, where B and C should not commute [B, C] $\neq$ 0:

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = U\left(B, \beta_p\right) U\left(C, \gamma_p\right) \cdots U\left(B, \beta_1\right) U\left(C, \gamma_1\right) |s\rangle \tag{2.11}$$

or represented alternatively:

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = e^{-i\beta_p H_B} e^{-i\gamma_p H_C} \cdots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |s\rangle \tag{2.12}$$

The expectation state can be obtained after repeating the above steps:

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) | C | \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle \tag{2.13}$$

Where M_p is the maximum of F_p over all the angles, where $M_p \geq M_{p-1}$:

$$M_p = \max_{\gamma, \beta} F_p(\gamma, \beta) \tag{2.14}$$

Finally, it can be proven that the maximum of C(z) is found when the limit of p to infinity is taken over M_p:

$$\lim_{p \to \infty} M_p = \max_z C(z) \tag{2.15}$$

The approximation ratio r is:

$$r = \frac{C(z)}{C_{max}} \tag{2.16}$$

In general QAOA is symmetric, and thus reversible in time, $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = F_p(-\boldsymbol{\gamma}, -\boldsymbol{\beta})$, due to the mixer and cost Hamiltonian being real-valued [42].

The Quantum Alternating Operator Ansatz (QAOA) was proposed to generalize the Quantum Approximate Optimization Algorithm [40]. It allows for different mixer Hamiltonians then the initially proposed mixing operator B = σ_j^x by Farhi *et al.* [21], where the the full search space was explored. It is possible to restrict this full search space to a feasible subspace, where only feasible/valid solutions are allowed. Therefore, other mixer Hamiltonians are introduced such as $H_{j,k} = \frac{1}{2}(X_j X_k + Y_j Y_k)$, where this mixer restricts the algorithm to only allow swapping of qubits j and k, further explained in Appendix B.3. This generalized version of QAOA can be useful for optimization problems that have to satisfy hard constraints [40].

2.5.1. GRAPHICAL EXPLANATION OF QAOA

The theory of the Quantum Approximate Optimization Algorithm (QAOA) has been explained in Section 2.5. For illustrative purposes the algorithm is shown in Figure 2.1 in its initial form by Farhi *et al.* [21]. In this subsection the QAOA is explained in a graphical way to provide the reader with an extra alternative explanation.

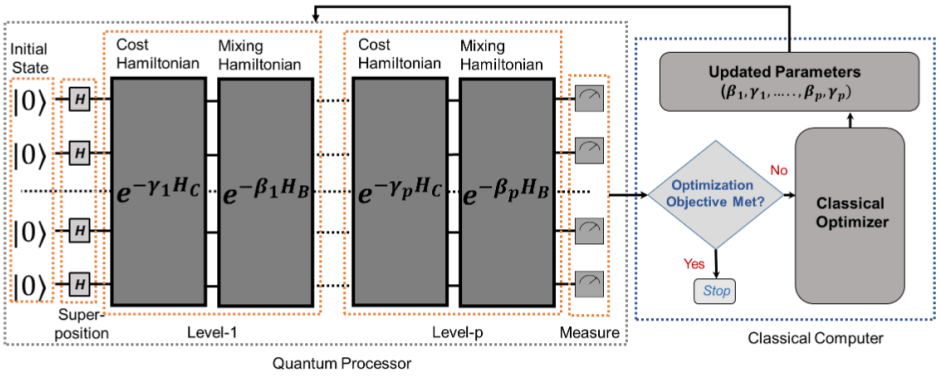


Figure 2.1: General overview of the QAOA algorithm. Figure retrieved from Alam *et al.* [46].

Figure 2.1 is explained from left to right in this paragraph: at first the initial state $|s\rangle$ is prepared in superposition by applying one Hadamard gate to each qubit, which are initially all in the $|0\rangle$ state. After which the cost unitary operator $U_C = e^{-i\gamma_1 H_C}$ is applied, where H_C is the cost Hamiltonian encoding the objective function and γ is a parameter to be optimized. Subsequently, the mixer unitary operator $U_M = e^{-i\beta_p H_B}$ is applied, where H_B is the mixer Hamiltonian, which mixes the states, and β is a parameter to be optimized. This is done with p-steps until level-p in order to amplify the probability of measuring the correct outcome. Finally, the quantum state is measured into a classical bit string register. This whole identical process is run with n shots pre-defined by the user, to obtain a distribution of n counts (bit strings). The classical computer then checks the expectation value with Equation (2.18). If the optimization objective is not met, then the expectation value is fed into the classical optimizer, after which β_1, γ_1, ..., β_p, γ_p are updated and fed back into the quantum processer until convergence is met. In equation form Figure 2.1 is summarized in Equation (2.17) and Equation (2.18).

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = e^{-i\boldsymbol{\beta}_p H_B} e^{-i\boldsymbol{\gamma}_p H_C} \ldots e^{-i\boldsymbol{\beta}_1 H_B} e^{-i\boldsymbol{\gamma}_1 H_C} |s\rangle \tag{2.17}$$

The expectation state can be obtained after repeating the above steps with n shots:

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) | C | \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle \tag{2.18}$$

The goal is to optimize for the angles that maximize the expectation state F_p. A practical example problem, the Max-Cut problem, to show the theory above in practice, can be found in Appendix C.

3

METHODOLOGY

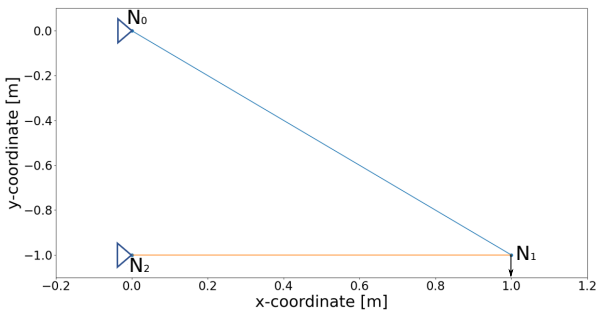
In this chapter the methodology of optimizing a determinate 2D truss structure is explained. Firstly, a general introduction to the approach of optimizing a truss structure is given in Section 3.1. The 2D truss problem is then casted into Quantum Alternating Operator Ansatz (QAOA) format for the three-member truss structure in Section 3.2. Subsequently, an explanation on how the expectation is calculated is described in Section 3.3. A detailed explanation about the hardware, including the connectivity of qubits, noise and computational speed of the quantum algorithm is given in Section 3.4. Finally, Section 3.5 will elaborate on translating a Quantum Annealing problem to QAOA format. Finally, the influence of the step parameter p on the gate volume is described in Section 3.6.

3.1. A GENERAL INTRODUCTION TO THE TRUSS PROBLEMS

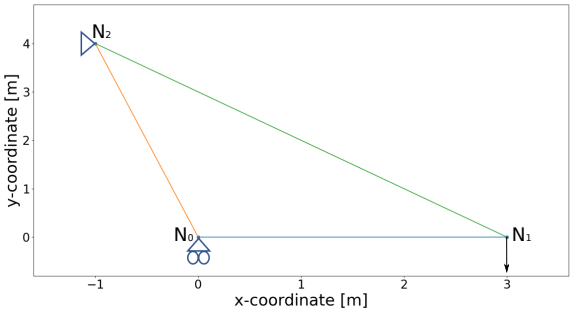
One of the objectives in aerospace engineering is to minimize weight and thus kerosene and costs. The objective of this thesis is to investigate the feasibility of optimizing a truss structure with a quantum algorithm on a GBQC. To the best of the author's knowledge, quantum optimization for trusses has only been done once before with parameterized FEM by using Quadratic Unconstrained Binary Optimization (QUBO) for Quantum Annealing (QA) computers by Wils [4]. In this thesis the focus will be on describing determinate truss structures with a parameterized internal force method and subsequently optimize them with a quantum-suitable algorithm on a GBQC.

In classical computing it is rather easy to represent a number in binary form, since the number of bits generally is not a limiting factor. In quantum computing this quickly becomes unfeasible, because there is a limited number of quantum bits, named qubits hereafter, in quantum computers. There are 32 or 64 (qu)bits required to represent only a number with the floating point number scheme with single or double precision respectively. To avoid the use of many qubits to represent numbers, the qubits are assigned more efficiently to a discrete set of cross-sectional area choices.

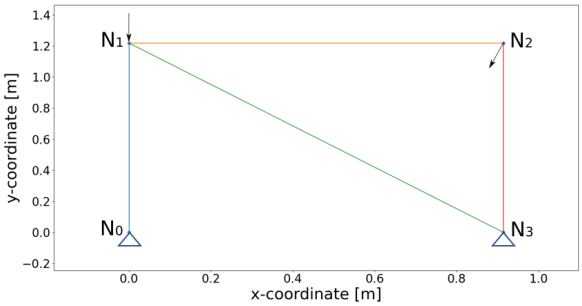
In this thesis three determinate structural truss problems were analyzed, a two-member, three-member and four-member truss structure, as can be observed in Figure 3.1a, Figure 3.1b and Figure 3.1c respectively.



(a) Two-member truss problem



(b) Three-member truss problem



(c) Four-member truss problem

Figure 3.1: 2D determinate truss structures to be solved

Three different area options are assigned to each truss member as can be observed in Figure 3.2. To be clear here, the blue curved truss member A_2 does not represent a curved truss member, however it represents $A_0 + dA$. Indexing in this thesis starts from zero instead of one for programming reasons. In between node N1 and N2, three discrete area options can be chosen for ($A_0 - dA$, A_0 and $A_0 + dA$), where A_0 represents the area for truss member one with index zero and dA is a predefined constant increment of A. It is convenient to relabel $A_0 - dA$, A_0 and $A_0 + dA$ to A_0, A_1 and A_2 respectively. For the truss structures with three options per truss member, all choices of cross-sectional areas A can be captured with Equation (3.1):

$$A = \left\{ Aq \;\middle|\; A_{3i+j} q_{3i+j} \right\}, \tag{3.1}$$

where i cycles through 0 to # trusses - 1, and j cycles through 0 to # options - 1, q $\in \{0, 1\}$ and $\sum_{j=0}^{opt-1} q_j = 1$, where opt represents the number of options per truss member. As an example the set A for the three-member truss structure is shown:

$$A_{3-truss} = \left\{ Aq \;\middle|\; A_{3i+j} q_{3i+j} \right\} = \{A_0 q_0, A_1 q_1, \ldots, A_8 q_8\} \tag{3.2}$$

It is possible to include more or less cross-sectional area choices per truss member. Having two choices per truss member would result in a cross-sectional area A and either a A+dA choice or a A-dA choice, which would have to be alternated, which is impractical. Furthermore, having more than three choices per truss member would result in a larger circuit depth and circuit width, increasing the noise levels. Considering the above, it has been deemed most reasonable to have three choices per truss member and not less or more.

In this simple truss structure the weight of the structure is minimized by varying the area options discretely to find a local minimum. As an example, the optimal cross-sectional area of truss member

zero is $9.0 \cdot 10^{-4} m^2$, however only options $1.0 \cdot 10^{-3} m^2$, $1.1 \cdot 10^{-3} m^2$ and $1.2 \cdot 10^{-3} m^2$ for truss member zero are given. The algorithm will then (most probably) choose the truss member with an area of $1.0 \cdot 10^{-3} m^2$ as the local minimum. In order to find the optimum $9.0 \cdot 10^{-4} m^2$ for truss member zero, the areas have to get updated and iterated. The other truss members have to get minimized as well until the global optimum is found, meaning that the residual ≈ 0.

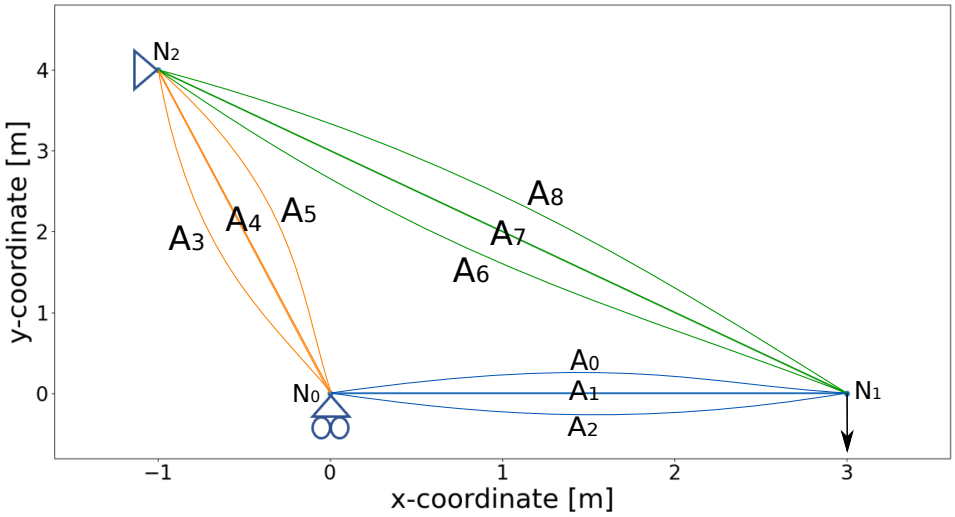


Figure 3.2: Three options per truss member for three-member truss structure.

If all options would be considered in this three-member truss structure there would be a total of $\mathcal{O}(2^{(\#trusses \cdot \#options)}) = 2^9 = 512$ many options. Feasible/valid solutions are solutions that only choose one cross-sectional area per truss member ($\sum_{j=0}^{opt-1} q_j = 1$), where opt is the number of options per truss member. When only the valid bit string states would be considered in this three-member truss structure, it results in a total of $\mathcal{O}(\#\text{trusses}^{\#\text{options}}) = 3^3 = 27$ valid options, shown in Table 3.1.

As the number of trusses and options grow, the difficulty of optimizing the truss structure increases exponentially in time or resources. That is where quantum comes into play. Quantum computers can investigate all choices simultaneously, however noise requires to repeat the experiment with n shots, leading to a distribution of counts.

The configuration values used for the two, three and four-member truss structures are shown in Table 3.2. Initially the internal forces are assumed to be in tension. In this work it is assumed that the material has equal maximum tensile and compressive stress, i.e. $\sigma_c = \sigma_t = \sigma_{\{c,t\}}$. In the configuration table it can be observed that there is a sign bit 1 or -1, where 1 indicates tension and -1 indicates compression. The four-member truss structure was an example problem in a textbook that used English Engineering units and these units were converted to SI units. For this reason "odd" numbers for the forces Fx and Fy can be observed in the configuration table for the four-member truss system. Two classical ways of obtaining the areas for the 3-member truss that minimize the weight are shown in Appendix D. The areas found by the reference solutions that minimize the weight for the 2-truss, 3-truss and 4-truss structures are shown in Table 3.3.

Table 3.1: Valid bit string states for three-member truss system

Option	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
1	0	0	1	0	0	1	0	0	1
2	0	0	1	0	0	1	0	1	0
3	0	0	1	0	0	1	1	0	0
4	0	0	1	0	1	0	0	0	1
5	0	0	1	0	1	0	0	1	0
6	0	0	1	0	1	0	1	0	0
7	0	0	1	1	0	0	0	0	1
8	0	0	1	1	0	0	0	1	0
9	0	0	1	1	0	0	1	0	0
10	0	1	0	0	0	1	0	0	1
11	0	1	0	0	0	1	0	1	0
12	0	1	0	0	0	1	1	0	0
13	0	1	0	0	1	0	0	0	1
14	0	1	0	0	1	0	0	1	0
15	0	1	0	0	1	0	1	0	0
16	0	1	0	1	0	0	0	0	1
17	0	1	0	1	0	0	0	1	0
18	0	1	0	1	0	0	1	0	0
19	1	0	0	0	0	1	0	0	1
20	1	0	0	0	0	1	0	1	0
21	1	0	0	0	0	1	1	0	0
22	1	0	0	0	1	0	0	0	1
23	1	0	0	0	1	0	0	1	0
24	1	0	0	0	1	0	1	0	0
25	1	0	0	1	0	0	0	0	1
26	1	0	0	1	0	0	0	1	0
27	1	0	0	1	0	0	1	0	0

Table 3.2: Configuration values for the two, three and four-member truss structures, including material, nodal coordinates, force and boundary conditions with dA = $100 \cdot 10^{-6} m^2$.

	2-member truss structure						3-member truss structure						4-member truss structure					
Item	Coordinates		Force		BC		Coordinates		Force		BC		Coordinates		Force		BC	
Nodes	x[m]	y[m]	Fx[N]	Fy[N]	dx[m]	dy[m]	x[m]	y[m]	Fx[N]	Fy[N]	dx[m]	dy[m]	x[m]	y[m]	Fx[N]	Fy[N]	dx[m]	dy[m]
N0	0	0			0	0	0	0					0	0			0	0
N1	1	-1	0	-70000			3	0	0	-2000		0	0	1.219	272.16			
N2	0	-1			0	0	0	-1			0	0	0.914	1.219	181.44	314		
N3													0.914	0			0	0
Item	Connected		Material	Areas			Connected		Material	Areas			Connected		Material	Areas		
Elements	Start	End	$\sigma_{\{c,t\}}[Pa]$	sign		$A[m^2]$	Start	End	$\sigma_{\{c,t\}}[Pa]$	sign		$A[m^2]$	Start	End	$\sigma_{\{c,t\}}[Pa]$	sign		$A[m^2]$
E0	N0	N1	$100 \cdot 10^6$	1		$900 \cdot 10^{-6}$	N0	N1	$5 \cdot 10^6$	-1		$510 \cdot 10^{-6}$	N0	N1	$5 \cdot 10^6$	-1		$400 \cdot 10^{-6}$
E1	N1	N2	$100 \cdot 10^6$	-1		$600 \cdot 10^{-6}$	N0	N2	$5 \cdot 10^6$	1		$1500 \cdot 10^{-6}$	N1	N2	$5 \cdot 10^6$	-1		$236.29 \cdot 10^{-6}$
E2							N1	N2	$5 \cdot 10^6$	1		$260 \cdot 10^{-6}$	N1	N3	$5 \cdot 10^6$	1		$260.48 \cdot 10^{-6}$
E3													N2	N3	$5 \cdot 10^6$	-1		$62.851 \cdot 10^{-6}$

Table 3.3: Optimized areas for two, three and 4-member truss structure

	2-member truss structure		3-member truss structure		4-member truss structure	
	Areas		Areas		Areas	
Elements	sign	A[m^2]	sign	A[m^2]	sign	A[m^2]
E0	1	$989.95 \cdot 10^{-6}$	-1	$400 \cdot 10^{-6}$	-1	$102.8 \cdot 10^{-6}$
E1	-1	$700 \cdot 10^{-6}$	-1	$1649 \cdot 10^{-6}$	-1	$36.29 \cdot 10^{-6}$
E2			1	$565.7 \cdot 10^{-6}$	1	$60.48 \cdot 10^{-6}$
E3					-1	$62.85 \cdot 10^{-6}$

3.1.1. CLASSICALLY OPTIMIZING THE TRUSS STRUCTURE CONTINUOUSLY

There are several ways of obtaining the minimal weight configuration of truss structures. Two approaches can be found in Appendix D as reference solutions. Another approach of optimizing the truss members, described in this subsection, would be to assume that all truss members experience the maximum allowable stress. Furthermore, the equilibrium equations are enforced as outcomes of an optimization in order to avoid the fractional part of the equation, since it is not possible to evaluate fractions in GBQC. In this thesis it is assumed for simplicity that both maximum tensile stress σ_t and compressive stress σ_c equal each other, $\sigma_t = \sigma_c = \sigma_{max}$. Setting the sum of forces in x and y equal to the residual equations results in Equation (3.3).

$$Res_{1,x} = \Sigma F_{1,x} \quad Res_{1,x} = \Sigma F_{2,x} \quad Res_{2,y} = \Sigma F_{2,y} \tag{3.3}$$

Then by minimizing the squared residuals to zero or to a value extremely close to zero, it is possible to reach the maximum stress state for all truss members, while enforcing the equilibrium equations, meaning the structure is optimized for weight:

$$\text{Objective } J = min((Res_{1,x})^2 + (Res_{2,x})^2 + (Res_{2,y})^2) \tag{3.4}$$

Minimizing the residuals of the equations in matrix form, which is the equivalent of what was shown in Equation (3.4):

$$M \times \vec{F_{int}} = \vec{F_{ext}} \rightarrow min||(M \times \vec{F_{int}} - \vec{F_{ext}})||^2 \tag{3.5}$$

Using the same matrix as obtained in Equation (D.1): $\begin{bmatrix} 1 & -.243 & 0 \\ -1 & 0 & -.707 \\ 0 & 0 & -.707 \end{bmatrix} \times \begin{bmatrix} F_{int_0} \\ F_{int_1} \\ F_{int_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix}$

Filling in M, $\vec{F_{int}}$ and $\vec{F_{ext}}$ in Equation (3.5) results in:

$$min||\left(\begin{bmatrix} 1 & -.243 & 0 \\ -1 & 0 & -.707 \\ 0 & 0 & -.707 \end{bmatrix} \times \begin{bmatrix} F_{int_0} \\ F_{int_1} \\ F_{int_2} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix} \right)||^2 \tag{3.6}$$

Finally, resulting in:

$$\begin{aligned} min\Big(&\Big(\sigma_0 F_{int_0} - 0.243\sigma_0 F_{int_1}\Big)^2 + \Big(-\sigma_0 F_{int_0} - 0.707\sigma_0 F_{int_2} + -2000\Big)^2 \\ &+ \Big(-0.707\sigma_0 F_{int_2} - 2000\Big)^2\Big) \end{aligned} \tag{3.7}$$

It is observed after classically optimizing with the scipy.optimize.minimize Python package to obtain the values for $A_0 = 4.0 \cdot 10^{-4} m^2$, $A_1 = 1.649 \cdot 10^{-3} m^2$ and $A_2 = 5.657 \cdot 10^{-4} m^2$ with $\sigma_{max} = 5 \cdot 10^6$ Pa.

3.2. CAST THE THREE-MEMBER TRUSS PROBLEM TO HAMILTONIANS

In this subsection the three-member truss problem is casted into Hamiltonians. The Hamiltonians have to be Hermitian to ensure that the operator U is unitary, where U = e^{iHt}. In order to obtain unitary operators U the Hamiltonians consist of Pauli gates only. In the next subsubsections the initial state, the mixer Hamiltonian H_B and the cost Hamiltonian H_C are treated one-by-one.

INITIAL STATE

The initial state is analyzed first. The quantum computer prepares the qubits to be in the $|0\rangle$ state. Instead of putting qubits in superposition state, the middlest area options are put in the $|1\rangle$ state by applying X-gates to qubits one, four and seven, X_1, X_4, X_7. This will initialize the qubits to be in bit string 010010010, which is option 14 in Table 3.1.

MIXER HAMILTONIAN

The mixer Hamiltonian is simply obtained by applying the SWAP gate to all feasible qubit states. A SWAP gate is implemented by applying the double Pauli-X gate with double Pauli-Y gate as explained in Appendix B.3. The mixer Hamiltonian H_B for the three-member truss system is:

$$\begin{aligned} H_B = {} & (0.5+0j)*X1*X0 + (0.5+0j)*Y1*Y0 + (0.5+0j)*X0*X2 + (0.5+0j)*Y0*Y2 + \\ & (0.5+0j)*X2*X1 + (0.5+0j)*Y2*Y1 + (0.5+0j)*X4*X3 + (0.5+0j)*Y4*Y3 + \\ & (0.5+0j)*X3*X5 + (0.5+0j)*Y3*Y5 + (0.5+0j)*X5*X4 + (0.5+0j)*Y5*Y4 + \\ & (0.5+0j)*X7*X6 + (0.5+0j)*Y7*Y6 + (0.5+0j)*X6*X8 + (0.5+0j)*Y6*Y8 + \\ & (0.5+0j)*X8*X7 + (0.5+0j)*Y8*Y7, \end{aligned} \tag{3.8}$$

where X = $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and Y = $\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$

So, applying this sequence of gates: (0.5+0j)*X1*X0 + (0.5+0j)*Y1*Y0, will swap the initial state from bit string 010010010 to bit string 100010010, which is option 23 of Table 3.1.

COST HAMILTONIAN

Finally, the cost Hamiltonian H_C is analyzed, starting with:

$$M \times \vec{F_{int}} = \vec{F_{ext}} \tag{3.9}$$

Starting with the same matrix as in Equation (D.1):

$$\begin{bmatrix} 1 & -.243 & 0 \\ -1 & 0 & -.707 \\ 0 & 0 & -.707 \end{bmatrix} \times \begin{bmatrix} F_{int_0} \\ F_{int_1} \\ F_{int_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix} \tag{3.10}$$

Assume that every truss experiences σ_0:

$$\begin{bmatrix} 1 & -.243 & 0 \\ -1 & 0 & -.707 \\ 0 & 0 & -.707 \end{bmatrix} \times \begin{bmatrix} \sigma_0 A_0 \\ \sigma_0 A_1 \\ \sigma_0 A_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix} \tag{3.11}$$

Replace continuous A_j to three discrete options per truss member:

$$\begin{bmatrix} 1 & -.243 & 0 \\ -1 & 0 & -.707 \\ 0 & 0 & -.707 \end{bmatrix} \times \begin{bmatrix} \sigma_0((A_0 - dA)q_0 + A_0 q_1 + (A_0 + dA)q_2) \\ \sigma_0((A_1 - dA)q_3 + A_1 q_4 + (A_1 + dA)q_5) \\ \sigma_0((A_2 - dA)q_6 + A_2 q_7 + (A_2 + dA)q_8) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix} \tag{3.12}$$

Where A_x represents the area choices and σ_0 is specified by the material and q acts as a switch (0 or 1), only one q per truss member can be one. As can be seen from the matrix, the first option has an area A_0 - dA, the second option is the area A_0 and the third option is A_0 + dA. Numbering the areas

in this way is inconvenient and therefore $A_0 - dA$, A_0 and $A_0 + dA$ are relabeled to A_0, A_1 and A_2 respectively:

$$M \times \vec{F_{int}} = \vec{F_{ext}} = \begin{bmatrix} 1 & -.243 & 0 \\ -1 & 0 & -.707 \\ 0 & 0 & -.707 \end{bmatrix} \times \begin{bmatrix} \sigma_0(A_0 q_0 + A_1 q_1 + A_2 q_2) \\ \sigma_0(A_3 q_3 + A_4 q_4 + A_5 q_5) \\ \sigma_0(A_6 q_6 + A_7 q_7 + A_8 q_8) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix} \tag{3.13}$$

The norm of the vector is simply squared in order to connect the costs of the trusses with one another, similar to Equation (3.4). The q-symbols are highlighted in blue to show that these discrete variables contribute to finding the minimum when the best combination is chosen:

$$||(M \times \vec{F_{int}} - \vec{F_{ext}})||^2 = ||\left(\begin{bmatrix} 1 & -.243 & 0 \\ -1 & 0 & -.707 \\ 0 & 0 & -.707 \end{bmatrix} \times \begin{bmatrix} \sigma_0(A_0 q_0 + A_1 q_1 + A_2 q_2) \\ \sigma_0(A_3 q_3 + A_4 q_4 + A_5 q_5) \\ \sigma_0(A_6 q_6 + A_7 q_7 + A_8 q_8) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix} \right)||^2 \tag{3.14}$$

Multiplying Equation (3.14) to obtain the objective function, results in:

$$\begin{aligned} ||(M \times \vec{F_{int}} - \vec{F_{ext}})||^2 = &\Big(\big(\sigma_0(A_0 q_0 + A_1 q_1 + A_2 q_2) - 0.243\sigma_0(A_3 q_3 + A_4 q_4 + A_5 q_5)\big)^2 \\ &\big(-\sigma_0(A_0 q_0 + A_1 q_1 + A_2 q_2) - 0.707\sigma_0(A_6 q_6 + A_7 q_7 + A_8 q_8) \\ &- 2000\big)^2 - \big(0.707\sigma_0(A_6 q_6 + A_7 q_7 + A_8 q_8) - 2000\big)^2\Big) \end{aligned} \tag{3.15}$$

Now expanding Equation (3.15):

$$\begin{aligned} ||(M \times \vec{F_{int}} - \vec{F_{ext}})||^2 = &\Big(2A_0^2 {q_0}^2 \sigma_0^2 + 4A_0 A_1 q_0 q_1 \sigma_0^2 + 4A_0 A_2 q_0 q_2 \sigma_0^2 - 0.486 A_0 A_3 q_0 q_3 \sigma_0^2 - \\ &0.486 A_0 A_4 q_0 q_4 \sigma_0^2 - 0.486 A_0 A_5 q_0 q_5 \sigma_0^2 + 1.414 A_0 A_6 q_0 q_6 \sigma_0^2 + 1.414 A_0 A_7 q_0 q_7 \sigma_0^2 + \\ &1.414 A_0 A_8 q_0 q_8 \sigma_0^2 + 4000 A_0 q_0 \sigma_0 + 2A_1^2 {q_1}^2 \sigma_0^2 + 4A_1 A_2 q_1 q_2 \sigma_0^2 - \\ &0.486 A_1 A_3 q_1 q_3 \sigma_0^2 - 0.486 A_1 A_4 q_1 q_4 \sigma_0^2 - 0.486 A_1 A_5 q_1 q_5 \sigma_0^2 + 1.414 A_1 A_6 q_1 q_6 \sigma_0^2 + \\ &1.414 A_1 A_7 q_1 q_7 \sigma_0^2 + 1.414 A_1 A_8 q_1 q_8 \sigma_0^2 + 4000 A_1 q_1 \sigma_0 + 2A_2^2 {q_2}^2 \sigma_0^2 - \\ &0.486 A_2 A_3 q_2 q_3 \sigma_0^2 - 0.486 A_2 A_4 q_2 q_4 \sigma_0^2 - 0.486 A_2 A_5 q_2 q_5 \sigma_0^2 + 1.414 A_2 A_6 q_2 q_6 \sigma_0^2 + \\ &1.414 A_2 A_7 q_2 q_7 \sigma_0^2 + 1.414 A_2 A_8 q_2 q_8 \sigma_0^2 + 4000 A_2 q_2 \sigma_0 + 0.05905 A_3^2 {q_3}^2 \sigma_0^2 + \\ &0.1181 A_3 A_4 q_3 q_4 \sigma_0^2 + 0.1181 A_3 A_5 q_3 q_5 \sigma_0^2 + 0.05905 A_4^2 {q_4}^2 \sigma_0^2 + 0.1181 A_4 A_5 q_4 q_5 \sigma_0^2 + \\ &0.05905 A_5^2 {q_5}^2 \sigma_0^2 + 0.9997 A_6^2 {q_6}^2 \sigma_0^2 + 1.999 A_6 A_7 q_6 q_7 \sigma_0^2 + 1.999 A_6 A_8 q_6 q_8 \sigma_0^2 + \\ &5656.0 A_6 q_6 \sigma_0 + 0.9997 A_7^2 {q_7}^2 \sigma_0^2 + 1.999 A_7 A_8 q_7 q_8 \sigma_0^2 + 5656.0 A_7 q_7 \sigma_0 + \\ &0.9997 A_8^2 {q_8}^2 \sigma_0^2 + 5656.0 A_8 q_8 \sigma_0 + 8000000 I\Big) \end{aligned} \tag{3.16}$$

Qubits q_0 and q_1 cannot be in $|1\rangle$ at the same time, since it is not allowed to select more than one area per truss member. Therefore, costs that contribute to invalid solutions are nullified, which results in the following equation:

$$\begin{aligned}\sum(M\times F_{int}^{\vec{}}-F_{ext}^{\vec{}})^2 = \Big(&2A_0^2{q_0}^2\sigma_0^2-0.486A_0A_3q_0q_3\sigma_0^2-0.486A_0A_4q_0q_4\sigma_0^2-0.486A_0A_5q_0q_5\sigma_0^2\\&+1.414A_0A_6q_0q_6\sigma_0^2+1.414A_0A_7q_0q_7\sigma_0^2+1.414A_0A_8q_0q_8\sigma_0^2+4000A_0q_0\sigma_0+\\&2A_1^2{q_1}^2\sigma_0^2-0.486A_1A_3q_1q_3\sigma_0^2-0.486A_1A_4q_1q_4\sigma_0^2-0.486A_1A_5q_1q_5\sigma_0^2+\\&1.414A_1A_6q_1q_6\sigma_0^2+1.414A_1A_7q_1q_7\sigma_0^2+1.414A_1A_8q_1q_8\sigma_0^2+4000A_1q_1\sigma_0+\\&2A_2^2{q_2}^2\sigma_0^2-0.486A_2A_3q_2q_3\sigma_0^2-0.486A_2A_4q_2q_4\sigma_0^2-0.486A_2A_5q_2q_5\sigma_0^2+\\&1.414A_2A_6q_2q_6\sigma_0^2+1.414A_2A_7q_2q_7\sigma_0^2+1.414A_2A_8q_2q_8\sigma_0^2+4000A_2q_2\sigma_0+\\&0.05905A_3^2{q_3}^2\sigma_0^2+0.05905A_4^2{q_4}^2\sigma_0^2+0.05905A_5^2{q_5}^2\sigma_0^2+0.9997A_6^2{q_6}^2\sigma_0^2+\\&5656.0A_6q_6\sigma_0+0.9997A_7^2{q_7}^2\sigma_0^2+5656.0A_7q_7\sigma_0+0.9997A_8^2{q_8}^2\sigma_0^2+\\&5656.0A_8q_8\sigma_0+8000000I\Big)\end{aligned} \tag{3.17}$$

To obtain the cost Hamiltonian H_C, that encodes the objective function, it is required to substitute q_x with the measurement operator $\frac{1}{2}(I-Z_x)$. In order for the quantum computer to measure in the computational basis (i.e. $|0\rangle$ and $|1\rangle$) the Z operator is applied. The discrete binary variable $q_x \in \{0,1\}$ has to get mapped to the measurement operator, because Z has eigenvalues $\{-1,1\}$. For more details, refer to Appendix B.1 and Hadfield *et al.* [40]. After substitution:

$$\begin{aligned}H_C = \Big(&2A_0^2\frac{1}{4}(I-Z_0)^2\sigma_0^2+0.486A_0A_3\frac{1}{2}(I-Z_0)\frac{1}{2}(I-Z_3)\sigma_0^2-0.486A_0A_4\frac{1}{2}(I-Z_0)\frac{1}{2}(I-Z_4)\sigma_0^2-\\&0.486A_0A_5\frac{1}{2}(I-Z_0)\frac{1}{2}(I-Z_5)\sigma_0^2+1.414A_0A_6\frac{1}{2}(I-Z_0)\frac{1}{2}(I-Z_6)\sigma_0^2+1.414A_0A_7\frac{1}{2}(I-Z_0)\frac{1}{2}(I-Z_7)\sigma_0^2+\\&1.414A_0A_8\frac{1}{2}(I-Z_0)\frac{1}{2}(I-Z_8)\sigma_0^2+4000A_0\frac{1}{2}(I-Z_0)\sigma_0+2A_1^2\frac{1}{4}(I-Z_1)^2\sigma_0^2-0.486A_1A_3\frac{1}{2}(I-Z_1)\frac{1}{2}(I-Z_3)\sigma_0^2-\\&0.486A_1A_4\frac{1}{2}(I-Z_1)\frac{1}{2}(I-Z_4)\sigma_0^2-0.486A_1A_5\frac{1}{2}(I-Z_1)\frac{1}{2}(I-Z_5)\sigma_0^2+1.414A_1A_6\frac{1}{2}(I-Z_1)\frac{1}{2}(I-Z_6)\sigma_0^2+\\&1.414A_1A_7\frac{1}{2}(I-Z_1)\frac{1}{2}(I-Z_7)\sigma_0^2+1.414A_1A_8\frac{1}{2}(I-Z_1)\frac{1}{2}(I-Z_8)\sigma_0^2+4000A_1\frac{1}{2}(I-Z_1)\sigma_0+\\&2A_2^2\frac{1}{4}(I-Z_2)^2\sigma_0^2-0.486A_2A_3\frac{1}{2}(I-Z_2)\frac{1}{2}(I-Z_3)\sigma_0^2-0.486A_2A_4\frac{1}{2}(I-Z_2)\frac{1}{2}(I-Z_4)\sigma_0^2-\\&0.486A_2A_5\frac{1}{2}(I-Z_2)\frac{1}{2}(I-Z_5)\sigma_0^2+1.414A_2A_6\frac{1}{2}(I-Z_2)\frac{1}{2}(I-Z_6)\sigma_0^2+1.414A_2A_7\frac{1}{2}(I-Z_2)\frac{1}{2}(I-Z_7)\sigma_0^2+\\&1.414A_2A_8\frac{1}{2}(I-Z_2)\frac{1}{2}(I-Z_8)\sigma_0^2+4000A_2\frac{1}{2}(I-Z_2)\sigma_0+0.05905A_3^2\frac{1}{4}(I-Z_3)^2\sigma_0^2+0.05905A_4^2\frac{1}{4}(I-Z_4)^2\sigma_0^2+\\&0.05905A_5^2\frac{1}{4}(I-Z_5)^2\sigma_0^2+0.9997A_6^2\frac{1}{4}(I-Z_6)^2\sigma_0^2+5656.0A_6\frac{1}{2}(I-Z_6)\sigma_0+0.9997A_7^2\frac{1}{4}(I-Z_7)^2\sigma_0^2+\\&5656.0A_7\frac{1}{2}(I-Z_7)\sigma_0+0.9997A_8^2\frac{1}{4}(I-Z_8)^2\sigma_0^2+5656.0A_8\frac{1}{2}(I-Z_8)\sigma_0+8000000I\Big)\end{aligned} \tag{3.18}$$

Applying a double Pauli-Z gate acting on a single qubit results in the identity gate, which will not be comparable with the classical solution. For this reason, double Pauli-Z gates are reduced to a single Pauli-Z gate in the final equation. Further expansion and reordering results in:

$$
\begin{aligned}
H_C = ((&0.5A_0^2\sigma_0^2 - 2A_0^2\sigma_0^2 - A_0A_1\sigma_0^2 - A_0A_2\sigma_0^2 + 0.1215A_0A_3\sigma_0^2 + 0.1215A_0A_4\sigma_0^2 + 0.1215A_0A_5\sigma_0^2 - \\
&0.3535A_0A_6\sigma_0^2 - 0.3535A_0A_7\sigma_0^2 - 0.3535A_0A_8\sigma_0^2 - 2000A_0\sigma_0)Z_0 + \\
(&-A_0A_1\sigma_0^2 + 0.5A_1^2\sigma_0^2 - 2A_1^2\sigma_0^2 - A_1A_2\sigma_0^2 + 0.1215A_1A_3\sigma_0^2 + 0.1215A_1A_4\sigma_0^2 + 0.1215A_1A_5\sigma_0^2 - \\
&0.3535A_1A_6\sigma_0^2 - 0.3535A_1A_7\sigma_0^2 - 0.3535A_1A_8\sigma_0^2 - 2000A_1\sigma_0)Z_1 + \\
(&-A_0A_2\sigma_0^2 - A_1A_2\sigma_0^2 + 0.5A_2^2\sigma_0^2 - 2A_2^2\sigma_0^2 + 0.1215A_2A_3\sigma_0^2 + 0.1215A_2A_4\sigma_0^2 + 0.1215A_2A_5\sigma_0^2 - \\
&0.3535A_2A_6\sigma_0^2 - 0.3535A_2A_7\sigma_0^2 - 0.3535A_2A_8\sigma_0^2 - 2000A_2\sigma_0)Z_2 + \\
(&0.1215A_0A_3\sigma_0^2 + 0.1215A_1A_3\sigma_0^2 + 0.1215A_2A_3\sigma_0^2 + 0.01476A_3^2\sigma_0^2 - 0.059049A_3^2\sigma_0^2 - 0.02952A_3A_4\sigma_0^2 - 0.02952A_3A_5\sigma_0^2)Z_3 + \\
(&0.1215A_0A_4\sigma_0^2 + 0.1215A_1A_4\sigma_0^2 + 0.1215A_2A_4\sigma_0^2 - 0.02952A_3A_4\sigma_0^2 + 0.01476A_4^2\sigma_0^2 - 0.059049A_4^2\sigma_0^2 - 0.02952A_4A_5\sigma_0^2)Z_4 + \\
(&0.1215A_0A_5\sigma_0^2 + 0.1215A_1A_5\sigma_0^2 + 0.1215A_2A_5\sigma_0^2 - 0.02952A_3A_5\sigma_0^2 - 0.02952A_4A_5\sigma_0^2 + 0.01476A_5^2\sigma_0^2 - 0.059049A_5^2\sigma_0^2)Z_5 + \\
(&-0.3535A_0A_6\sigma_0^2 - 0.3535A_1A_6\sigma_0^2 - 0.3535A_2A_6\sigma_0^2 + 0.2499A_6^2\sigma_0^2 - 0.999698A_6^2\sigma_0^2 - 0.4998A_6A_7\sigma_0^2 - 0.4998A_6A_8\sigma_0^2 - 2828.0A_6\sigma_0)Z_6 + \\
(&-0.3535A_0A_7\sigma_0^2 - 0.3535A_1A_7\sigma_0^2 - 0.3535A_2A_7\sigma_0^2 - 0.4998A_6A_7\sigma_0^2 + 0.2499A_7^2\sigma_0^2 - 0.999698A_7^2\sigma_0^2 - 0.4998A_7A_8\sigma_0^2 - 2828.0A_7\sigma_0)Z_7 + \\
(&-0.3535A_0A_8\sigma_0^2 - 0.3535A_1A_8\sigma_0^2 - 0.3535A_2A_8\sigma_0^2 - 0.4998A_6A_8\sigma_0^2 - 0.4998A_7A_8\sigma_0^2 + 0.2499A_8^2\sigma_0^2 - 2828.0A_8\sigma_0 - 0.9997A_8^2\sigma_0^2)Z_8 - \\
&0.1215A_0A_3Z_0Z_3\sigma_0^2 - 0.1215A_0A_4Z_0Z_4\sigma_0^2 - 0.1215A_0A_5Z_0Z_5\sigma_0^2 + 0.3535A_0A_6Z_0Z_6\sigma_0^2 + 0.3535A_0A_7Z_0Z_7\sigma_0^2 \\
&+ 0.3535A_0A_8Z_0Z_8\sigma_0^2 - 0.1215A_1A_3Z_1Z_3\sigma_0^2 - 0.1215A_1A_4Z_1Z_4\sigma_0^2 - 0.1215A_1A_5Z_1Z_5\sigma_0^2 \\
&+ 0.3535A_1A_6Z_1Z_6\sigma_0^2 + 0.3535A_1A_7Z_1Z_7\sigma_0^2 + 0.3535A_1A_8Z_1Z_8\sigma_0^2 - 0.1215A_2A_3Z_2Z_3\sigma_0^2 - 0.1215A_2A_4Z_2Z_4\sigma_0^2 - 0.1215A_2A_5Z_2Z_5\sigma_0^2 + \\
&0.3535A_2A_6Z_2Z_6\sigma_0^2 + 0.3535A_2A_7Z_2Z_7\sigma_0^2 + 0.3535A_2A_8Z_2Z_8\sigma_0^2 + 0.1215A_0A_4\sigma_0^2 + 0.1215A_0A_5\sigma_0^2 - 0.3535A_0A_6\sigma_0^2 - \\
&0.3535A_0A_7\sigma_0^2 - 0.3535A_0A_8\sigma_0^2 + 0.1215A_1A_3\sigma_0^2 + 0.1215A_1A_4\sigma_0^2 + 0.1215A_1A_5\sigma_0^2 - 0.3535A_1A_6\sigma_0^2 - 0.3535A_1A_7\sigma_0^2 - \\
&0.3535A_1A_8\sigma_0^2 + 0.1215A_2A_3\sigma_0^2 + 0.1215A_2A_4\sigma_0^2 + 0.1215A_2A_5\sigma_0^2 - 0.3535A_2A_6\sigma_0^2 - 0.3535A_2A_7\sigma_0^2 - 0.02952A_3^2\sigma_0^2 \\
&- 0.02952A_3A_4\sigma_0^2 - 0.02952A_3A_5\sigma_0^2 - 0.02952A_4^2\sigma_0^2 - 0.02952A_4A_5\sigma_0^2 - 0.02952A_5^2\sigma_0^2 - 0.4998A_6^2\sigma_0^2 - 0.4998A_6A_7\sigma_0^2 - \\
&0.4998A_6A_8\sigma_0^2 + 2000A_1\sigma_0 + 2828.0A_6\sigma_0 - 0.4998A_7^2\sigma_0^2 - 0.4998A_7A_8\sigma_0^2 + 2828.0A_7\sigma_0 - 0.4998A_8^2\sigma_0^2 + 0.1215A_0A_3\sigma_0^2 - \\
&0.3535A_2A_8\sigma_0^2 + 2000A_2\sigma_0 - A_2^2\sigma_0^2 - A_1A_2\sigma_0^2 - A_0A_2\sigma_0^2 + 2828.0A_8\sigma_0 + 2000A_0\sigma_0 - A_1^2\sigma_0^2 - A_0A_1\sigma_0^2 - A_0^2\sigma_0^2 + 8000000 \cdot I)
\end{aligned}
\tag{3.19}
$$

The cost Hamiltonian is obtained by filling in the known values for A_x and σ_0 in Equation (3.19). It can be seen from the cost Hamiltonian that it does not include the density ρ and the length of the trusses L. Thereby, the cost Hamiltonian minimizes the areas rather than the weight. If one would like to obtain the weight, then simply multiply the minimized areas, i.e. when the cost is 0, with the constant density ρ and the length of the truss L.

3.3. CALCULATING THE EXPECTATION WITH VQE

The expectation is calculated as it will be used as the objective function to minimize with the scipy.optimize.minimize package. In order to calculate this a Python package was used, named *quantum-grove*, written by Rigetti. It is currently not maintained and has a number of open issues[1]. For this reason the author of this thesis has modified the source code to make it compatible with the Python package "pyquil", which is written and actively maintained by Rigetti.

Initially, the entire quantum circuit was created where only one or two qubits were actually measured, depending on the Pauli Z-gate(s) of which the expectation had to be evaluated. For example, if the expectation of Z_0 and Z_3 had to be evaluated, only qubits 0 and 3 were measured in the quantum circuit. It becomes clear that, when the cost Hamiltonian consists of many Pauli Z-gates the quantum circuit has to be run many times to evaluate the total expectation. Thereby, it was too slow for the Aspen-9 quantum computer to run within 15 minutes, which is the minimum time that a user of Rigetti's services is required to reserve. To reduce the computational time all qubits are measured simultaneously.

[1] https://github.com/rigetti/grove/issues

The expectation is then calculated by taking the following steps. Firstly, the quantum circuit was run, where all Pauli terms are simultaneously measured into a set of bit strings. Thereafter, the expectation is calculated with the VQE method explained in Section 2.4 and further explained below.

Imagine a cost Hamiltonian:

$$H_C = aI + \sum_{i=0}^{P-1} b_i Z_i + \sum_{i=0}^{P-1} \sum_{i \neq j} (c_{ij}) Z_i Z_j, \tag{3.20}$$

where a, b_i and $c_{ij} \in \mathbb{R}$, Z_i is the Pauli-Z gate operator acting on qubit i, P is the number of Pauli-Z gates. Calculating the expectation of the cost Hamiltonian for all measured bit strings up to n bit strings is done in the following way:

$$F_p = \begin{cases} a \cdot 1 - \mathrm{m} \cdot \sum_{i}^{P-1} b_i - \mathrm{m} \cdot \sum_{i=0}^{P-1} \sum_{i \neq j} (c_{ij}), & \text{if } \sum_{i=0}^{P-1} Z_i|1\rangle, \sum_{i=0}^{P-1} \sum_{i \neq j} Z_i Z_j|10\rangle \text{ and } \sum_{i=0}^{P-1} \sum_{i \neq j} Z_i Z_j|01\rangle \\ a \cdot 1 + \mathrm{m} \cdot \sum_{i}^{P-1} b_i + \mathrm{m} \cdot \sum_{i=0}^{P-1} \sum_{i \neq j} (c_{ij}), & \text{otherwise} \end{cases} \tag{3.21}$$

where $\mathrm{m} = \sum_{i=0}^{n-1} \frac{\text{counts}_i}{\text{\# valid counts}}$ and stands for measurement from bit string zero up to n bit strings.

Initially, invalid entries were also used to calculate the expectation value. This caused severe noise in the expectation value, which made it impossible for the optimizer to converge. To prevent this from happening, those invalid bit strings were deleted.

3.4. HARDWARE

In this section hardware specific subjects such as connectivity of qubits, reduction of computational time and reduction of noise are treated in Subsection 3.4.1, Subsection 3.4.2 and Subsection 3.4.3 respectively.

3.4.1. CONNECTIVITY OF QUBITS

Quantum computers are not fully or densely connected, instead quantum computers currently are sparsely connected and do not consist of all-to-all connected qubits. In order to map the qubits specified by the quantum circuit with the physical qubits, it is required to introduce extra swap gates, which produces extra noise. Future prospects show possibilities of fully connecting all qubits by using similar techniques as done in conventional microprocessors with the Complementary Metal-Oxide-Semiconductor (CMOS) technology by solving the wire bottleneck [47]. In Figure 3.3, the lattice from quantum computing company Rigetti of the Aspen-9 is shown. As can be seen qubit 5 is connected to qubits 4 and 6 and not to all qubits. For more information on the lattice structure of quantum virtual machines refer to Rigetti Computing [48].

Figure 3.3: Figure retrieved from Rigetti[2].

[2] https://qcs.rigetti.com/lattices at 15 June 2021.

Rigetti has various ways to simulate a quantum computer with quantum virtual machines. A quantum virtual machine that has full connectivity between qubits, is for example the 12q-qvm, which consists of twelve fully connected qubits. The Aspen-9-qvm is the most realistic quantum virtual machine, because it is a one-to-one mapping to the real quantum hardware of Rigetti. For more information about quantum virtual machines, refer to Rigetti Computing [48].

There is also a way to calculate the expectation value of quantum observables (e.g. a sum of Pauli gates) directly and exactly without having to simulate the quantum circuit. The WavefunctionSimulator of Rigetti has access to the full wavefunction and therefore it can exactly calculate the expectation value of quantum observables. For more information about the WavefunctionSimulator, refer to Rigetti Computing [48].

3.4.2. REDUCTION OF COMPUTATIONAL TIME

In total four runs of fifteen minutes on real quantum hardware were available. At first, the author of this report reserved a fifteen minute time slot to run the obtained quantum circuit. It was noticed that it was not possible to obtain results within this fifteen minute time slot. Therefore, a number of methods have been tried on a quantum simulator to reduce the computational time.

Firstly, it was tried to reduce the number of shots to evaluate one Pauli term from 1024 to 64. The number of shots defines the number of times a quantum circuit with the same input is rerun in order to produce a probability distribution of counts. Decreasing the number of shots, will therefore decrease the computational time. At the same time it reduces the probability of finding correct results to unacceptable levels, which was to be expected. It was not possible for the expectation value to converge and for this reason this method did not help to reduce the computational time.

Secondly, the used python package *quantum-grove*, measured qubits corresponding to certain Pauli-Z gate(s) one-by-one. This was computationally expensive, so instead the code from the Python package *quantum-grove* was modified to measure all qubits at once, such that the expectation value could be simultaneously calculated for all Pauli-Z gates. This reduced the time to run the quantum circuit by P times, where P is the number of Pauli terms in the cost Hamiltonian as earlier explained in Section 3.3.

Thirdly, different global and local optimizers to optimize for the expectation value by varying β and γ parameters were investigated. All optimizers started with a random starting angle for both β and γ, except for the Simplicial Homology Global Optimizer (SHGO) optimization routine. The Python package scipy.optimize.minimize[3] was used as a classical minimizer. All standard minimization routines, that do not require a Jacobian or Hessian function as inputs were compared. It is, namely not possible to calculate a Jacobian or Hessian from the objective function, because the objective function, the expectation, is just a number. The used optimizers include Nelder-Mead, Powell, Conjugate Gradient (CG), Broyden–Fletcher–Goldfarb–Shanno (BFGS), Limited-memory Broyden–Fletcher–Goldfarb–Shanno Bounded (L-BFGS-B), Truncated Newton (TNC), Constrained Binary Optimization By Linear Approximation (COBYLA), Sequential Least Squares Programming (SLSQP), Trust region method (trust-constr) and the more recent SHGO[49]. For more information about how these algorithms work refer to the documentation provided in Scipy[4].

In Figure 3.4 the time and minimized cost is shown for all used optimizers for the two-member truss

[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html
[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

structure with step parameter p set to two. It can be seen that some minimizers, such as CG, BFGS, L-BFGS-B, TNC increase the cost rather than minimizing it. It is hard for these minimization routines to find the minimum due to the low p number.

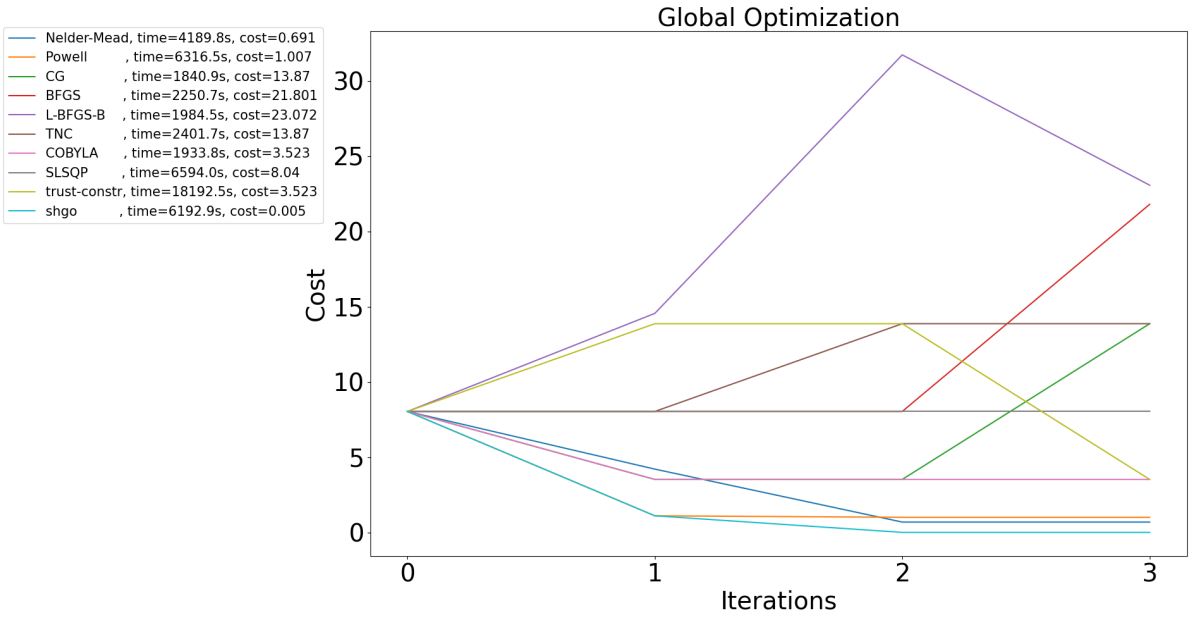


Figure 3.4: Global optimization of the three-member truss structure to show the difference in time and cost for the different local and global optimizers.

It can be seen in Figure 3.4 that the only minimization routine able to minimize the cost to a value close to zero after two iterations is SHGO. Therefore, it can be concluded that the SHGO minimization routine was most effective when considering computational speed and final cost value.

Furthermore, the use of different stopping criteria was investigated by setting the tolerance to different levels by trial-and-error. When optimizing with the minimization package from scipy, the tolerance has to be predefined as an input. One option that was considered was to put a tolerance on the expectation value. For example, the tolerance could have been set to an expectation value reaching an increment of 0.1% when compared to the previously obtained expectation value. It was hard to estimate this figure, since generally the user does not know in advance what the expectation value will be. It was hard to find generalizing stopping criteria for various truss structures, therefore the stopping criteria that was considered most effective was one with a cap on the number of iterations, determined by trial-and-error, in this case 250. Convergence cannot be guaranteed when capping the number of iterations. For this reason the user has the responsibility of capping the number of iterations to a reasonable number, such that the minimum is obtained.

The quantum circuit that was constructed by the QAOA can be parameterized, such that the quantum circuit does not have to recompile at every iteration. Parameterizing the compilation, is supposed to reduce the computational time. However, by parameterizing the compilation, an increase was observed in the total number of gates used. The reason for it being is that instead of having one value for the RZ gate, for example RZ(2.525), two values are assigned to the RZ gate, e.g. RZ(π+2γ). To take the extra information in the RZ gate into account, the quantum circuit required extra RX gates. Introducing more gates when using real quantum hardware results in longer execution time and introduces more noise. All in all, parameterizing the compilation did not help to reduce the computational time and was not used for this reason.

Finally, the RESET gate was used to actively reset the qubits prior to running the quantum circuit. Actively resetting the qubits entails that qubits are actively reset to their $|0\rangle$ state, instead of waiting several times the coherence length for the qubits to decay to their $|0\rangle$ state. Actively resetting is faster, however it introduces ~1% of error per qubit. Naturally waiting until the qubits decay to $|0\rangle$ has the same order of magnitude of noise levels, due to thermal noise [48]. This strategy to reduce the computational time was considered helpful.

3.4.3. NOISE

The Rigetti compiler has three strategies (naive, partial and greedy) to map virtual qubits to physical qubits on the quantum lattice. If no instruction is given the compiler will choose between the strategies naive and partial. Choosing the naive rewiring strategy means that the logical qubits are mapped to physical qubits without thinking if these qubits have optimal fidelity (quality of qubits). If qubits have to be rewired, because they cannot be directly mapped from logical qubits to physical qubits, the partial strategy will be used as the default method. This partial strategy will take more time to compile, while it optimizes for fidelity. The greedy approach favors compilation time over fidelity by a simple optimization heuristic that optimizes for distances between virtual qubits and physical qubits. The author of this report has explicitly chosen for the highest fidelity partial strategy, so that noise is reduced. For more information refer to Rigetti Computing [48].

A python package named *mitiq* was tried, which is a Python package for error mitigation on NISQ computers, to find the standard repeatable error of the quantum computer and account for that in a smart way. Error mitigation is, as the name indicates, mitigating the error, as opposed to error correction, which completely eliminates errors [50]. There are several error mitigation techniques to mitigate errors, such as Zero-Noise Extrapolation (ZNE), Probabilistic Error Cancellation (PEC) and Clifford Data Regression Module described in the following subsubsections as summary of LaRose *et al.* [50].

ZERO-NOISE EXTRAPOLATION (ZNE)

The ZNE technique was tried to explore techniques to mitigate noise. ZNE scales noise by artificially increasing the number of gates by unitary folding of gates: $GG^{\dagger}G$, where a circuit G is multiplied by its inverse and by the circuit G again. This artificially increases the noise by keeping the circuit equivalent, after which the noise can be extrapolated back to a zero-noise limit.

For illustrative purposes this is shown in Figure 3.5, where on the horizontal axes the noise is increased by λ by applying the unitary folding technique. The noise is increased up to a certain number of times λ, after which the noise can be extrapolated back to a zero-noise limit with different extrapolation methods, such as linear, Richardson, quadratic and exponential.

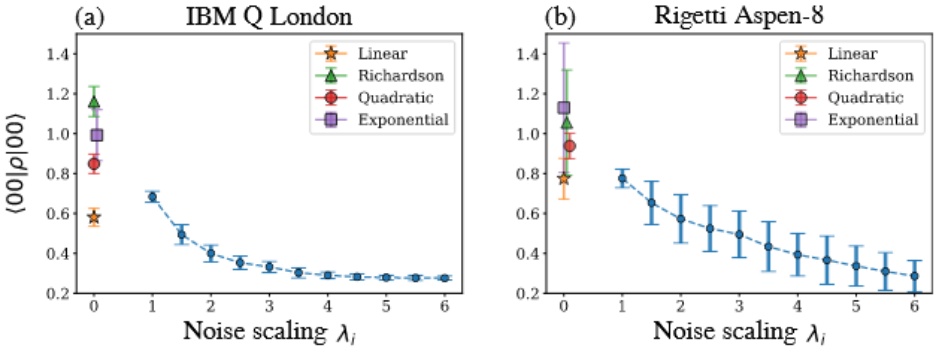


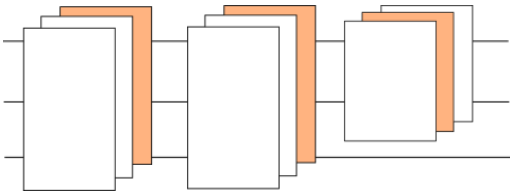
Figure 3.5: Figure retrieved from LaRose *et al.* [50].

PROBABILISTIC ERROR CANCELLATION (PEC)

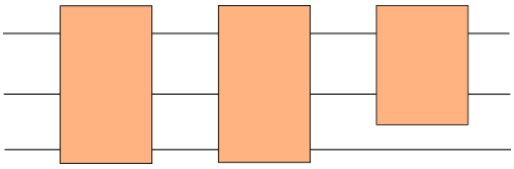
Another technique used to mitigate the noise is called Probabilistic Error Cancellation, described here as a summary of LaRose *et al.* [50]. The technique describes a perfectly noiseless unitary channel U by a linear combination of noisy channels to be implemented with multiple unitary channels constructing a quantum circuit, shown graphically in Figure 3.6a, by a quantum computer:

$$\mathscr{G} = \sum_{\alpha} \eta(\alpha)\mathscr{O}_{\alpha}, \tag{3.22}$$

where $\sum_{\alpha} \eta(\alpha) = 1$ to preserve the trace and $\eta \in \mathbb{R}$, $\mathscr{G}$ and $\mathscr{O}$ are density operators, describing the quantum state of a physical system. When each ideal unitary is known with sufficiently high accuracy, PEC can converge to the ideal expectation value when taking the limit of the number of sampled circuits. Where the expectation value is estimated by taking the Monte Carlo average over the total number of sampled circuits.



(a) Pool of 3x3 different noisy unitary channels. From left to right, on the left-hand side of the figure, three 3-qubit gates are shown, in the middle another three 3-qubit gate and finally on the right-hand side three 2-qubit gates are shown. Random selection from this pool of unitary channels results in the orange colored circuit.



(b) Circuit randomly created from different unitary channels

Figure 3.6: Sampled circuits created with PEC. Figure adapted from LaRose *et al.* [50].

CONCLUSION

All in all, ZNE was considered unhelpful, because the circuit size was already too large to give a zero-noise limit, as was described in Casares *et al.* [51]. Furthermore, Sopena *et al.* [52] stated that the ZNE technique assumes low hardware noise, which is not valid for large quantum circuit sizes and depth. Using ZNE or other different error mitigation techniques, such as Probabilistic Error Cancellation, will increase the computational time, because extra iterations are required. After careful consideration the above described noise mitigation techniques were deemed unhelpful, because the computational time will increase.

3.5. FROM QUBO TO QAOA

Quantum optimization for (in)determinate truss structures has been done before by parameterized FEM by using Quadratic Unconstrained Binary Optimization (QUBO) suitable for Quantum Annealing (QA) computers by Wils [4]. QUBO problems are as the name suggests optimization problems that are quadratic, unconstrained and binary. The objective function is obtained by manipulating, scaling, truncating and applying Ajagekar's iterative method to arrive to the final Quadratic Unconstrained Binary Optimization (QUBO) format.

The work of Wils [4] is mapped from QUBO to QAOA. This can be done by mapping the cost Hamiltonian of the QUBO function from Equation (3.23)[53] to the QAOA format shown in Equation (3.24)[53, 54]. As can be observed this is simply a mapping of the discrete binary variable x $\in \{0,1\}$ to $\frac{1}{2}(I-Z)$, where I is the Identity gate and Z represents the Pauli-Z gate $\in \{-1,1\}$ as described earlier in Section 3.2.

$$f(x) = a + \sum_{j=1}^{n} c_j x_j + \sum_{j<k} d_{jk} x_j x_k \tag{3.23}$$

$$H_c = (a+c+d)I - \frac{1}{2}\sum_{j=1}^{n}\left(c_j + d_j\right)Z_j + \frac{1}{4}\sum_{j<k} d_{jk} Z_j Z_k, \tag{3.24}$$

where $c = \frac{1}{2}\sum_{j=1}^{n} c_j, d = \frac{1}{4}\sum_{j<k} d_{jk}$, and $d_j = \frac{1}{2}\sum_{k\neq j} d_{jk}$ with $d_{jk} = d_{kj}$

Furthermore, the initial state $|s\rangle = |+\rangle^{\otimes n}$ and the mixer Hamiltonian is simply $H_B = \sum_{j=1}^{n} X_j$. It has been shown by Hen and Spedalieri [55] in 2016 that the full search space can be restricted to a feasible subspace by the mixer Hamiltonian in quantum annealing computers. This method could be used instead of using penalty terms in the cost Hamiltonian of QUBO problems to favor valid solutions over invalid ones.

The mixer Hamiltonian can be used to constrain the search space by using for example Equation (3.25)[55], where σ_i^x is the Pauli-X gate and σ_i^y is the Pauli-Y gate. This would significantly reduce the number of required qubits in quantum annealers[55]. Current quantum annealers[5], however have a fixed/non-constraining mixer Hamiltonian and this constraining technique cannot be utilized.

$$H_d = -\sum_{i=1}^{n}\left(\sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y\right) \tag{3.25}$$

When mapping from QUBO to QAOA, it is possible to use the initial mixer Hamiltonian $H_B = \sum_{j=1}^{n} X_j$ and initial state $|s\rangle = |+\rangle^{\otimes n}$. At the same time, it is possible to utilize the constraining mixer Hamiltonian, since it can be easily implemented on a GBQC. Both methods will be explored in Chapter 4.

3.6. INFLUENCE OF STEP PARAMETER P ON GATE VOLUME

In this work quantum software and quantum hardware from Rigetti is used and therefore their corresponding quantum virtual machines and quantum computers. The influence of step parameter p on the gate volume is displayed in Figure 3.7 for the Aspen-9-qvm, to illustrate how the program scales with an increasing number of trusses. The Aspen-9-qvm is a one-to-one virtual copy of the real quantum hardware. After trial-and-error it was found that the step parameter p should be set to two, three and four for the two-member, three-member and four-member truss structures respectively. In the legend it is shown that the orange line represents the QUBO problem when mapped to

[5] https://docs.dwavesys.com/docs/latest/c_gs_2.html#the-hamiltonian-and-the-eigenspectrum

QAOA from Wils [4] and the blue line represents the QAOA objective worked out in this thesis. The relation between the number of trusses and the gate volume seems to indicate exponential growth for both problems. Higher member truss structures have to be analyzed to review the actual relation between the number of trusses and the gate volume. The relation will certainly change when the number of options would be increased. The gate volume for the two-member, three-member and four-member truss structure for the internal force method is 688, 1962 and 4495 respectively. For the two-member, three-member and four-member truss structure QUBO to QAOA problem the gate volume is 1461, 16540 and 105675.

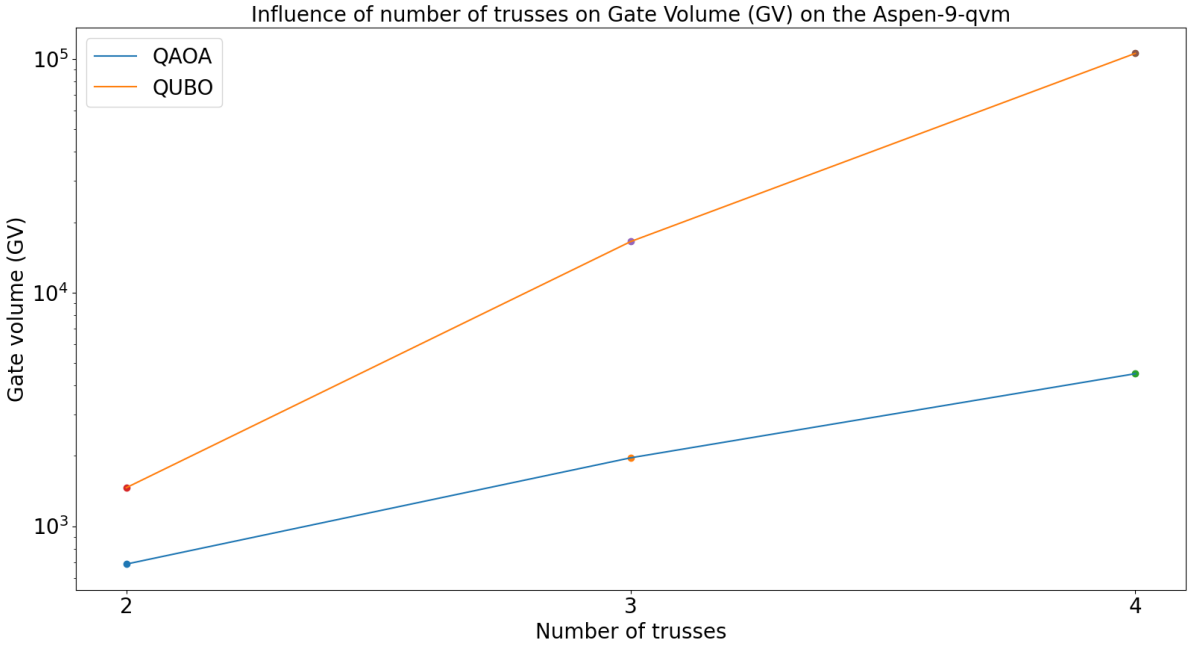


Figure 3.7: Influence of step parameter p on the gate volume before program compilation with Aspen-9-qvm.

4

RESULTS AND DISCUSSION

In this chapter results will be presented and discussed. Firstly, the QAOA results are discussed in Section 4.1. Secondly, the QUBO to QAOA results are presented in Section 4.2.

4.1. RESULTS OF TRUSS STRUCTURES IN QAOA FORMAT

In this section the two-truss, three-truss and four-truss structures will be analyzed one-by-one in Subsection 4.1.1, Subsection 4.1.2 and Subsection 4.1.3 respectively.

4.1.1. RESULTS AND DISCUSSION OF TWO-MEMBER TRUSS STRUCTURE

In this subsection the two-member truss structure is analyzed. The results after optimization are displayed in Figure 4.1. The brute-force solution is plotted with the dotted orange line and the cost is given in blue. It can be observed that the brute-force solution and the cost Hamiltonian completely overlap each other. The minimum and maximum energy are shown with an orange and green plus symbol respectively, as indicated on the legend on the left-hand side. On the right-hand side of the figure the probabilities of finding the minimum are shown. The valid bit strings are plotted on the x-axis. In order to simulate this quantum circuit the Aspen-9-qvm was used. The classical minimization routine SHGO was used as an optimizer method, since it is fast and provides good results as was shown in Subsection 3.4.2.

It is clear from Figure 4.1 that QAOA finds the local minimum of the two-member truss structure, ran with step parameter p is two steps. The same simulation has been run ten times in order to investigate the dependence on the probabilistic nature of quantum. Ten simulations out of ten found the minimum with bit string [1,0,0,1,0,0].

It is important to remember that this minimum does not resemble the optimized area subjected to the strength constraint. To ensure that the residual is equal to a value extremely close to zero the initial areas A have to get updated to the selected values A_0 and A_3 (bit string: [1,0,0,1,0,0]). To obtain the areas that do satisfy the strength constraint, the program has to be iterated as was explained in Section 3.1.

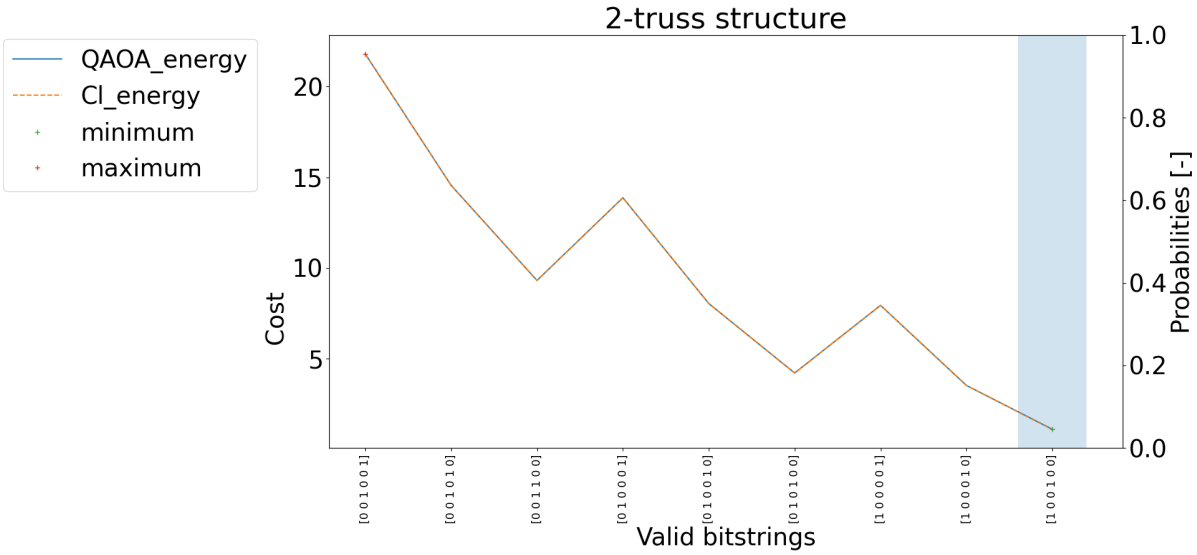


Figure 4.1: QAOA results of the two-member truss structure, simulated on Rigetti's most realistic quantum virtual machine, named Aspen-9-qvm

Running the quantum circuit on real quantum hardware with 10,000 shots, named the Aspen-9, does not provide correct results as shown in Figure 4.2. Due to the noise, the probability of valid solutions significantly decreases and introduces invalid solutions, which are not shown in the figure. Since, it is not possible to obtain correct results when using real quantum hardware for the smallest 2-member truss structure, it is considered not useful to try for the three-truss and four-truss structures. The reason for it being that the number of gates only increases for higher number of trusses as was shown in Section 3.6. It was tried to run the quantum circuit with predefined optimal angles obtained from simulation in order to circumvent the need for VQE, to directly try to obtain the optimum. The result, however is still polluted with noise. When decreasing the number of gates further to 325 by decreasing the step parameter p to one, it results in a similar polluted solution. This result shows the limits of NISQ computers.

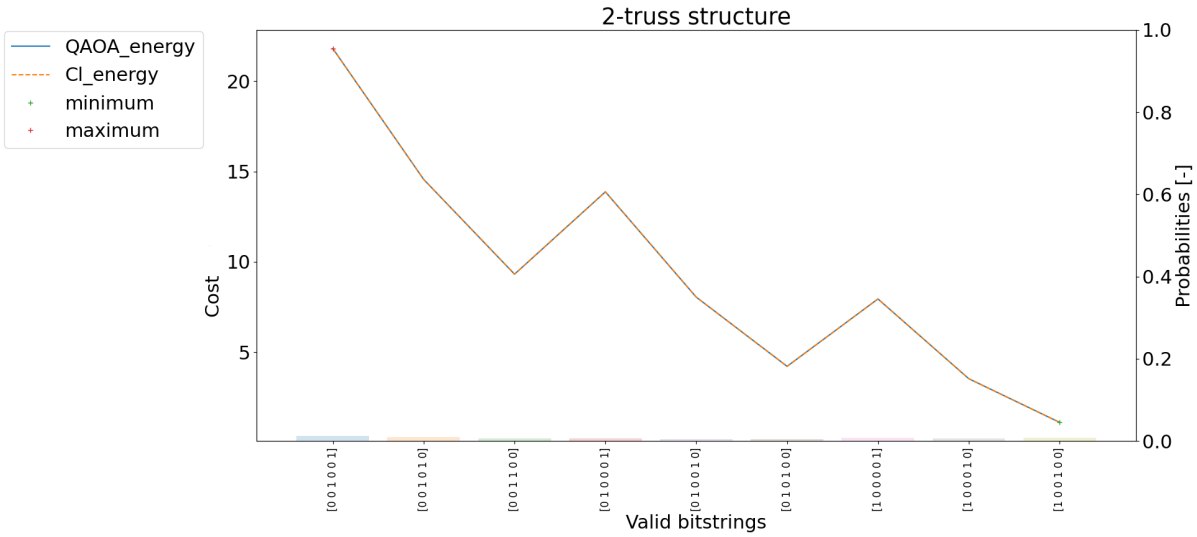


Figure 4.2: QAOA results from real quantum computer, named Aspen-9. The quantum computer did not find a minimum, due to noise caused by high gate volume.

4.1.2. RESULTS AND DISCUSSION OF THREE-MEMBER TRUSS STRUCTURE

In this subsection the simulation results of the three-member truss structure can be observed. After trial-and-error it was found that the step parameter p should be three to find the minimum with a high probability. It can be seen in Figure 4.3 that five out of ten simulations found the minimum that the correct result is found (bit string: [0,0,1,1,0,0,0,0,1]) when simulating on the Aspen-9-qvm. Due to the probabilistic nature of the quantum computers the minimum is not always found, since three out of the ten simulations found the bit string: [0,0,1,0,0,1,0,0,1] and two out of the ten simulations found the bit string: [0,0,1,0,1,0,0,0,1]. It is of no surprise that these other found bit strings resemble values very close to the actual minimum.

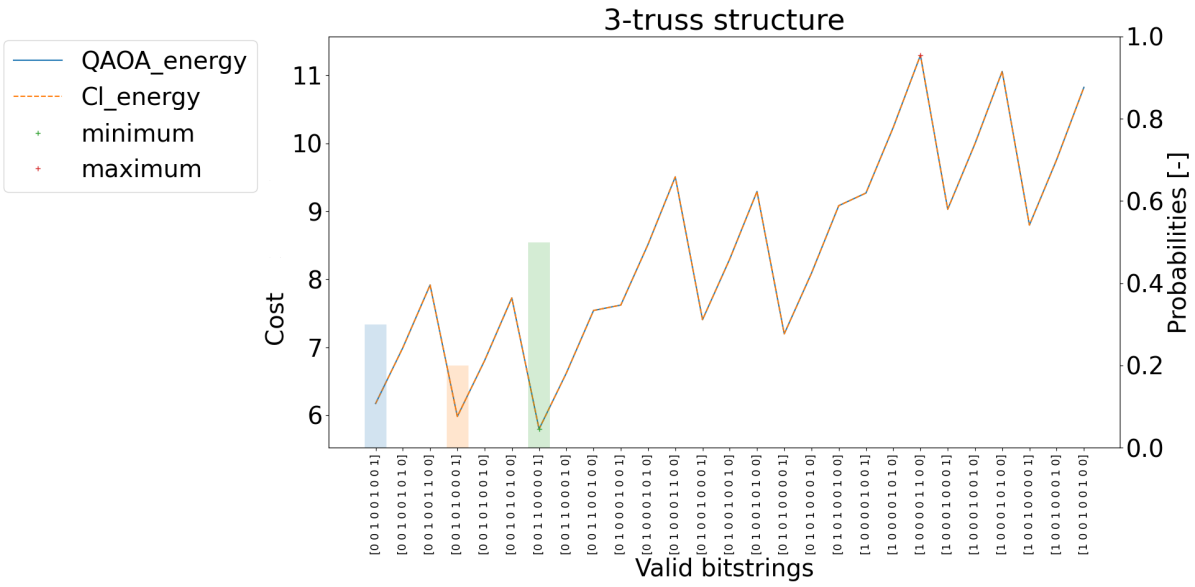


Figure 4.3: QAOA results for the three-member truss structure, simulated on Rigetti's most realistic quantum virtual machine, named Aspen-9-qvm.

4.1.3. RESULTS AND DISCUSSION OF FOUR-MEMBER TRUSS STRUCTURE

In this subsection the simulation results for the four-member truss structure are presented and discussed. The four-member truss structure is only simulated on the 12q-qvm without noise with step parameter p set to four. It was deemed impractical to run it on the Aspen-9-qvm, because it took more than 5,000 seconds to construct and run the circuit just once with 1024 samples. Nevertheless, the algorithm proves to work on the Aspen-9-qvm for the two-member truss and three-member truss structure, it is therefore assumed that it will also work for the four-member truss structure.

As can be seen from Figure 4.4 the probability of finding the minimum is five out of the ten runs. What cannot be seen from the figure is that the difference between the sampled bit strings is not large. Meaning that the found minimum is only barely found, for example the bit string [0,0,1,0,0,1,0,1,0,0,1,0] is sampled 245 times and the bit string [0,0,1,0,0,1,0,0,1,0,0,1] is sampled 235 times out of 1024 samples. Simulating the circuit on the Aspen-9-qvm or simulating with noise will most likely not yield the minimum. This is partly due to the increased feasible subspace, however the main contributing factor to barely finding the minimum is, because the minimum cost value has a value close to the other valid solutions. For this reason, it is harder for the algorithm to find the minimum. A solution could be to distance the minimum from other valid solutions by adding extra Pauli terms to penalize valid solutions, that are not the minimum.

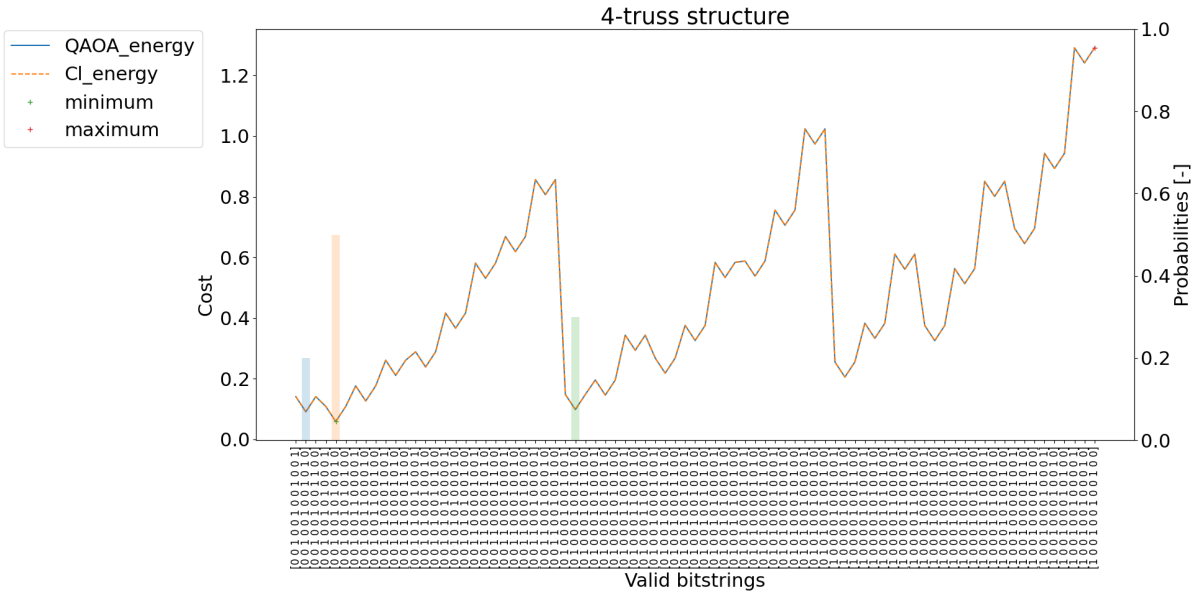


Figure 4.4: QAOA results for the four-member truss structure, simulated on the fully connected quantum virtual machine, named 12q-qvm.

4.2. QUBO TO QAOA

In this section the (in)determinate two-member, three-member and four-member truss structures from Wils [4] will be analyzed one-by-one in Subsection 4.2.1, Subsection 4.2.2 and Subsection 4.2.3 respectively. The truss structures treated in this section are completely different from the ones treated in the previous section, for more details refer to Wils [4]. The QUBO results were used as input in order to map it to QAOA format using the steps provided in Section 3.5. The two-member, three-member and four-member truss structures are not simulated ten times as was done in Section 4.1, because of time and simulation resource constraints. In addition to that, since previous results in Section 4.1 have shown the viability of the chosen approach, the author of this report believes that the use of one simulation result is justified, with the requirement that the found minimum has a high probability.

Due to time and resource constraints, the four-member truss structure is simulated without noise on the fully connected quantum virtual machines, named 12q-qvm. For more information on the connectivity of qubits refer to Subsection 3.4.1.

4.2.1. RESULTS AND DISCUSSION OF TWO-MEMBER TRUSS STRUCTURE

In this subsection the results for the two-member truss structure are presented and discussed. The results for the non-constraining mixer Hamiltonian are shown in Figure 4.5 and the results for the constraining mixer Hamiltonian are shown in Figure 4.6. On the left-hand side of Figure 4.5 a legend is displayed, wherein Ajagekar stands for the iterative method of Ajagekar *et al.* [56] used by Wils [4]. It is not an energy, because energies cannot be negative. It is rather the objective function used required to be able to run it on the Quantum Annealing computer of D-Wave. It can be observed from Figure 4.5 that the QAOA cost and Ajagekar completely overlap, which means that the transformation was successful.

In Figure 4.5 the results are shown with a non-constraining mixer Hamiltonian as earlier described in Section 3.5. The Aspen-9-qvm is a noiseless quantum virtual machine. The probabilities of the valid solutions only add up to 23.6%. Due to the non-constraining mixer Hamiltonian, the full

search space was explored. Thereby, the probabilities of the valid bit strings do not add up to 100%. The minimum was found with a probability of only 20.2%. It is concluded that this approach that penalizes invalid solutions in the cost Hamiltonian does not provide a convincing result with QAOA.

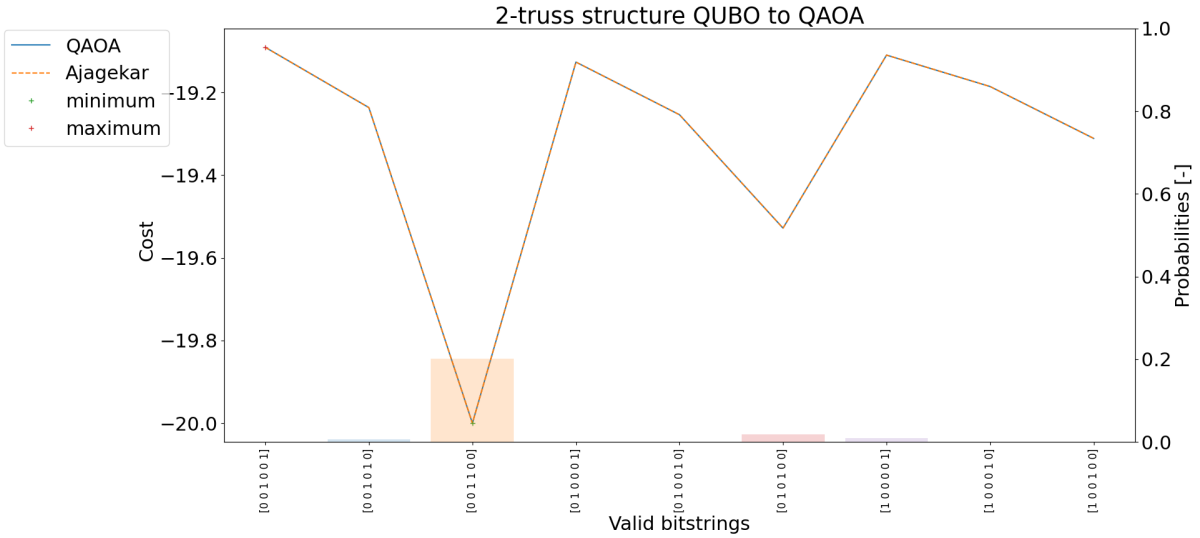


Figure 4.5: QAOA results for the two-member truss structure, simulated on Rigetti's most realistic quantum virtual machine, named Aspen-9-qvm with step parameter p is 3.

In Figure 4.6 the results are shown when using the constrained mixer Hamiltonian as explained in Section 3.5. It can be observed that the exact same result was found as in Wils [4], the minimum is found with a probability of >40% with bit string [0,0,1,1,0,0]. This high probability shows that this approach with constraining mixer Hamiltonian works better with QAOA than the approach where invalid solutions are penalized in the cost Hamiltonian.

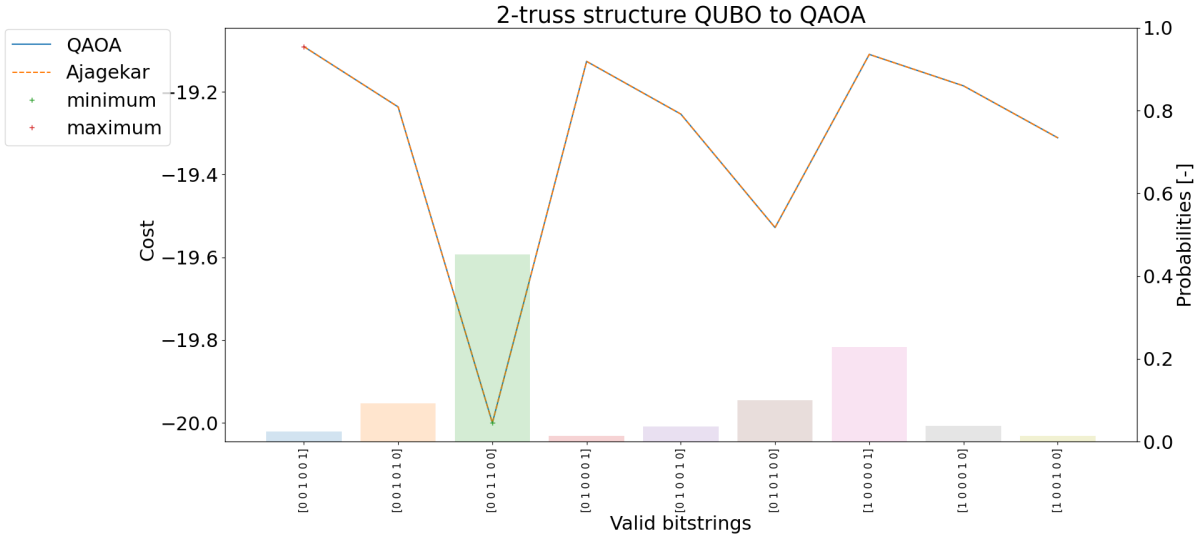


Figure 4.6: QAOA results for the two-member truss structure, simulated on Rigetti's most realistic quantum virtual machine, named Aspen-9-qvm with step parameter p is 3.

4.2.2. RESULTS AND DISCUSSION OF THREE-MEMBER TRUSS STRUCTURE

In this subsection the results for the three-member truss structure are presented and discussed. The results for the non-constraining mixer Hamiltonian are shown in Figure 4.7 and the results for

the constraining mixer Hamiltonian are shown in Figure 4.8. The QAOA algorithm is simulated on the Aspen-9-qvm with step parameter p equal to three. It can be observed that the search space is so large that none of the valid solutions are sampled with the highest probability. This non-constraining mixer Hamiltonian approach clearly does not work for the three-member truss structure. Since, the four-member truss structure involves more gates and a larger search space, the chances of success are significantly lower and will therefore not be tried.

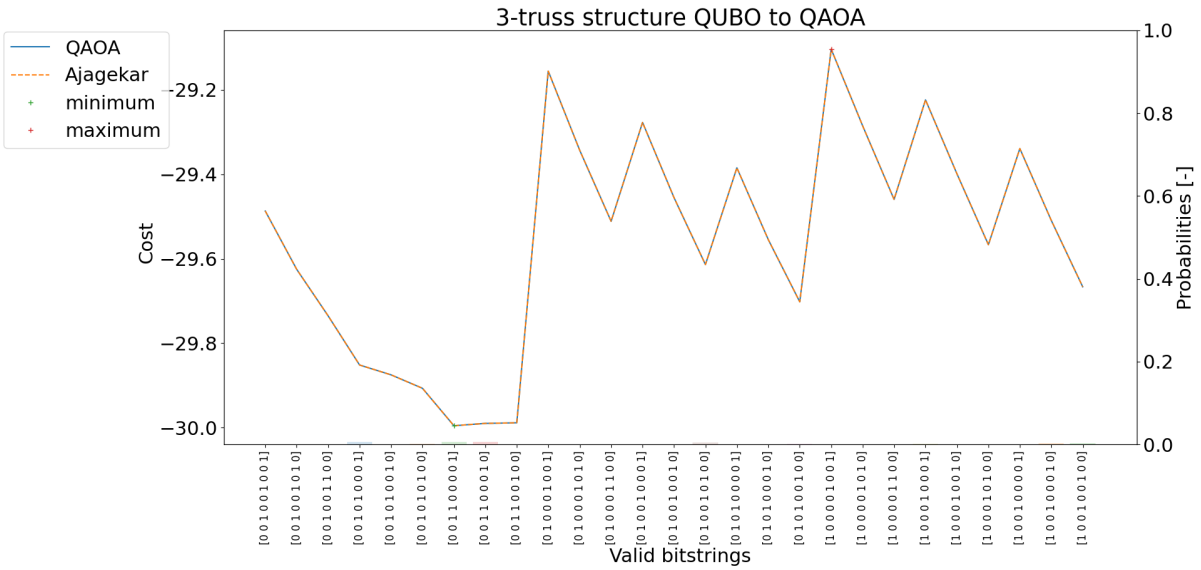


Figure 4.7: QAOA results for the three-member truss structure, simulated on Rigetti's most realistic quantum virtual machine, named Aspen-9-qvm.

In Figure 4.8 the constrained mixer Hamiltonian results are shown. The results for bit strings: [0,0,1,1,0,0,0,0,1], [0,0,1,1,0,0,0,1,0] and [0,0,1,1,0,0,1,0,0] have values that are extremely close to each other. For this reason, the QAOA algorithm does not find the minimum, which is in this case bit string: [0,0,1,1,0,0,0,0,1]. Wils [4] already recognized that it did not matter too much which of these three bit strings is chosen as minimum, as long as the choices for the first two trusses are correct. It can be concluded that finding the minimum with QAOA when other valid solutions are extremely close to this minimum is hard. The reason for that is that the expectation value obtained with the VQE method sums all the counts of the bit strings into one expectation value. The SHGO optimizer will then still find a low expectation value even though it is not the minimum.

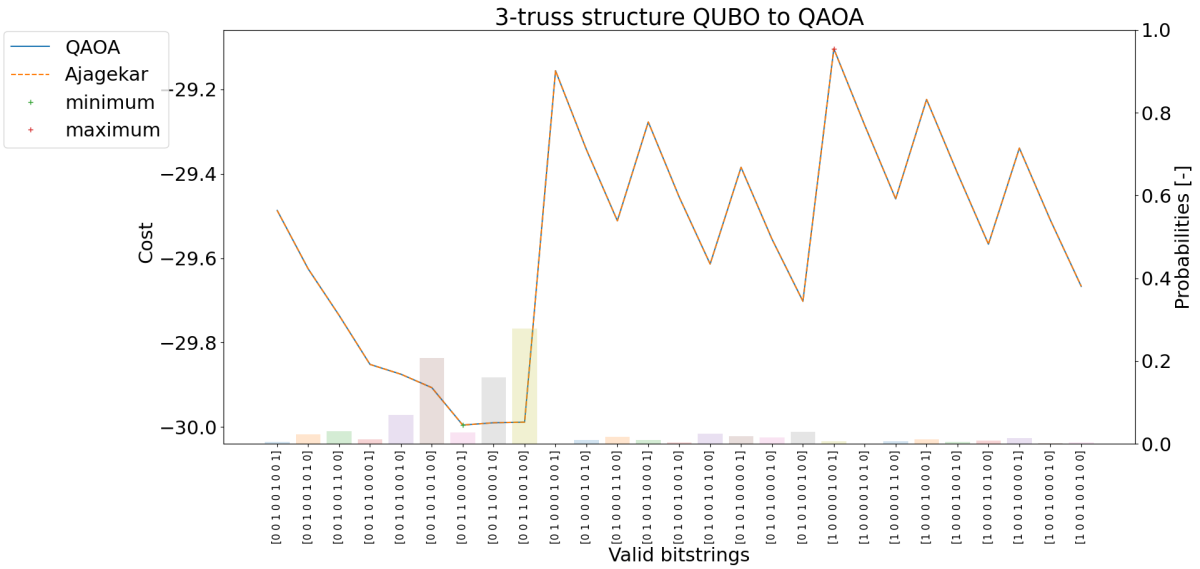


Figure 4.8: QAOA results for the three-member truss structure, simulated on Rigetti's most realistic quantum virtual machine, named Aspen-9-qvm.

4.2.3. RESULTS AND DISCUSSION OF FOUR-MEMBER TRUSS STRUCTURE

In this subsection the results for the four-member truss structure are presented and discussed. The four-member truss structure is simulated with the 12q-qvm, of which the results are presented in Figure 4.9. The computational time to simulate the four-member truss structure is so large that the author of this report is unable to estimate a time figure. Therefore, the WavefunctionSimulator was used to directly obtain the expectation value. It can be observed that the minimum cost value is far distanced from the other valid bit string values. QAOA is, therefore able to find it with a probability of 34.2% when setting step parameter p to four. It should be stressed that, it is not known if it will also provide correct results when simulating with the Aspen-9-qvm. Thereby, Figure 4.9 should rather be interpreted as illustrative rather than definite.

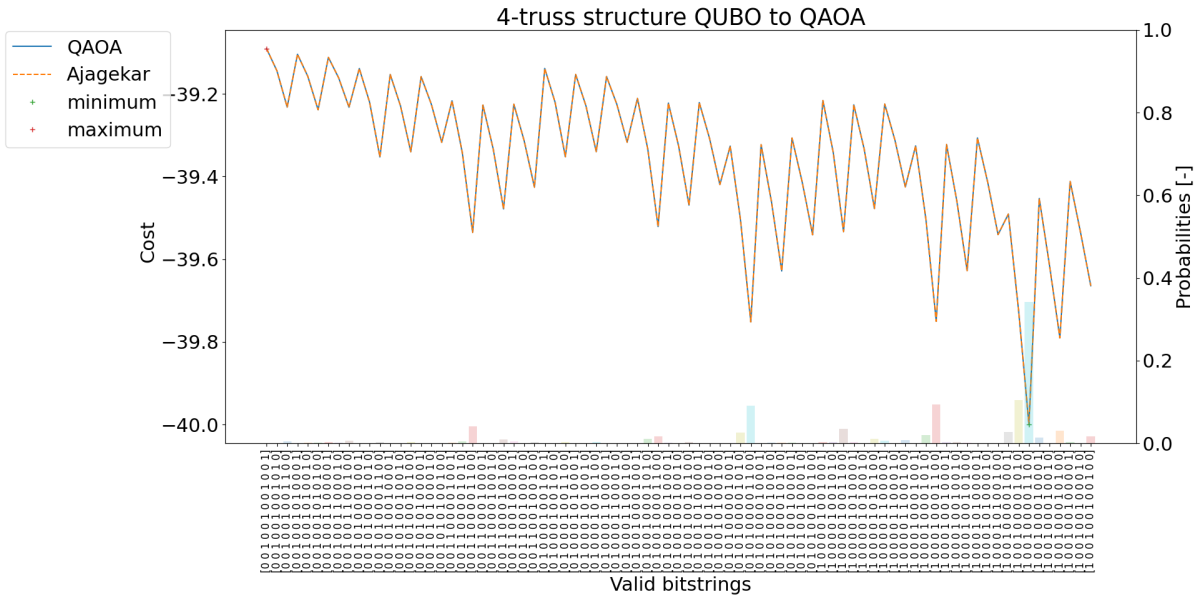


Figure 4.9: QAOA results for the four-member truss structure, simulated on the fully connected quantum virtual machine, named 12q-qvm.

5

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

In this last chapter conclusions will be drawn in Section 5.1. Then the main research question is answered in Section 5.2 by answering the supportive subquestions, which were initially asked in Section 1.2. Finally, recommendations for future work are given in Section 5.3.

5.1. CONCLUSION

For the analyzed two-member, three-member and four-member truss structures, three discrete options per truss member were defined to be utilized by the Quantum Alternating Operator Ansatz (QAOA). When the optimization is finished, then a local minimum is found. The residual has to be equal to or at least be extremely close to zero to satisfy the equilibrium equations. The program should be iterated until the global minimum (residual ≈ 0) is found.

It can be concluded that it is possible to map truss structures to a Quantum Alternating Operator Ansatz (QAOA) objective function and optimize it. The program proves to be working on quantum virtual machines. Currently, however it is not possible to obtain correct results when running on real quantum hardware. The reason for it being that the total noise levels of the real quantum hardware is currently too high for the required gate volume.

Also, mapping Quadratic Unconstrained Binary Optimization (QUBO) from Wils [4] to QAOA format has been shown to be working perfectly. A non-constraining mixer Hamiltonian only shows to be working for the two-member truss structure. It is more effective to make use of a constraining mixer Hamiltonian, since that also proves to be working for the three-member and four-member truss structures.

5.2. ANSWERING THE RESEARCH QUESTION

In this section the main research question is answered by answering the supporting subquestions first.

ANSWERING QUESTION 1

The following sub-questions and sub-sub-questions were asked:

1. How can Gate-Based Quantum Computers (GBQC) be utilized to optimize truss structures?
   (a) How to cast the truss problem to GBQC quantum-suitable format?

(b) How to map a QA problem truss optimization problem from Wils [4] to a QBQC quantum-suitable format?

QAOA can optimize combinatorial optimization problems. In scientific literature the most optimized problem by QAOA is the Maximum Cut problem, which is a graph theory problem treated in Appendix C. To the best of the author's knowledge, QAOA has not been used before to optimize a truss structure.

In order to cast the truss problem into QAOA form, it is required to introduce discrete options for the areas by parameterization. In this thesis, the internal force method is used, where the residuals are squared in order to obtain an objective function. Subsequently, discrete binary variables have to be mapped from discrete variable q $\in \{0,1\}$ to measurement operator $\frac{1}{2}(I-Z)$, where Z is the Pauli-Z gate $\in \{-1,1\}$ and I is the identity gate. Then the objective function is supposed to be minimized to a value extremely close to zero in order to satisfy the equilibrium equations.

QUBO function from Wils [4] is mapped by the same transformation with discrete variable q $\in \{0,1\}$ to measurement operator $\frac{1}{2}(I-Z)$, where Z is the Pauli-Z gate $\in \{-1,1\}$ and I is the identity gate. Subsequently, the mixer Hamiltonian is set to constrain the full search space to a feasible subspace. The initial state is simply an ansatz where all qubits are in $|0\rangle$ state, except the qubits that comprise of the middle area options. As an example, the initial state for the two-member truss structure is $|010010\rangle$.

To conclude, GBQC can be utilized to optimize truss structures by using a quantum algorithm such as QAOA.

ANSWERING QUESTION 2

2. What is the influence on the results when varying some of the parameters of a GBQC?
   (a) What will be the effect on quantum performance when the noise in quantum computers and quantum gates will be decreased?
   (b) What will be the effect on quantum performance when the number of qubits in quantum computers will be increased?

The effect of reducing the noise levels in quantum computers and quantum gates, would result in a couple of things. First and foremost, it would result in better quality solutions. The noise levels in quantum computers nowadays are one of the factors that limit the capacity of GBQC to solve larger problems. The smallest problem, which is the two-member truss structure was not able to be solved on real quantum hardware due to the noise levels. This shows the limits of NISQ computers.

When the number of qubits in quantum computers would be increased, it would open up the way for quantum error correction, which is currently not possible. Furthermore, by introducing more qubits larger optimization problems could be analyzed or more options per truss member could be introduced in the truss structure. Quantum error mitigation techniques were tried in this report, however due to an increase in computational time and gate volume, it was considered unhelpful.

To conclude this subsubsection, the influence on the results when varying some of the parameters are significant. Noise levels hopefully will be decreased in the future in order to be able to optimize different combinatorial optimization problems, such as truss structures.

ANSWERING QUESTION 3

3. How can the results of the quantum computer best be interpreted?

(a) How does the probabilistic nature of quantum computers influence the results?
(b) How can the results from the quantum computer be verified?

Due to the probabilistic nondeterministic nature of quantum computers the QAOA algorithm produces different solutions for the same input. It does, however oftentimes sample the correct results with the highest probability. It becomes harder to find the minimum when the value of the minimum is close to other values. It is not only the nondeterministic nature of quantum computers that influences the results, also all minimization methods (except SHGO) used a random starting angle $\beta \in [0,\pi]$ and $\gamma \in [0,2\pi]$, providing different solutions. Not to forget that real quantum hardware does exhibit noise and will introduce invalid unfeasible solutions even though the QAOA tries to restrict that.

The results of the quantum computer can best be verified by comparing the quantum result with the result obtained after brute-force analysis on the classical computer.

It is not evident to determine if the solution found by the quantum computer or the quantum virtual machine is found by the appropriate Hamiltonians or found by luck due to the probabilistic nondeterministic nature of quantum computers. Initially, the author of this report found the correct solution after simulating, however later came to realize that the solution was correct, however the used cost Hamiltonian was incorrect. It is, therefore important to review the found solutions with scrutiny. The person who constructed the quantum circuit should ask the following questions: is the solution found consecutively after multiple simulations? Is the solution found with different minimization routines? Is this solution found while increasing step parameter p (in order to increase the likelihood of a better approximation of the optimum)?

ANSWERING THE MAIN RESEARCH QUESTION

The research question and its sub-questions are repeated and answered below. The research question of this thesis was formulated as:

Is it feasible to optimize for the cross-sectional areas of a 2D truss stucture with a quantum algorithm on a GBQC?

After providing the answers to the subquestions, it can be concluded that the main research question has been sufficiently answered. It is feasible to optimize for the cross-sectionial areas of a 2D truss structure with a quantum algorithm on a GBQC. Simplifications were made along the way, such as that the tensile stress and compressive stress would equal each other and that the algorithm only solves determinate truss structures. To generalize the truss structure more research has to be done.

5.3. RECOMMENDATIONS AND GENERAL THOUGHTS

In this section, recommendations and general thoughts for future work are given.

The noise levels could potentially be further reduced by giving the compiler hints about commuting gates to reduce the gate depth or by using a python package called pytket to compile the circuit in an optimized format or by using Quil-T from Rigetti to manually optimize the gates. Another approach would be to run the quantum algorithm on different hardware vendors. Possibly the whole program can be implemented in LibKet[57] to provide an easy way to run the quantum circuit on different platforms. To reduce the computational time to simulate higher truss-member structures, the author of this report proposes to parallellize the Python program in order to increase the number of CPU cores utilized to run the simulations on.

Recommendations for future research include improving the program so that it can include indeterminate truss structures. Possibly this can be implemented by using Maxwell-Betti's theorem. The fraction that will subsequently arise can be solved with Agajekar's iterative method. That would defeat part of the purpose of optimizing it with the parameterized internal force method in the first place, which was to avoid the fractional part. An advantage of the parameterized internal force method is, however that the classical processing time to obtain the expressions is low when compared to the QUBO method. Another approach would be to use parameterized FEM as Wils [4] did, however without penalizing the invalid solutions and instead use a constraining mixer Hamiltonian. The fractions that arise can again be solved with Agajekar's iterative method. This would significantly reduce the number of gates required. A potential way of decreasing the step parameter p is by penalizing valid solutions, that do not resemble the minimum, by adding Pauli-Z gates to higher cost. This might be difficult to implement without increasing the gate volume by such an amount that the decrease of gate volume by reducing p exceeds the initial gate volume.

Iterations are required to reach a global minimum of zero or a value extremely close to zero, such that the equilibrium equations are satisfied. Perhaps there is a smart way of adjusting the dA parameter, used to calculate A-dA and A+dA, while iterating, such that the global optimum is found faster. Another way of decreasing the number of iterations is by increasing the number of options per truss member. A disadvantage of this method is that the gate volume and subsequently the noise will increase, which will decrease the possibility of success.

With this work to prove feasibility the Technology Readiness Level (TRL) is estimated to be around three. Further research has to be performed to validate the experiment on a real quantum computer to increase the TRL to four.

It would be interesting to investigate other types of problems. For example, to expand the chosen approach to beam structures modeled with beam elements or plate structures modeled with shell elements. These problems can be tackled with two different approaches, the internal force method and displacement based FEM. Regardless of the chosen approach, the author of this report expects that the number of gates will approximately double for every added unconstrained Degree-of-Freedom, based on the observation that the relation between gate volume and truss member seemed to double for every added truss member.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor Dr. Boyang Chen who gave me the opportunity to work on this challenging thesis topic. Our cooperation has always been pleasant, supportive and encouraging. I would like to thank Dr. Matthias Möller for joining this project with a lot of enthusiasm and seemingly endless energy. Both my supervisors were invaluable to the completion of this thesis with their support and guidance on all aspects of this thesis work and with their creativity and out-of-the-box thinking. I would also like to thank my committee members, Prof. Dr. ir. S. van der Zwaag and Dr. ir. M.I. Gerritsma for assessing my work and giving me feedback. Furthermore, I would like to thank Giorgio Tosti Balducci who was so friendly to take the time to update me on his work and vice versa. One of our conversations has been one of the keys to find the solution to this problem. Also, I would like to thank Joost Bus and Kevin Wils who explained and shared their work with me. Lastly, I would like to thank my fellow students that have proofread my thesis.

Finally, I would like to express my gratitude to family and friends that encouraged me to practice persistence and diligence in order to accomplish important things in life.

BIBLIOGRAPHY

[1] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, *Characterizing quantum supremacy in near-term devices,* Nature Physics **14**, 595 (2018), arXiv:1608.00263, 2017 .

[2] J. Preskill, *Quantum Computing in the NISQ era and beyond,* Quantum **2**, 79 (2018), arXiv:1801.00862 .

[3] M. I. Dyakonov, *When will we have a quantum computer?* (2019), arXiv:1903.10760 [quant-ph] .

[4] K. Wils, *Quantum computing for structural optimization,* (2020).

[5] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2010).

[6] D. Dervovic, M. Herbster, P. Mountney, S. Severini, N. Usher, and L. Wossnig, *Quantum linear systems algorithms: a primer,* (2018), arXiv:1802.08227 .

[7] K. M. Svore, M. B. Hastings, and M. Freedman, *Faster phase estimation,* (2013), arXiv:1304.0741v1 [quant-ph] .

[8] D. J. Griffiths, *Introduction to Quantum Mechanics* (Prentice Hall, Inc., 1995) pp. 1–409.

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

[10] IBM research blog, (Accessed on 08 September 2020), https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/ .

[11] K. Svozil, *Comment on "quantum supremacy using a programmable superconducting processor",* (2019), arXiv:1911.00577v1 [quant-ph] .

[12] S. D. Sarma, M. Freedman, and C. Nayak, *Topological quantum computation,* Physics Today July 2006, page 32-38 **32** (2019), 10.1063/1.2337825.

[13] P. W. Shor, *Scheme for reducing decoherence in quantum computer memory,* Phys. Rev. A **52**, R2493 (1995).

[14] A. M. Steane, *Error correcting codes in quantum theory,* Phys. Rev. Lett. **77**, 793 (1996).

[15] J. Preskill, *Battling Decoherence : The Fault-Tolerant Quantum Computer,* Physics Today (1999), 10.1063/1.882692.

[16] D. Gottesman, *An introduction to quantum error correction and fault-tolerant quantum computation,* (2009), arXiv:0904.2557 [quant-ph] .

[17] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, *Adiabatic quantum computation is equivalent to standard quantum computation,* SIAM Review **50**, 755 (2008).

[18] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, E. W. Draeger, E. T. Holland, and R. Wisnieff, *Pareto-efficient quantum circuit simulation using tensor contraction deferral,* (2020), arXiv:1710.05867 [quant-ph] .

[19] S. Aaronson and L. Chen, *Complexity-Theoretic Foundations of Quantum Supremacy Experiments,* in *32nd Computational Complexity Conference (CCC 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 79, edited by R. O'Donnell (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017) pp. 22:1–22:67.

[20] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, *A variational eigenvalue solver on a photonic quantum processor,* Nature Communications **5** (2014).

[21] E. Farhi, J. Goldstone, and S. Gutmann, *A Quantum Approximate Optimization Algorithm,* (2014), arXiv:1411.4028 .

[22] D. Wang, O. Higgott, and S. Brierley, *Accelerated variational quantum eigensolver,* Physical Review Letters **122** (2019).

[23] Harrow, Aram W. and Hassidim, Avinatan and Lloyd, Seth, *Quantum algorithm for linear systems of equations,* Physical Review Letters **103**, 1 (2009).

[24] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subaşı, L. Cincio, and P. J. Coles, *Variational quantum linear solver: A hybrid algorithm for linear systems,* Bulletin of the American Physical Society (2020).

[25] E. Farhi, J. Goldstone, and S. Gutmann, *A numerical study of the performance of a quantum adiabatic evolution algorithm for satisfiability,* (2000), arXiv:quant-ph/0007071 [quant-ph] .

[26] Y. Subaşl, R. D. Somma, and D. Orsucci, *Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing,* Physical Review Letters **122**, 1 (2019), arXiv:1805.10549 .

[27] L. Vandenberghe and S. Boyd, *Semidefinite programming,* Society for Industrial and Applied Mathematics **38**, 49 (1996).

[28] S. Lloyd, M. Mohseni, and P. Rebentrost, *Quantum principal component analysis,* Nature Physics **10**, 631–633 (2014).

[29] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Quantum machine learning,* Nature **549**, 195–202 (2017).

[30] M. Schuld and N. Killoran, *Quantum machine learning in feature hilbert spaces,* Phys. Rev. Lett. **122**, 040504 (2019).

[31] K. P. Murphy, *Machine Learning: A Probabilistic Perspective* (Massachusetts Institute of Technology, 2012) pp. 1–1098.

[32] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd, *Quantum gradient descent and newton's method for constrained polynomial optimization,* New Journal of Physics **21**, 073023 (2019).

[33] P. J. J. O'Malley, R. Babbush, I. D. Kivlichan, J. Romero, J. R. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, P. V. Coveney, P. J. Love, H. Neven, A. Aspuru-Guzik, and J. M. Martinis, *Scalable quantum simulation of molecular energies,* Phys. Rev. X **6**, 031007 (2016).

[34] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets,* Nature **549**, 242–246 (2017).

[35] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, and et al., *Towards quantum chemistry on a quantum computer,* Nature Chemistry **2**, 106–111 (2010).

[36] J. Du, N. Xu, X. Peng, P. Wang, S. Wu, and D. Lu, *Nmr implementation of a molecular hydrogen quantum simulation with adiabatic state preparation,* Phys. Rev. Lett. **104**, 030502 (2010).

[37] Y. Wang, F. Dolde, J. Biamonte, R. Babbush, V. Bergholm, S. Yang, I. Jakobi, P. Neumann, A. Aspuru-Guzik, J. D. Whitfield, and et al., *Quantum simulation of helium hydride cation in a solid-state spin register,* ACS Nano **9**, 7769–7774 (2015).

[38] S. Paesani, A. Gentile, R. Santagati, J. Wang, N. Wiebe, D. Tew, J. O'Brien, and M. Thompson, *Experimental bayesian quantum phase estimation on a silicon photonic chip,* Physical Review Letters **118** (2017), 10.1103/physrevlett.118.100503.

[39] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, M. Y. Niu, R. Halavati, E. Peters, M. Leib, A. Skolik, M. Streif, D. Von Dollen, J. R. McClean, S. Boixo, D. Bacon, A. K. Ho, H. Neven, and M. Mohseni, *TensorFlow Quantum: A Software Framework for Quantum Machine Learning,* (2020), arXiv:2003.02989 .

[40] 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 **12**, 1 (2019), arXiv:1709.03489 .

[41] G. G. Guerreschi and A. Y. Matsuura, *Qaoa for max-cut requires hundreds of qubits for quantum speed-up,* Scientific Reports **9** (2019).

[42] 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 **10**, 1 (2020), arXiv:1812.01041 .

[43] P. Vikstål, M. Grönkvist, M. Svensson, M. Andersson, G. Johansson, and G. Ferrini, *Applying the quantum approximate optimization algorithm to the tail-assignment problem,* Phys. Rev. Applied **14**, 034009 (2020).

[44] A. Borle, V. E. Elfving, and S. J. Lomonaco, *Quantum approximate optimization for hard problems in linear algebra,* (2021), arXiv:2006.15438 [quant-ph] .

[45] G. Verdon, M. Broughton, and J. Biamonte, *A quantum algorithm to train neural networks using low-depth circuits,* (2017), arXiv:1712.05304 .

[46] M. Alam, A. Ash-Saki, and S. Ghosh, *Analysis of quantum approximate optimization algorithm under realistic noise in superconducting qubits,* (2019), arXiv:1907.09631 [quant-ph] .

[47] X. Xue, B. Patra, J. P. G. van Dijk, N. Samkharadze, S. Subramanian, A. Corna, B. Paquelet Wuetz, C. Jeon, F. Sheikh, E. Juarez-Hernandez, B. P. Esparza, H. Rampurawala, B. Carlton, S. Ravikumar, C. Nieva, S. Kim, H.-J. Lee, A. Sammak, G. Scappucci, M. Veldhorst, F. Sebastiano, M. Babaie, S. Pellerano, E. Charbon, and L. M. K. Vandersypen, *CMOS-based cryogenic control of silicon quantum circuits,* Nature **593**, 205–210 (2021).

[48] Rigetti Computing, *pyQuil Documentation: Simulators,* (2020).

[49] S. Endres, C. Sandrock, and W. Focke, *A simplicial homology algorithm for lipschitz optimisation,* Journal of Global Optimization **72** (2018), 10.1007/s10898-018-0645-y.

[50] R. LaRose, A. Mari, S. Kaiser, P. J. Karalekas, A. A. Alves, P. Czarnik, M. E. Mandouh, M. H. Gordon, Y. Hindy, A. Robertson, P. Thakre, N. Shammah, and W. J. Zeng, *Mitiq: A software package for error mitigation on noisy quantum computers,* (2021), arXiv:2009.04417 [quant-ph] .

[51] P. A. M. Casares, R. Campos, and M. A. Martin-Delgado, *QFold: Quantum Walks and Deep Learning to Solve Protein Folding,* (2021), arXiv:2101.10279 .

[52] A. Sopena, M. H. Gordon, G. Sierra, and E. López, *Simulating quench dynamics on a digital quantum computer with data-driven error mitigation,* Quantum Science and Technology **6**, 045003 (2021).

[53] 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 (Association for Computing Machinery, New York, NY, USA, 2017) p. 15–21.

[54] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, *Improving variational quantum optimization using cvar,* Quantum **4**, 256 (2020).

[55] I. Hen and F. M. Spedalieri, *Quantum annealing for constrained optimization,* Phys. Rev. Applied **5**, 034007 (2016).

[56] A. Ajagekar, T. Humble, and F. You, *Quantum computing based hybrid solution strategies for large-scale discrete-continuous optimization problems,* Computers & Chemical Engineering **132**, 106630 (2020).

[57] M. Möller and M. Schalkers, *: A cross-platform programming framework for quantum-accelerated scientific computing,* in *Computational Science – ICCS 2020,* edited by V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira (Springer International Publishing, Cham, 2020) pp. 451–464.

[58] J. Bus, *The Quantum Approximate Optimization Algorithm: Performance on Max-Cut using Heuristic Parameter determination,* (2020).

A

INTRO TO QC

A.1. GATE-BASED QUANTUM COMPUTING (GBQC)

A quantum state $|\psi\rangle$ can be described as a linear combination, often named superposition, of two computational basis states, $|0\rangle$ and $|1\rangle$, consisting of complex amplitudes α_0 and α_1:

$$|\psi\rangle = \alpha_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \alpha_0|0\rangle + \alpha_1|1\rangle. \tag{A.1}$$

Because P=$|\alpha_0|^2 + |\alpha_1|^2 = \sum_{n=0}^{1} |\alpha_n|^2 = 1$, Equation (A.1) can be rewritten as:

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle) \tag{A.2}$$

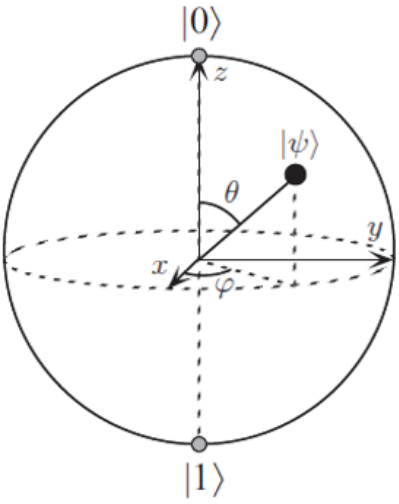
Global phases cannot be measured on a quantum computer and for this reason, Equation (A.2)[5] can be also represented as:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle \tag{A.3}$$

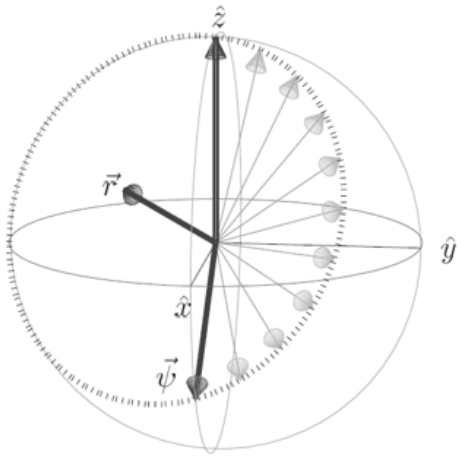
In Figure A.1a a representation of this quantum state $|\psi\rangle$ from Equation (A.3)[5] is shown in the so-called Bloch sphere. In Figure A.1b [5] the influence of applying quantum gates to a qubits state ψ is represented in the Bloch sphere.

In layman terms, the potential beauty of quantum computing is that 2^n states are encoded into this superposition state ψ, n being the number of quantum bits. So, the computational power doubles by adding one bit only! By applying gates the quantum state ψ can be altered as shown in Figure A.1b, such that the probability of measuring certain bitstrings is either increased or decreased, where favourable states should be increased. Measurement of qubits are performed at the end of the quantum circuit, since the superposition state collapses to a classical state upon measurement. The promise of quantum computing is that it can obtain solutions to NP-hard and NP-complete problems much faster than classical computing. Since, GBQC currently exhibits noisy effects it is required to repeat the experiment with a number of n shots.

A quantum X-gate shown in Equation (A.4) acting on a qubit in $|0\rangle$ state and its effect is shown in Equation (A.5). Often the X-gate is compared with the logical NOT-gate. Other commonly used



(a) Bloch sphere representation of quantum state $|\psi\rangle$. Figure retrieved from Mathsisfun[1].



(b) Influence of gates on quantum state $|\psi\rangle$. Figure retrieved from Nielsen and Chuang [5]

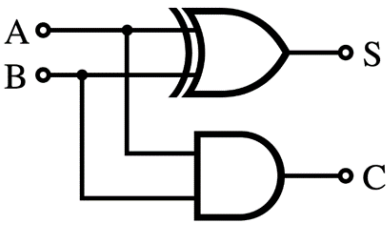
Figure A.1

gates are shown in Table A.1.

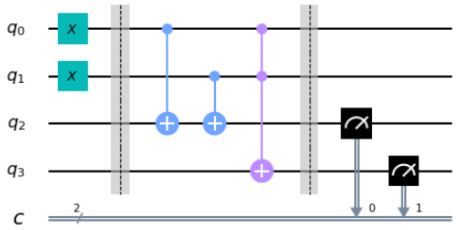
$$|0\rangle \text{ —}\boxed{X}\text{— } |1\rangle \tag{A.4}$$

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{A.5}$$

In Figure A.2a a logical gate sequence is compared with its quantum counterpart in Figure A.2b. Logical gates are irreversible, since it is not possible to retrieve the initial state AB state after applying for example an OR-gate. Quantum gates are reversible, since the initial state is not destroyed, all the information is still in the quantum state and so by appling the inverse of the earlier applied gates, it is possible to retrieve the initial state.



(a) Logical gates - irreversible

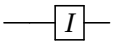
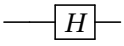
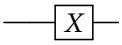
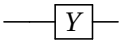
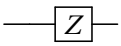
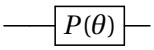
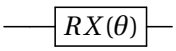
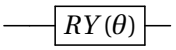
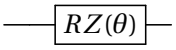
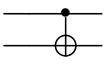
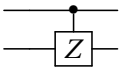
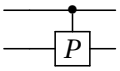
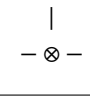


(b) Quantum gates - reversible

Figure A.2: Figures retrieved from Qiskit[2].

[1] https://www.mathsisfun.com/physics/bra-ket-notation.html at 19 April 2021.
[2] https://qiskit.org/textbook/ch-states/atoms-computation.html at 19 April 2021.

Table A.1: Commonly used gates

Name	Matrix	Circuit
Identity	$\mathrm{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	
Hadamard	$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	
Pauli X	$\mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	
Pauli Y	$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	
Pauli Z	$\mathrm{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	
Phase	$P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$	
Rotation X	$R_x(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$	
Rotation Y	$R_y(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$	
Rotation Z	$R_z(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$	
Controlled NOT (Pauli X)	$CX(0,1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	
Controlled (Pauli Z)	$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	
Controlled phase	$CP(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{bmatrix}$	
SWAP	$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	
ISWAP	$XY = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\frac{\phi}{2} & 1j*sin\frac{\phi}{2} & 0 \\ 0 & 1j*sin\frac{\phi}{2} & cos\frac{\phi}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	

B

MATHEMATICAL REFERENCE WORK

B.1. MAPPING BINARY VARIABLE Q TO QUANTUM MEASURABLE FORMAT

Quantum computers measure in the computational basis ($|0\rangle$ and $|1\rangle$) by applying the Pauli-Z gate, with eigenvalues 1 and -1. Some discrete binary variable q acts as a switch, so $q \in \{0,1\}$. In order to map this to a measurement operator that is suitable to a quantum computer, the following mapping is applied: $q \leftrightarrow \frac{1}{2}(I-Z)$, where I is the identity gate. It is simple to show that the above substitution is valid as $q = \frac{1}{2}(1-1) = 0$ and $q = \frac{1}{2}(1--1) = 1$. Alternatively one could verify it with this mapping: Z = 1-2q, so that $Z = 1-2\cdot 0 = 1$ and $Z = 1-2\cdot 1 = -1$.

B.2. COST HAMILTONIAN FOR THE MAX-CUT PROBLEM

To obtain the cost Hamiltonian H_C the following procedure is used:
Cost operator C consists of $C_{\langle jk\rangle}$, where every $C_{\langle jk\rangle}$ entails the predicate $z_j \oplus z_k$, where $\oplus$ represents the exclusive or (xor) gate
Performing some algebra:

$$\overline{z_j} z_k + z_j \overline{z_k} = \tag{B.1}$$
$$(1-z_j)z_k + z_j(1-z_k) = \tag{B.2}$$
$$z_j + z_k - 2z_j z_k \tag{B.3}$$

Substituting $\frac{1}{2}(I-Z)$ for binary variable z results in:

$$\frac{1}{2}(I-Z_j) + \frac{1}{2}(I-Z_k) - 2\cdot\frac{1}{2}(I-Z_j)\frac{1}{2}(I-Z_k) = \tag{B.4}$$
$$\frac{1}{2}(I-Z_j) + \frac{1}{2}(I-Z_k) - 2(\frac{1}{4}I - \frac{Z_j}{4} - \frac{Z_k}{4} + \frac{Z_j Z_k}{4}) = \tag{B.5}$$
$$\frac{1}{2}(I-Z_j Z_k) \text{ or equivalently } \frac{1}{2}(I-\sigma_j^z \sigma_k^z), \tag{B.6}$$
$$\text{where } Z_i \text{ and } \sigma_i^z \text{ correspond to the earlier described Pauli-Z gates} \tag{B.7}$$

B.3. SWAP GATE

An alternative way to describe the swap gate is by:
$\frac{1}{2}(I + XX + YY + ZZ)$ according to Hen and Spedalieri [55] and Hadfield *et al.* [53]

Verifying:

$$II = I \otimes \mathrm{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 0 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 0 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 1 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$X_0X_1 = X_0 \otimes X_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 \times \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & 1 \times \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ 1 \times \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & 0 \times \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$Y_0Y_1 = Y_0 \otimes Y_1 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = \begin{bmatrix} 0 \times \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & -i \times \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \\ i \times \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & 0 \times \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}$$

$$Z_0Z_1 = Z_0 \otimes Z_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \times \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \times \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ 0 \times \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & -1 \times \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Adding $\frac{1}{2}(I + XX + YY + ZZ)$, the following matrix is obtained: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$,

which is equivalent to the SWAP gate.

Since I and ZZ commute with the cost Hamiltonian, only XX and YY are required to cycle through the states.

C

MAX-CUT EXAMPLE

QAOA solves combinatorial optimization problems. It was first illustrated to solve the Maximum Cut or Max-Cut problem by Farhi *et al.* [21]. The problem is defined as a Graph G consisting of vertices $|V|$ of size n and an edge set $|E|$ (or $\{\langle jk\rangle\}$) of size m, so G = (V, E). The goal is to maximize the number of edges with one vertex in the set V and the other in V \ V', where V is partitioned in V and V \ V' as described by Hadfield *et al.* [40]. An example (butterfly) graph to be cut is shown in Figure C.1.

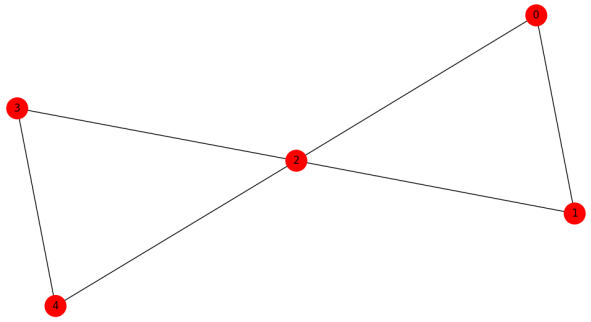


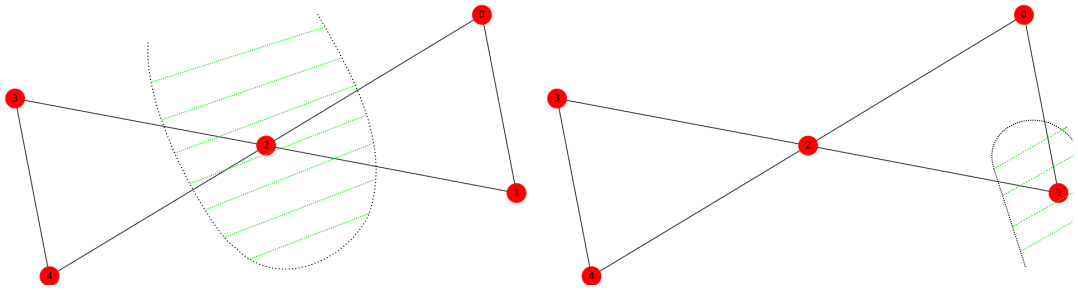
Figure C.1: Butterfly graph. To produce this figure code was used from Qiskit[1].

Two ways of cutting the butterfly graph are shown in Figure C.2a and in Figure C.2b. Bit string 00100 entails a four edge cut (cost of 4) and bit string 01000 entails an associated cost of 2. The cost Hamiltonian (H_C) is defined as follows, which was verified in Appendix B.2: $C = \sum_{\langle jk\rangle} C_{\langle jk\rangle}$, with $C_{\langle jk\rangle} = \frac{1}{2}(I - \sigma_j^z \sigma_k^z)$, where the unitary cost operator is: $U(C,\gamma) = e^{-i\gamma C} = \prod_{\langle jk\rangle} e^{-i\gamma \frac{1}{2}(I-\sigma_j^z\sigma_k^z)}$. In Figure C.3 the cost is plotted as a function of bit string i.

Figure C.4 illustrates the superposition state $|s\rangle$, see Equation (C.1), of the different bit strings when applying one Hadamard gates to each qubit. As can be seen every state has equal number of counts, or in other words, equal probability.

$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} |z\rangle \tag{C.1}$$

[1] https://qiskit.org/textbook/ch-applications/qaoa.html



(a) Max-Cut with bit string 00100.

(b) Max-Cut with bit string 01000.

Figure C.2

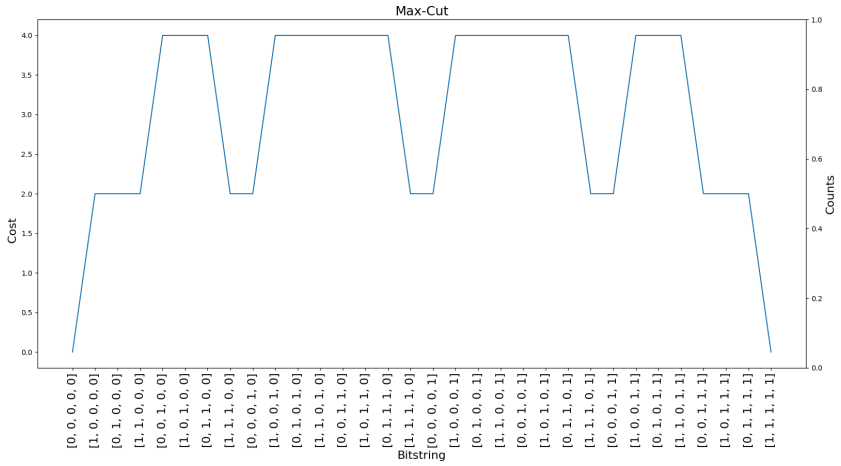


Figure C.3: Cost plotted as function of bit string i

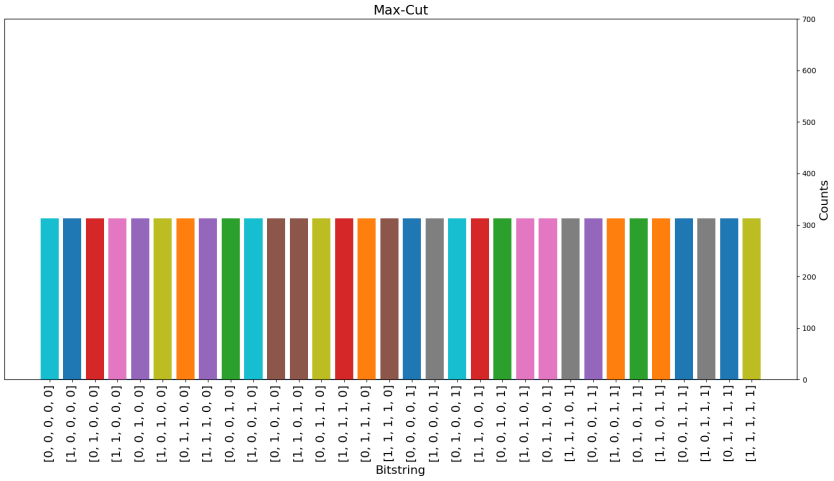


Figure C.4: Superposition

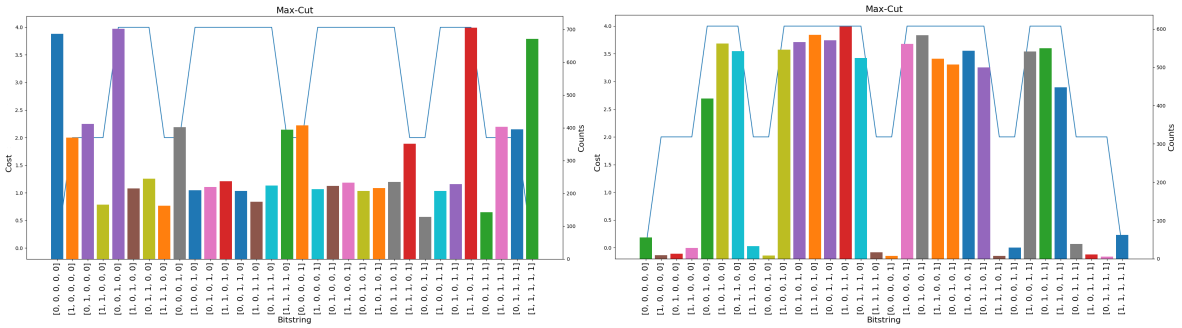
Since all states are feasible, the problem is unconstrained and for this reason, the mixer Hamiltonian $H_B = \mathrm{B} = \sum_{i=1}^{n} \sigma_i^x$, where σ_i^x is the Pauli-X gate. The unitary operator is: $\mathrm{U}(\mathrm{B}, \beta) = \prod_{i \in |V|} e^{-i\beta\sigma_i^x}$. It is important to know that for the Max-Cut problem other mixing and initial states are possible.

A different distribution of counts is shown, where non-optimized β and γ angles for favoring low cost states are shown in Figure C.5a and optimized β and γ angles for maximum expectation value

favoring high cost states are shown in Figure C.5b. After simulating with 10000 shots for the optimized β and γ angles, there is a distribution of counts, for example:

Counts = {'11000': 103, '01010': 231, '00100': 751, '01101': 228, '11111': 635, '01011': 244, '10011': 224, '00101': 222, '01111': 363, '10001': 238, '10010': 220, '00011': 127, '10000': 376, '11101': 363, '10101': 215, '10111': 392, '00010': 415, '11011': 778, '01000': 383, '00000': 632, '11110': 381, '11001': 227, '10100': 217, '00110': 225, '00111': 129, '11010': 233, '01100': 224, '00001': 399, '10110': 240, '01001': 222, '01110': 237, '11100': 126}
With Equation (C.2) the expectation associated with this distribution of count bit strings can be calculated. Ranging γ from 0 to 2π and β from 0 to π with 100 discretized steps results in Figure C.6.

$$\text{Expectation} = \sum_{i=0}^{n-1} \text{Counts}[i] * H_C[i] \tag{C.2}$$



(a) Non-optimized β and γ angles for maximum expectation value favoring low cost states.

(b) Optimized β and γ angles for maximum expectation value favoring high cost states.

Figure C.5: Count distribution as a function of bit strings i

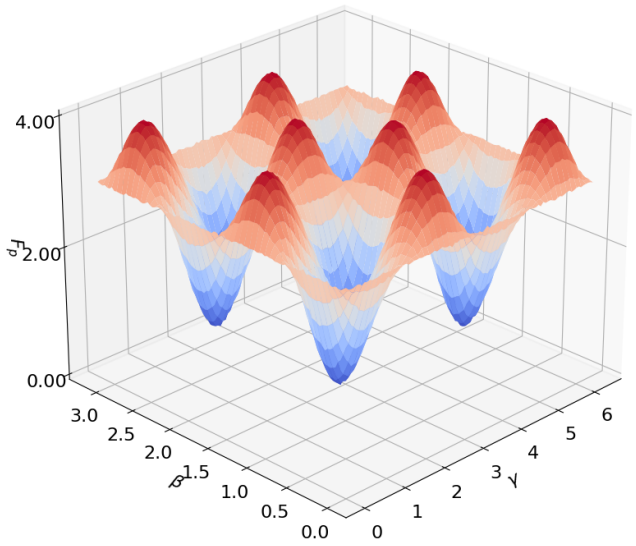


Figure C.6: 3D plot of expectation F for parameters β and γ. To produce this figure, code was adapted from Bus [58].

D

REFERENCE SOLUTIONS TO THE 2D TRUSS PROBLEM

There are different methods to solve a determinate truss problem, namely solve it by: hand, Finite Element Method (FEM) and matrix inversion (solve $\vec{F_{int}}$ from $M\vec{F_{int}} = \vec{F_{ext}}$). The solutions can be obtained trivially when using the classical methods described above in continuous form. These classical methods are used as a reference to provide a solution to compare the quantum result with.

SOLVING TRUSS PROBLEM BY HAND

The three-member truss problem, shown in Figure 3.1b, serves as an example to show the reference solution for this thesis. Firstly, a Free-Body Diagram (FBD) was made and shown in Figure D.1.

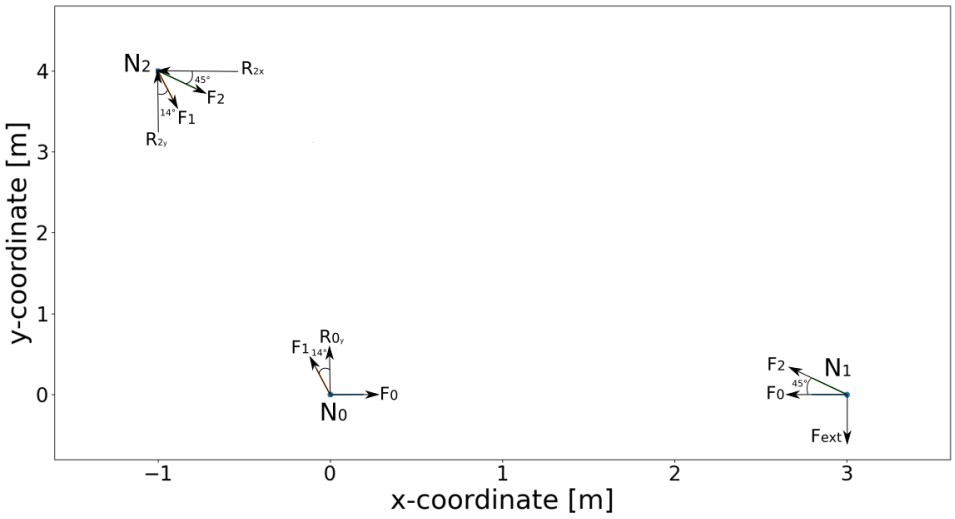


Figure D.1: Free-body diagram of the three truss problem.

After which the sum of the forces in x and y are taken for each node/joint, where C stands for compression and T for tension:

Joint 0:

$\overset{+}{\rightarrow} \Sigma F_x = 0; \quad -F_1 \sin 14.036° + F_0 = 0 \qquad F_1 = 8246.2\text{N (C)}$

$+\uparrow \Sigma F_y = 0; \quad F_1 \cos 14.036° - R_{0y} = 0 \qquad R_{0y} = 8000\text{N (T)}$

Joint 1:

$\overset{+}{\rightarrow} \Sigma F_x = 0; \quad -F_0 - F_2 \cos 45° = 0 \qquad F_0 = 2000\text{N (C)}$

$+\uparrow \Sigma F_y = 0; \quad F_2 \sin 45° - 2000\text{N} = 0 \qquad F_2 = 2828.4\text{N (T)}$

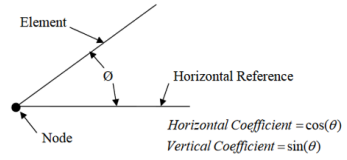
Joint 2:

$\overset{+}{\rightarrow} \Sigma F_x = 0; \quad -R_{2x} + F_1 \sin 14.036° + F_2 \cos 45° = 0 \qquad R_{2x} = 0\text{N}$

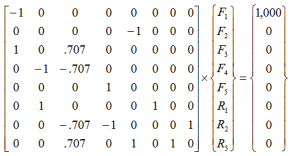
$+\uparrow \Sigma F_y = 0; \quad R_{2y} - F_1 \cos 14.036° - F_2 \sin 45° = 0 \qquad R_{2y} = 6000\text{N (C)}$

The stress state is then obtained by dividing the internal force by the area of the truss-member: $\sigma = \frac{F}{A}$. Subsequently, it is compared with the maximum stress that the truss can experience in compression or tension. The Reserve Factor (RF) = $\frac{\sigma_{max}}{\sigma}$ can be calculated to verify if the stress is higher, equal or lower than the maximum allowable stress or not. When the RF factor is higher than, equal to or lower than one it indicates that it is overdesigned, properly designed or underdesigned (meaning structural failure) respectively.

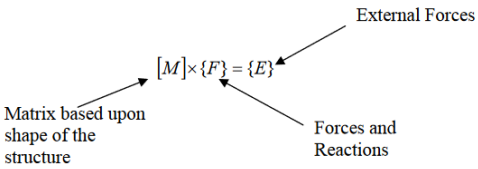
The equilibrium equations are systematically described in matrix form to make sure the equilibrium equations can be easily described by a computer, shown in Figure D.2. Systematic description is performed by enumerating from node to node, where the horizontal coefficient is determined by taking the cosine of ϕ and the vertical coefficient with the sine of ϕ. Plugging the obtained values from the geometry by enumerating node by node would result in an example matrix shown below, where M is the matrix based on the shape of the structure, F is a vector with forces (and reactions) and E stands for the external forces.



(a) Step 1: Systematically describing the geometry by enumerating from node to node.



(b) Step 2: Plugging the obtained values from step 1 results in the above example matrix.



(c) The matrix from step 2 can be summarized by the figure above.

Figure D.2: Systematically describing truss structure with matrices. Figures retrieved from[1].

SOLVING TRUSS STRUCTURE CONTINOUSLY BY MATRIX INVERSION

After systematically describing the truss in matrix form, without taking the unnecessary reaction forces into account, the following matrix is obtained:

$$\begin{bmatrix} 1 & -.243 & 0 \\ -1 & 0 & -.707 \\ 0 & 0 & -.707 \end{bmatrix} \times \begin{bmatrix} F_{int_0} \\ F_{int_1} \\ F_{int_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix} \tag{D.1}$$

Simply inverting the matrix results in:

$$\begin{bmatrix} F_{int_0} \\ F_{int_1} \\ F_{int_2} \end{bmatrix} = \begin{bmatrix} 1 & -.243 & 0 \\ -1 & 0 & -.707 \\ 0 & 0 & -.707 \end{bmatrix}^{-1} \times \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix}$$

Thereafter it is easy to obtain the area's with $\sigma_{max} = 5$ MPa, then A_0, A_1 and A_2 are $4.0 \cdot 10^{-4} m^2$, $1.649 \cdot 10^{-3} m^2$ and $5.657 \cdot 10^{-4} m^2$ respectively.

[1] https://www.unm.edu/~bgreen/ME360/Statics%20-%20Truss%20Problem.pdf at 19 April 2021.