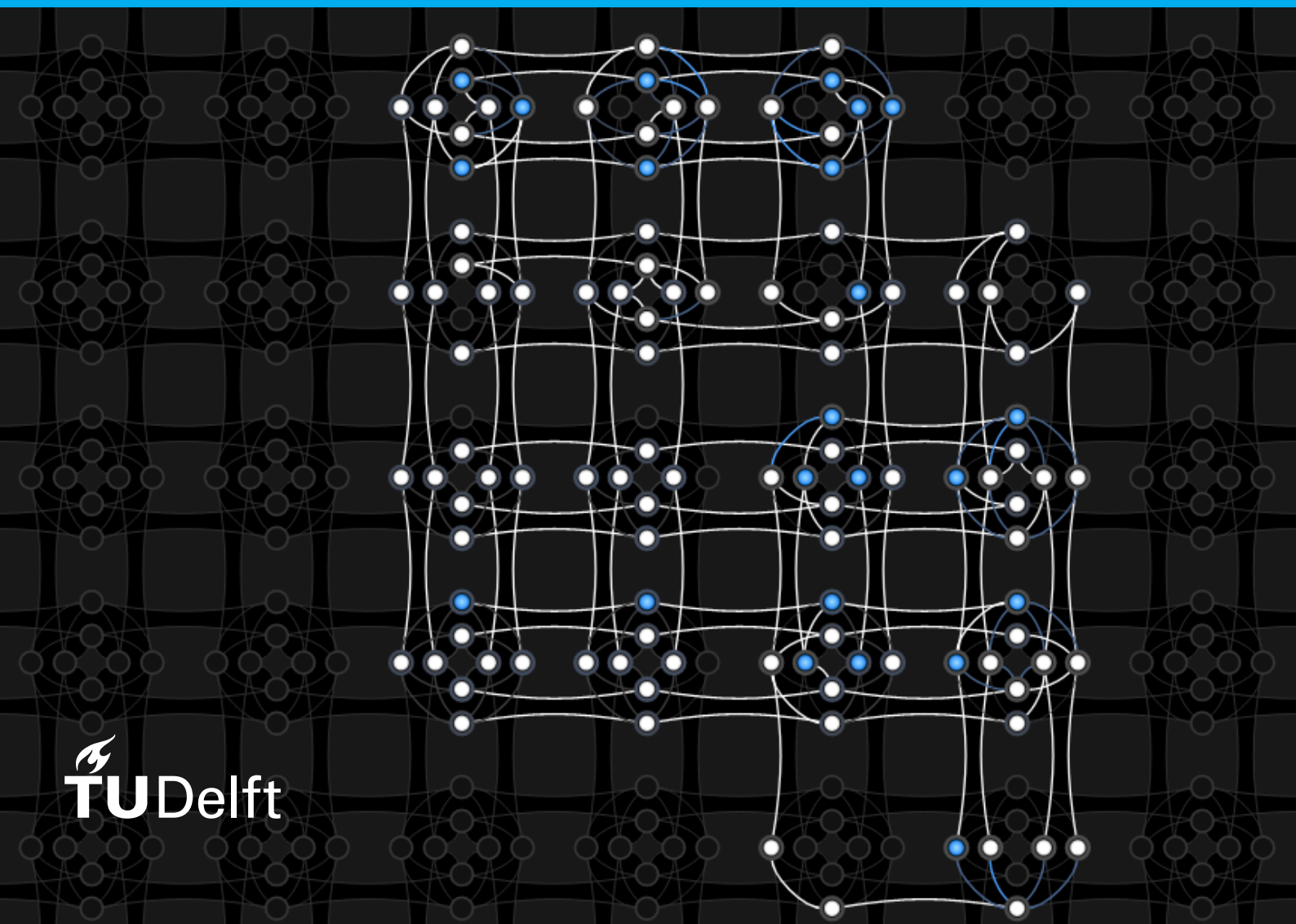
Quantum Computing for Structural Optimization

K. A. Wils

MSc. Thesis



TU Delft

Quantum Computing for Structural Optimization

by

K. A. Wils

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday October 30, 2020 at 13:30 o'clock.

Student number:	4150929	
Project duration:	September 2, 2019 – October 30, 2020	
Thesis committee:	Dr. ir. R. de Breuker	- TU Delft, Aerospace Engineering, chair holder
	Dr. B. Chen	- TU Delft, Aerospace Engineering, supervisor
	Dr. M. Möller	- TU Delft, EEMCS Faculty, examiner

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Cover image: An optimization problem embedded onto the Chimera graph structure of the D-Wave quantum annealer. Figure produced using the D-Wave Inspector tool [25].

TU Delft

Preface

Approximately one year ago, I had the luxury of being able to choose from many different thesis topics. Amongst some of the more traditional *ASCM* topics was one particularly unorthodox option: an exploration into the feasibility of using quantum computing for finite-element applications. The risk-reward tradeoff that I was facing was best worded by Dr. Boyang Chen himself:

> *This work is explorative in nature. It is for the adventurous souls. The field is wide open, whatever you establish could potentially be ground-breaking. The risk is of course also high: it is possible to end up being completely lost. I will be there to support you as far as I can. – Dr. Boyang Chen, 2019*

I decided to take the risk. It was daunting and confusing at first, because I was delving into a topic on which I had essentially no prior knowledge. However, the more I learned, the more I started to become passionate and enthusiastic about the possibilities that laid ahead of me. A great starting point for my research turned out to be the book *Quantum Computing for Everyone*, by Chris Bernhardt. The book was not only an enjoyable read, but did an excellent job of explaining the basic principles behind quantum computing, and even showed some of the math that is involved. For anyone that is intimidated by quantum computing, has no prior knowledge on the topic, but wants to learn more, I highly recommend this book.

Of course, throughout the project, it certainly helped to have an equally passionate and supportive supervisor joining me on this adventure. Important decisions had to be made, discussions and brainstorming sessions were had, and there were even some nasty bugs in the code which had to be fixed. Boyang was there to support me and give advice every step of the way. I want to extend my utmost thanks to him for being so understanding and patient with me while I worked my way through this project.

One of the unexpected challenges faced in this project had nothing to do with quantum computing, QUBO formulations, Python codes, or any of the other fancy technical terms I have come across. For me personally, one of the hardest parts of the project was having to finish the thesis during the Coronavirus epidemic. It certainly took time to get used to the sudden mandated work-from-home situation. Fortunately, my family and friends, my lovely girlfriend, and my brilliant ASM classmates were there to support me when I needed them. I would not have been able to finish my thesis without you. To the scientists who are working tirelessly to develop a vaccine: you have my thanks, and may you swiftly find success.

Looking back to last year, I can honestly say that choosing this thesis topic has been one of the best decisions I have ever made. It was worth it to take the risk. I have found a passion that I didn't know I had, and this project may have even opened doors to career opportunities that I would never have considered. My eyes have opened to a whole new world of possibilities, and I can't wait to see what the future of quantum computing brings. Although parts of the project have been somewhat of a rollercoaster, I wouldn't have done it in any other way. I hope you enjoy reading about my work.

K. A. Wils
Delft, September 2020

Abstract

Quantum computing is a new form of computational technology, which can potentially be used to solve certain problems faster than is possible using classical computers. For this reason, there is an industry drive to develop early quantum computing applications. In this thesis, an overview of quantum computing technologies is provided, along with a practical discussion of the Traveling Salesman Problem, making use of the D-Wave quantum annealer. Subsequently, the main objective of the thesis can be investigated, which is to explore how quantum computing can be used to aid in solving structural optimization problems.

Two methods are developed with which simple 2-dimensional truss systems can be optimized using the D-Wave quantum annealer. The methods aim to find the most lightweight choices for the truss cross-sectional areas while complying with material limit stress constraints. The first method directly casts such an optimization problem into a QUBO format. However, due to difficulties with formulating the stress constraint, this method was found to produce a trivial optimization problem. The second method attempts to symbolically solve a truss finite-element problem, using the resulting symbolic expressions to set up an optimization objective function. Although these objective functions are confirmed to work via classical brute-force analysis, the quantum annealer is shown to have difficulty finding the global optimum solution for truss systems with three or more elements. These results indicate that it is not currently beneficial to use quantum annealing for these structural optimization problems. Nevertheless, some improvements to the method for setting up the objective functions are suggested. The next generation of quantum annealers is expected to perform better for these practical applications, potentially becoming a useful tool in the engineering toolbox.

Contents

Abstract v

List of Abbreviations ix

1 Introduction 1
1.1 Aim and Scope . . . 2
1.2 Thesis Layout . . . 3

2 Literature Study 5
2.1 General Purpose Quantum Computers . . . 5
2.1.1 Quantum Bits . . . 5
2.1.2 Hardware Overview . . . 6
2.1.3 Applications . . . 6
2.2 Quantum Annealers. . . . 8
2.2.1 Overview and Applications. . . . 8
2.2.2 Problem Formulation . . . 9
2.2.3 Hardware Limitations . . . 10
2.3 Selecting a Quantum Computer for Practical Applications . . . 11
2.4 Optimization Problems . . . 13
2.4.1 Optimization Applications . . . 13
2.4.2 Quantum Assisted Genetic Algorithm . . . 15
2.4.3 Finite-Element Shape Optimization . . . 15
2.5 Conclusion . . . 15

3 Introduction to Practical QUBO Problems 17
3.1 Quadratic Unconstrained Binary Optimization . . . 17
3.2 Traveling Salesman Problem . . . 18
3.2.1 Beginnings of a TSP QUBO. . . . 18
3.2.2 Distances . . . 19
3.2.3 Constraints . . . 20
3.2.4 Embedding the TSP . . . 23
3.2.5 Results . . . 24
3.2.6 Final Comments . . . 25

4 Truss Sizing Optimization: Direct QUBO Method 27
4.1 Overview . . . 27
4.2 Direct QUBO Formulation . . . 27
4.2.1 Design Variables and Objective Function . . . 28
4.2.2 Unary Constraint . . . 29
4.2.3 Stress Constraint: Preliminary Information . . . 30
4.2.4 Stress Constraint: RF Dependent Preference in an Optimization Scheme . . . 31
4.3 Testing of Optimization Procedure . . . 33
4.4 Results . . . 34
4.4.1 Box Truss System. . . . 34
4.4.2 Bridge Truss System . . . 38
4.5 Triviality of the QUBO Formulation and Classical Reproduction . . . 41
4.6 Possible Extensions . . . 42

5 Truss Sizing Optimization: Symbolic Finite-Element Method 45
5.1 Phase 1: Preparation . . . 45
5.1.1 QUBO Basics and Plan . . . 45
5.1.2 Sample Problems . . . 46
5.1.3 Challenge: Symbolic Matrix Inversion and Setup of Symbolic Expressions . . . 48
5.2 Phase 2: Setup of the QUBO Problem . . . 49
5.2.1 Objective Function Evaluation Method . . . 50
5.2.2 Expected Optimization Outcomes . . . 50
5.2.3 Challenge: Setting up an Objective Function . . . 51
5.2.4 Challenge: Fractional Objective Functions . . . 59
5.2.5 Practical Implementation of Truss Sizing Optimization . . . 66
5.3 Phase 3: Solving the QUBO Problem . . . 72
5.3.1 Analysis Procedures . . . 72
5.3.2 Parameter Tuning . . . 73
5.3.3 Results: Two-Truss Problem . . . 74
5.3.4 Results: Three-Truss Problem . . . 77
5.3.5 Results: Four-Truss Problem . . . 79
5.4 Final Discussion . . . 81
5.5 Chapter Summary . . . 84

6 Conclusion 87
6.1 General Findings and Conclusions . . . 87
6.2 Answering the Research Questions . . . 88
6.2.1 Research Question 1 . . . 88
6.2.2 Research Question 2 . . . 89
6.2.3 Research Question 3 . . . 90
6.2.4 Main Research Question and Research Objective . . . 91

7 Recommendations and Future Work 93
7.1 Improvements to the Current Methodology . . . 93
7.2 Suggestions for Future Work . . . 94

Bibliography 97

List of Abbreviations

FEM	Finite-Element Method
GPQC	General Purpose Quantum Computer
HHL	Harrow, Hassidim, and Lloyd
LFP	Linear Fractional Programming
LP	Linear Programming
NA	Not Applicable
QA	Quantum Annealer
QPU	Quantum Processing Unit
QUBO	Quadratic Unconstrained Binary Optimization
RF	Reserve Factor
RT	Real Time
ST	Solve Time
TSP	Traveling Salesman Problem

1 Introduction

Imagine you find yourself standing in the middle of a vast landscape. Hills and valleys surround you in all directions, and someone has promised to give you cake if you can find which of the valleys is the deepest. Naturally, you want the cake, but how are you going to find the deepest valley if you don't know where it lies? You could start exploring the landscape, taking measurements of your altitude along the way. Investigating a valley is easy, since you will be walking downhill. However, once you've performed your measurement, you will have to go all the way back uphill again, just so you can make your way to the next valley. Furthermore, it is difficult to know with certainty which valley is the deepest, unless the depth of each and every one is measured. *If only there were tunnels between each of the valleys!*

If there were tunnels you could simply travel downhill between each of the valleys, until you find the one from which all tunnels lead uphill. Sometimes it may be difficult to tell if tunnels are sloping downwards or upwards, but eventually you would be fairly confident to have arrived at the lowest valley in the landscape, allowing you to claim your sweet reward. However, this plan would only work if such tunnels existed, which is unfortunately not the case.

Or is it? A clever person may invent a computer-powered robot that will do the exploration for them, simply waiting for the robot to return with its findings. However, it could take a very long time for the robot to return if it was asked to measure each and every valley, and the cake might go stale. But what if a quantum computer was used to power the robot? By using a quantum computer to guide its way, suddenly, the robot would be able to find the tunnels between the valleys. It turns out, they were there all along but were hidden from plain sight. Because the quantum computer enables the use of the tunnels, the task of finding the lowest valley becomes much easier, and the cake can be enjoyed in a much more timely manner.

Of course, this silly story is a metaphor for the potential that quantum computers offer, with their ability to quickly solve some of the most difficult problems that people have devised. The landscape represents every possible solution to an optimization problem, with the location of the lowest point corresponding to the most optimal solution. It also demonstrates some of the difficulties that classical computers can have when solving optimization problems. Namely, once a valley has been found, it can be difficult to escape, in order to search for potentially even lower points. This corresponds to the problem where classical solution algorithms might get trapped in local minimum solutions, being unable to further explore in search of the global minimum solution. One simple method to guarantee finding the global optimum solution is to test every possible solution to the optimization problem using a brute-force approach. However, such an approach is generally very inefficient, since, every additional variable in the optimization problem causes the total number of possible solutions to increase exponentially. This issue is known as *the curse of dimensionality*, and means that this approach is eventually intractable for large optimization problems. *So how can quantum computers help us?*

Quantum computers are rather unique devices that, by leveraging quantum mechanical principles, enable certain types of problems to be solved much more efficiently than is possible with classical computers. While classical computers use binary bits, 1s and 0s, to perform their computations, quantum computers make use of *quantum bits*. Quantum bits, or *qubits*, can not only represent the classical 0 and 1 states, but can also be in both of these states simultaneously. This quantum mechanical phenomenon is known as *superposition*, and when leveraged effectively, is one of the reasons why quantum computers promise better performance in certain applications.

There are two main types of quantum computers currently in development, being the General Purpose Quantum Computer (GPQC) and the Quantum Annealer (QA). With the GPQC, most of the potential improvements stem from the fact that these systems can run complex quantum algorithms, allowing for much

more efficient problem-solving methods to be devised. On the other hand, a QA can only use the quantum annealing algorithm to solve very specific types of optimization problems, known as quadratic unconstrained binary optimization (QUBO) or Ising model problems. However, as hinted at in the initial story, the QA is able to leverage *quantum tunneling* to aid in finding optimal solutions in the *energy landscapes* of the problems that it can solve. This effect helps the QA in quickly traversing the energy landscape, allowing for low-energy solutions to large optimization problems to be found.

Both quantum computing technologies are still quite novel, and quantum computing hardware is still in its infancy compared to the advanced state of classical computing technologies. Research into practical applications of quantum computing stem only from the last several years, however, some studies have already shown promising results [62, 81, 85, 97]. Furthermore, in industry, some companies are already pushing for the development of early practical applications of quantum computing. For example, Airbus has posted the Airbus Quantum Computing Challenge. One of the challenges is to optimize a wingbox structure, the main load-bearing component in aircraft wings, using a quantum computer [2]. Another example comes from Volkswagen, who have researched how a QA can be used to optimize a traffic flow problem [62]. Volkswagen has already applied this research for the real-time optimization of public transport routes in Lisbon [87].

Considering these early applications, there must certainly be some problems in aerospace engineering for which the application of quantum computing technologies will be beneficial. In the aerospace industry there is a continuous demand to develop the most lightweight structures, as this can lead to increased fuel efficiency, or increased payload capacity, both of which can lead to higher profits. To analyze structures, the finite-element method (FEM) is often used to calculate various structural properties, such as the structural stiffness or strength. Because aircraft must be extensively analyzed to determine if they meet safety standards, this means that finite-element problems are ubiquitous in the aerospace industry. However, such problems can be slow to solve, which is especially inconvenient when structures are iteratively changed and analyzed in an effort to optimize the structural weight. A quantum computer may be able to help solve FEM problems or structural optimization problems more quickly than is possible using the classical methods. Thus, the main idea that prompted this thesis is formed: let us investigate how quantum computing can be used to aid in solving the problems found in aerospace engineering.

1.1. Aim and Scope

This thesis aims to provide a practical look at quantum computing technology and investigate the feasibility of using this technology to aid in solving engineering problems. Of the two types of quantum computers, the main focus for this thesis will be on the quantum annealer. The QA is chosen due to the higher level of technological maturity compared to GPQC, offering significantly more qubits of processing power. Because the QA can only solve specific types of optimization problems, the objective in this thesis will be to investigate how a practical engineering optimization problem can be made compatible with the QA.

The main commercial supplier of QA technology is the company D-Wave Systems Inc. D-Wave produces the D-Wave 2000Q, which represents the state-of-the-art in QA hardware. D-Wave has developed many Python packages to interface with their QA [19]. Furthermore, the company provides various example problems and hosts community forums, giving ample opportunity to learn how their QA system works [24]. Thus, these tools make the D-Wave 2000Q an attractive platform to use for practical investigation into optimization problems from the aerospace engineering industry. More details will be provided in Chapter 2.

Due to the novelty of this particular topic of research, the optimization problems that will be investigated are limited to the optimization of simple 2-dimensional truss structures. Optimization of truss structures usually targets minimal structural weight, while aiming to comply with various displacement, stress, or stability requirements. To evaluate compliance with such requirements linear finite-element analysis techniques will be used. If a truss system structural optimization can be effectively performed using QA technology, this thesis could serve as a foundation for future work related to the more complicated structures found in the aerospace industry.

The goal of this project can thus be formalized with the following research objective:

> *Within the time-span of the thesis, the objective is to investigate the practical application of quantum computing to aerospace structural optimization problems, by developing a method to cast the problem into code that can be interpreted by the quantum computer, and evaluate the performance and reliability compared to classical methods.*

To guide the process of achieving this research objective a number of research questions have been formulated. The main research question will be:

> *What method can be used to solve practical structural optimization problems using a quantum computer, and how well does it perform compared to classical methods?*

This main research question may be supported by answering the following sub-questions:

1. What typical optimization problem can be formulated and solved classically, such that it can act as a reference for the performance of quantum solution algorithms?
   (a) What is a typical problem in aerospace engineering that is suitable for solving on both classical and quantum computers?
   (b) What method is used to classically solve this problem?
   (c) What performance metric can be used to compare the performance of classical and quantum solution algorithms?
2. What method can be used to cast an optimization problem into a formulation that the quantum computer can understand?
   (a) What programming interface exists that allows for problems to be submitted to the quantum computer?
   (b) How can the optimization problem be cast into a QUBO or Ising model formulation?
   (c) What parameters or settings in the quantum computer will influence the time-to-solution, and which settings give the best computational performance?
3. How does quantum computing compare to classical computing for solving practical optimization problems?
   (a) For the reference problem, what is the computational performance of both classical and quantum solution procedures?
   (b) What reliability issues exist with quantum computation, and how may these issues be addressed?
   (c) If the size of the problem is increased, how does this affect computational performance?
   (d) If the quantum computing hardware would be improved in the future, how would this affect computational performance?
   (e) At this point in time, is it beneficial to apply quantum computing to practical aerospace engineering problems?

By finding answers to these research questions, the research goal can be achieved in a structured manner.

1.2. Thesis Layout

This thesis is laid out in the following manner. Directly following this introduction, in Chapter 2, an overview of quantum computing and related literature is provided. This is a somewhat shortened version of a previously completed literature study. The literature study introduces both GPQC and QA technologies, discusses the applications, and provides the background for the choice to limit this project to using the quantum annealer for optimization problems. After the literature study, a deeper dive into practical quantum annealing is taken by exploring the Traveling Salesman Problem (TSP) in Chapter 3. Starting from basic knowledge on QUBO problem formulations, an intuitive method is shown for defining a quantum-compatible version of the TSP. The goal for this chapter is to further introduce quantum annealing to readers who may be new to the topic and provide insight into the underlying mathematical structure of QUBO problems.

Chapter 4 describes the first attempt at formulating a truss sizing optimization problem for use with the QA. Building on previous knowledge, this method attempts to directly formulate the truss optimization problem in a QUBO format. Hence, this method is referred to as the *direct method*. However, it will be seen that the method does not make beneficial use of the quantum annealer. Thus, in Chapter 5, a second attempt to formulate a truss sizing optimization problem is taken, using a completely different approach. In this case, symbolically defined finite-element truss problems are solved, eventually leading to objective functions that

the QA can minimize. This method is therefore referred to as the *symbolic finite-element method*. It will be seen that this method is much more successful than the initial direct method, however, it also comes with significant limitations on its practical usability.

In Chapter 6, the conclusion to the thesis project is offered, summarizing the major findings of the previous chapters. Furthermore, answers to the research questions will be provided. Finally, Chapter 7 discusses the possible improvements to the methods shown in this thesis and gives some recommendations for future research on the topic of quantum annealing for practical engineering optimization purposes.

2
Literature Study

In this chapter, some of the background and scientific literature relevant to quantum computing and optimization problems is discussed. An overview of both general purpose quantum computer (GPQC) and quantum annealer (QA) technologies is given, as well as some of the problems that these technologies might help to solve. From this review, it is decided that, based on the current state of both quantum computing technologies, QA is more likely to be useful to current practical optimization applications. Hence, a review of the applications of QA to optimization problems is also provided.

2.1. General Purpose Quantum Computers

In this section an overview of general purpose quantum computers is given. The foundation for quantum computers, the quantum bit, is first discussed. Then, some information on GPQC hardware, and some of the applications of the technology are highlighted. One of the most relevant algorithms that might be useful for engineering problems is the HHL algorithm, which is discussed more thoroughly. However, since the eventual focus of the thesis will be on quantum annealing, the information in this section will be presented in a rather brief manner.

2.1.1. Quantum Bits

In classical computing, information is always stored and processed in the form of bits. The bit is the smallest piece of information the computer can process, and is always in one of two possible states, being either in the 0 state or the 1 state. The computer can control and manipulate the state of these bits through the use of logical Boolean operators, such as the AND-, OR-, and NOT-gates. For example, the NOT-gate can flip a bit from a 0 state to a 1 state, or vice versa. The AND-gate acts as a comparison between two input bits. If the input bits are both in the 1 state, the AND-gate will output a 1-bit. When the input bits have opposite values, or are both in the 0 state, the AND-gate will output a 0-bit. The OR-gate also performs a comparison between two input bits. It will output a 1-bit when the two input bits have opposite values, or when the input bits are both in the 1 state. Otherwise, if both input bits are in the 0 state, the OR-gate will output a 0-bit. At the most basic level, all computer programs and operations are run by manipulating bits using these logical operations. For more information on this topic, and a comprehensive introduction to quantum computing, the book by Bernhardt can be referenced [9].

General purpose quantum computers (GPQC) attempt to extend computing technologies into the quantum realm by building a computer in which every quantum bit, or *qubit*, is fully controllable in a similar manner to how classical bits are controlled. However, the main difference between classical bits and qubits is that qubits can also have any arbitrary state between 0 and 1. More specifically, the qubits can be in any arbitrary quantum *superposition* of the 0 and 1 states.

Although qubits can be in a quantum superposition of the 0 and 1 states, when an attempt is made to measure the state of the qubit, a classical 0 or 1 state is always found. This is because the act of measuring the qubit state causes the quantum superposition to collapse. Whether either a classical 0 or 1 state is measured depends on what the exact quantum state of the qubit was. Since, the quantum state is defined by the probabilities of measuring each classical outcome state. For example, a qubit might be in a quantum state where it has a 40% chance of yielding a 0 state, and a 60% chance of yielding a 1 state upon performing a measurement. These probabilities of measuring either classical state are what the quantum computer can control and manipulate. However, the fact that qubits can be in a quantum superposition state before being measured is

part of what allows quantum computers to theoretically perform certain tasks faster than is possible using a classical computer.

In a GPQC, the quantum states of the qubits are manipulated using so-called *quantum gates*, hence the GPQC is also referred to as a gate-based or gate-model quantum computer. Quantum programs or algorithms generally consist of sequences of various quantum gates acting on multiple qubits, to achieve some desired output quantum state. Such quantum programs are typically referred to as *quantum circuits*. In principle, GPQC would not only be able to run quantum algorithms, but also be used to perform classical algorithms. This is because qubits are also able to represent the pure classical 0 and 1 states, in addition to being able to express any superposition thereof. Therefore, a GPQC is able to emulate the behavior of a classical computer. In other words, classical computing is simply a subset of quantum computing, where bits are limited to the values of 0 and 1 [9].

2.1.2. Hardware Overview

There are a number of different approaches to creating GPQC hardware. Although the most easily scalable are the superconducting quantum circuits [97], certain publications have also relied on GPQC based on photonic quantum circuitry, and nuclear magnetic resonance devices [8, 13, 67]. When it comes to running a program on the quantum computer, every hardware implementation can perform the same operations. However, it is expected that the superconducting quantum circuit GPQC is the most suitable for practical applications due to its scalability, thus having an increased capability of solving larger problems than the other hardware types.

Notable efforts in the production and operation of GPQC hardware were made by IBM, Intel, Rigetti, Google, and others [72]. Current state-of-the-art GPQC, such as the quantum processor by Google, have in the order of 50 qubits [7]. IBM's recent efforts have led to the launch of the IMB Q System One commercial hardware, which is the first commercial GPQC, having a 20-qubit processor [44]. Furthermore, IBM hosts the IMB Q Experience cloud-based quantum computing service that gives users several options from 5 to 14 qubits of computing power [43]. Rigetti offers a similar quantum computing cloud service which was recently updated to offer 31 qubits of processing capability, using their Aspen-8 QPU platform [73].

In the last year, a collaboration between NASA and Google has created some buzz in the news media [90]. A paper was published that claimed that Google and NASA had achieved a major milestone in the quantum computing industry. Namely, the paper indicated that *quantum supremacy* had been achieved. Quantum supremacy is defined as the point in time where state-of-the-art quantum computers can perform a calculation or simulation that no classical computer could possibly perform in a feasible amount of time. Google's quantum processing unit (QPU), Sycamore, contains 54 qubits. However, in the QPU that was used to demonstrate quantum supremacy, one qubit was defective, leaving 53 functional qubits. On this QPU the researchers were able to sample a quantum circuit one million times in approximately 200 seconds. The equivalent task would take approximately 10,000 years to simulate on a state-of-the-art supercomputer [7]. Since this is an infeasible amount of time, quantum supremacy has supposedly been achieved. In response, IBM has claimed that this simulation can be performed in only 2.5 days, rather than 10,000 years, and that therefore quantum supremacy has not yet been achieved [69].

As an alternative to GPQC hardware, there are also companies that offer quantum computing simulation capabilities. For example, the Quantum Inspire platform, provided by QuTech, allows users to build quantum circuits using up to 31 qubits. The quantum circuits are simulated on a classical computer system in order to produce the results of the 'quantum computation' [71]. The drawback to GPQC simulation is that it quickly becomes infeasible to perform simulations as the number of qubits increases. Since every qubit may end up in one of two possible states, and every additional qubit doubles the number of possible outcomes, the computational requirement increases exponentially with every additional qubit. In the absence of physical GPQC hardware, quantum computing simulation tools can be usefully applied for the theoretical development and testing of new quantum algorithms. However, the capabilities are limited due to the small number of simulated qubits.

2.1.3. Applications

There are some problems that are particularly well suited for general purpose quantum computing. There are three particularly important algorithms that have been proposed for solving problems on a GPQC in a faster manner than is classically possible. Namely, Shor's algorithm, Grover's algorithm, and the HHL algorithm.

Perhaps the most infamous of the quantum algorithms is Shor's algorithm, which has the potential of overthrowing the RSA encryption method that is currently ubiquitously used for internet security applica-

tions. Given a large number N, Shor's algorithm provides a method to find the factors p and q such that $pq = N$ [18]. Classically, this factoring problem becomes exponentially more difficult as the size of N increases. The fact that this is classically a difficult task forms the basis of the RSA encryption method [18, 96]. However, using Shor's algorithm the problem may be solved in time that scales roughly cubically with the size of the problem [18]. As such, for sufficiently large numbers, Shor's algorithm becomes much more efficient than classical methods. The recent work by Amico et al. demonstrates Shor's algorithm on the IBM Q Experience GPQC, for factoring the numbers 15, 21, and 35, using at most 7 qubits [6].

One task people commonly perform with computers is searching for information within a database. Consider for example the internet search engine that Google provides. Grover's algorithm may be useful for such purposes. The algorithm allows for a specific data entry within a randomly ordered database to be found more efficiently than is classically possible. Given a database of size N, it would classically take up to N attempts to find the desired data entry in the database. However, Grover's algorithm will allow for the correct entry to be found, with a probability greater than 50%, in only $\sqrt{N}$ steps [18]. An illustrative example of the application of this algorithm would be the following. Imagine a deck of playing cards lying face down on a table. If one wishes to find a specific card, such as the ace of spades, the classical approach would be to sequentially turn over cards at random until the target has been found. Given N cards on the table, in the worst possible case, it would classically take N attempts to find the desired card. Grover's algorithm has a greater than 50% chance of finding the correct card in $\sqrt{N}$ steps.

The HHL algorithm is related to solving linear systems of equations, which is relevant to many engineering problems, including structural finite-element problems [18, 39, 61]. A linear system of equations, in the context of finite-element problems, is typically defined as shown in Eq. (2.1). In this case, the matrix $[\mathbf{K}]$ is known, and represents the global stiffness matrix. This matrix defines the stiffness behavior of the finite-element structure. Furthermore, the vector $\mathbf{f}$ is also usually known, as this vector defines the forces which are applied to the structure. The goal for the finite-element problem is to find the solution vector $\mathbf{u}$, which contains the displacement of every node in the structure. By knowing the displacements of all nodes in the structure, other metrics such as the element stress and strain can also be calculated. This allows engineers to determine if the structure is compliant with requirements on the structural stiffness and strength, and can help identify structural weaknesses.

$$[\mathbf{K}]\,\mathbf{u} = \mathbf{f} \tag{2.1}$$

The first method that was proposed for using quantum computers to aid in solving linear systems of equations was by Harrow, Hassidim and Lloyd, in 2009. This method is now known as the HHL algorithm. If the number of equations in the linear system is defined as N, the HHL algorithm aims to solve certain types of linear systems in an amount of time that scales only logarithmically in N. This is an exponential improvement over the best known classical algorithm [39]. The HHL algorithm would rely on a GPQC, using quantum gates to manipulate qubits.

The HHL algorithm comes with a number of caveats, which were clearly explained by Aaronson in 2015 [1]. Since these issues with the algorithm are particularly important, they will be briefly explained here, in the context of the finite-element equation that was shown in Eq. (2.1).

1. The HHL algorithm, with its exponential speedup relative to the classical algorithm, assumes that the vector $\mathbf{f}$ can efficiently be prepared as a quantum state.

2. The matrix $[\mathbf{K}]$ must be a sparse matrix, containing mostly zero entries. If the matrix is not sparse enough, the exponential speedup is lost.

3. The matrix $[\mathbf{K}]$ must be well-conditioned, meaning that the difference between the largest and the smallest eigenvalues of the matrix should be minimal.

4. The HHL algorithm does not find the full solution vector $\mathbf{u}$. Instead, a quantum state solution vector is prepared which contains entries proportional to the entries of the solution vector $\mathbf{u}$. The user can only extract one scalar quantity of interest from this quantum state. The user cannot find the full solution vector $\mathbf{u}$ without destroying the promised exponential speedup.

Particularly the last point is critical to understand. The HHL algorithm is *not* able to find the full solution vector to the linear system of equations. This does not necessarily mean that the algorithm is useless in the context of finite-element problems. However, it does imply that for solving a finite-element problem on a general purpose quantum computer, a change in perspective and solution strategy is necessary.

In years subsequent to the publication of the HHL algorithm, there have been numerous attempts at practical implementations of the algorithm, as well as finding further speed improvements. Notable runtime improvements to the HHL algorithm were found by Ambainis in 2010, and in 2015 by Childs et al. [5, 15]. An extension of the HHL algorithm is shown in [16], which allows for the linear system to be preconditioned, making the system easier to solve. An adaptation by Wossnig et al. allows for dense linear systems to be solved [94]. Another adaptation of the HHL algorithm is given in [54], showing a hybrid computation method that relies on both quantum and classical computer systems in order to solve the system of equations. An extensive overview and explanation of the HHL algorithm and the improved versions can be found in [31]. With specific regard to finite-element problems, a review given in [95] indicates that two of the caveats for the HHL algorithm, matrix sparsity and conditioning, are often not an issue. A general overview of many different quantum algorithms, including the HHL algorithm, is provided in [18].

Concerning practical implementations, the first attempt at designing a quantum circuit that would run the HHL algorithm was by Cao et al. in 2012 [14]. This circuit design was recently investigated, adapted, and implemented on GPQC simulation software in [84]. Earlier work on the implementation of the HHL algorithm and experimental results are shown in [8, 13, 67]. The first work to consider the implementation of the HHL algorithm on a superconducting quantum circuit is shown in [97]. In this implementation, Zheng et al. solve a 2×2 linear system of equations using four qubits. The improved HHL algorithm that was proposed in [15] was reviewed in the context of finite-element problems by Montanaro and Pallister [61]. It was found this algorithm can indeed lead to a speedup for solving FEM problems, but the speedup is polynomial, not exponential.

2.2. Quantum Annealers

In this section quantum annealers will be discussed. These are a fundamentally different type of quantum computer, and have different capabilities compared to GPQC.

2.2.1. Overview and Applications

An alternative to the general purpose quantum computer is the quantum annealer type of quantum computer. Quantum annealers are special purpose quantum computers that are particularly well suited to solving a specific type of optimization problem [57]. They are less suited to general purpose calculations, since qubits are not fully controllable in the same way as with GPQC.

By far the most prevalent hardware implementation of QA type quantum computers is provided by the company D-Wave Systems Inc. Currently, D-Wave commercially offers the D-Wave 2000Q QA system. This system has 2048 qubits, which is a very large increase from the roughly 50 qubits that the state-of-the-art GPQC offers [24, 58]. Furthermore, D-Wave provides access to this quantum computer through a cloud service, meaning that the state-of-the-art in QA technology can be used by the general public [24].

The QA has been a technology of interest for both Google and NASA since 2012, when NASA started hosting a collaborative effort to investigate potential applications of quantum annealing. Compared to GPQC, quantum annealing is a more mature technology, allowing for higher numbers of qubits to be utilized for computations [10]. Furthermore, for specific types of problems, there is some evidence to suggest that a quantum speedup is possible compared to classical methods [48, 75, 80]. However, as of yet, the quantum speedup for practical problems that may be offered by the QA remains somewhat of an open question.

The range of applications for quantum annealers is quite diverse. Due to the manner in which they solve problems, they are good at finding optimal or near-optimal solutions to problems involving many possible combinations of parameters. One example of a problem well-suited for quantum annealing is the Traveling Salesman Problem, with the aim of minimizing the distance traveled between a set of destinations [21]. The Traveling Salesman Problem will be discussed in more detail in Chapter 3. The work by Lucas shows many of the typical academic problems that can be solved using the QA [55].

Another problem that is often referred to in the context of quantum computers is the prime factorization problem, in which a method is sought to find the prime factors of a large number. In other words, given a number N, the task is to find the prime numbers p and q, such that $pq = N$. This is a difficult problem to solve classically. For this problem one would typically consider Shor's algorithm, but this algorithm was developed for use with GPQC. A method that achieves the same goal has been developed by Jiang et al. for use with the QA [46]. Using their method, Jiang et al. were able to factor the number 249919 into its prime factors 509 and 491. This process used 1803 qubits of the 2048 qubits available in the D-Wave 2000Q quantum annealing processor.

An interesting application of quantum annealing was shown by Van Vreumingen et al. [85]. The research showed that a QA could be used for design optimization purposes. Given a 3-dimensional sphere, a method was developed that allowed the QA to alter the shape of the sphere such that it reflects a minimal number of rays towards a certain plane. The sphere was defined using finite-elements, the use of which is common practice in many fields of engineering. Since the design optimization method showed promising results, and the authors are aiming for further improvements, it is clear that the QA has the potential to be a valuable tool for engineers in the future.

2.2.2. Problem Formulation

Problems that can be solved by a QA need to be formulated in either an Ising model form, or as a Quadratic Unconstrained Binary Optimization (QUBO) problem. These two mathematical formulations are very similar. The Ising or QUBO problem formulation represents a total system energy, and the optimal solution to the problem will be the configuration of classical qubit states that yields the lowest total system energy. In the Ising form, the system energy is given by a Hamiltonian function, as shown in Eq. (2.2) [10–12, 21, 47, 55, 57, 58, 78].

$$H_p(\mathbf{s}) = \sum_{i=1}^{N} h_i s_i + \sum_{i<j} J_{ij} s_i s_j \tag{2.2}$$

In this equation, $\mathbf{s} = [s_1, s_2, \ldots, s_N]$ are Ising spins, which can take values of -1 or 1, i.e. $s_i \in \{-1, 1\}$. The role of the Ising spins is fulfilled by the individual qubits in the QPU. The parameters h_i and J_{ij} are qubit biases (self-interaction) and coupling strengths (qubit-qubit interaction) respectively. The summation as defined in Eq. (2.2) then yields the Ising Hamiltonian H_p [12, 21]. The Ising Hamiltonian is also referred to as the problem Hamiltonian, hence, the subscript p is used. By cleverly manipulating the qubit biases and coupling strengths, problems can be encoded onto the QPU, which will then find the optimal configuration of spins in $\mathbf{s}$ for which the problem Hamiltonian function is minimized.

The QUBO form of the problem definition is incredibly similar to the Ising form given in Eq. (2.2). The main difference between the two formulations is that in the Ising form the qubit states are representative of the values -1 and 1, while in the QUBO form qubit states are representative of the traditional binary states 0 and 1. If the problem is formulated in any of the two forms, it can freely be converted to the other. To this end, an Ising formulation can be converted to a QUBO formulation by substituting the relation from Eq. (2.3) into Eq. (2.2) [10, 12, 21].

$$s_i = 2q_i - 1 \tag{2.3}$$

To solve problems, the QPU is usually first initialized in a state of equal superposition, which defines the initial Hamiltonian, H_i. Then, over a period of time, the problem Hamiltonian is gradually introduced, while the influence of the initial Hamiltonian is gradually faded out. This is shown in Eq. (2.4) [4, 10, 55].

$$H(s) = (1-s) H_i + s H_p \tag{2.4}$$

for which typically:

$$s(t) = \frac{t}{T} \tag{2.5}$$

with t being current time, and T being a predefined total annealing time [10, 21, 55]. The value of the dimensionless time, s, will vary from 0 at $t = 0$, to a value of 1 at $t = T$. Therefore, this allows a smooth Hamiltonian evolution to take place.

Following the adiabatic theorem, if the evolution from initial Hamiltonian to problem Hamiltonian is performed slowly enough, the system will tend to remain in the lowest energy ground state, thus finding the optimal solution to the problem [4, 10, 55]. The practical implementation of this procedure, when transitioning from an initial Hamiltonian to a problem Hamiltonian, is called *quantum annealing*. Hence, the term quantum annealer is used for this type of quantum computer. In the idealized scenario, in a perfectly isolated system, the behavior is exactly described by the adiabatic theorem, thus the term adiabatic quantum computing is also commonly used [52, 78]. In practice, since the QA operates in a finite-temperature environment, around 15 mK [24], the behavior is not perfectly adiabatic.

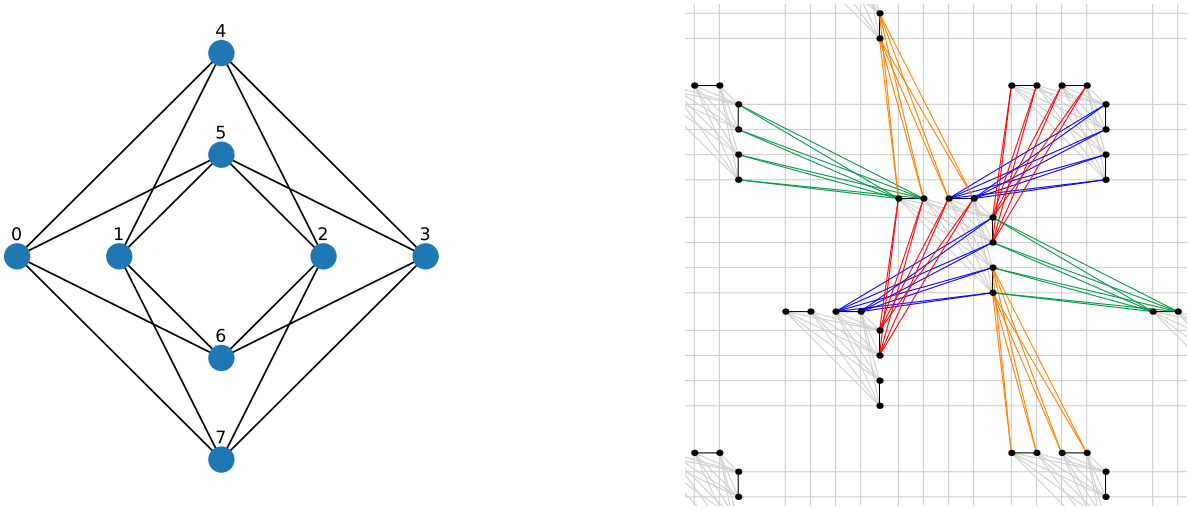
While the Hamiltonian evolution takes place, qubits gradually fall into their final classical states, whilst benefiting from the *quantum tunneling* phenomenon [52, 78]. The Hamiltonian function can be thought

of as representing an *energy landscape*. The optimal solution will be found at the lowest point in this energy landscape. During the evolution the qubit states are traversing this energy landscape to find the lowest point. To arrive at a low energy solution, the qubit states may need to overcome a local peak in the energy landscape. In classical annealing these energy barriers are overcome using additional energy in the form of thermal fluctuations. However, for quantum annealing, through the process of quantum tunneling, qubits can tunnel through peaks in the energy landscape, in order to fall into lower energy states [12, 52]. This prevents the QA from getting stuck in a local minimum of the problem Hamiltonian, but instead allows for the global minimum of the function to be found.

Practically, there are large differences between the GPQC and the QA in the way that problems are solved. For the GPQC, quantum circuits are built using quantum gates, which are then applied to individual qubits. For the QA, problems need to be formulated as Ising model or QUBO problems. However, in [59] a proof is shown which indicates that adiabatic quantum computing and gate-based quantum computing are fundamentally equivalent. In this case however, adiabatic quantum computing refers to the idealized version of quantum computing that strictly follows the adiabatic theorem. In general this is not the case for quantum annealers, which behave similarly, but do not strictly follow the adiabatic theorem [57]. Nevertheless, some of the common GPQC gates are translated for use with the D-Wave QA in [89]. As such, in the future it may be possible to convert GPQC algorithms to work with quantum annealers, and vice versa.

2.2.3. Hardware Limitations

Though the number of qubits in the D-Wave 2000Q QA is relatively large, not every qubit is connected to every other qubit. The hardware has a layout that can be represented using a Chimera graph [65]. Qubits are arranged in bipartite unit cells of eight qubits, containing two subsets of four qubits. Every qubit in a subset of four is connected to every other qubit in the opposite subset. Furthermore, unit cells of eight qubits are connected to neighboring unit cells, to a degree where individual qubits will have at most six connections. In total, the D-Wave 2000Q QPU contains an arrangement of 16×16 unit cells of eight qubits, thus having 2048 qubits in total [21]. In Fig. 2.1a a single unit cell of the Chimera graph is shown. In Fig. 2.2 the full arrangement of qubits in the D-Wave 2000Q QPU is shown.



(a) Single Chimera unit cell consisting of 8 qubits.

(b) Pegasus graph [83].

Figure 2.1: Quantum annealer hardware architectures.

The way in which the physical qubits are connected represents a disadvantage. For certain problems, such as in the work demonstrated in [74], full qubit connectivity is required. This means that every qubit must have a connection with every other qubit, in order to define the problem properly. Mathematically this would mean that the qubit coupling strengths J_{ij} from Eq. (2.2) would be non-zero for every possible combination of i and j. The Chimera graph, with its sparse connectivity between qubits, naturally forces many of the coupling strengths to be zero.

The process of mapping a problem onto the physical hardware of the QA is called *embedding*. Since the Chimera graph provides limited connectivity, certain problems are more difficult to embed onto the physical hardware [65]. This problem can partially be overcome by chaining together multiple *physical qubits* to form

logical qubits with higher degrees of connectivity [33, 35]. This approach is rather inefficient, and quickly reduces the size of the problem that can be solved. The problem could be alleviated altogether by implementing a hardware architecture with all-to-all connectivity, as has been proposed in [53]. Having full qubit connectivity can also speed up the solution time by several orders of magnitude for certain problems [38]. At the time of writing, rather than full connectivity hardware, the next generation of quantum processors being developed by D-Wave promise increased qubit connectivity with up to 15 possible connections per qubit. This hardware layout is known as a Pegasus graph [30, 58], part of which is shown in Fig. 2.1b [83].

Another problem, as with GPQC, is that not every QPU is perfect, and some qubits may not be functional. As is evident through D-Wave's online quantum computing service Leap, the two QPUs available for public use at the time of writing have 2030 and 2038 functional qubits respectively [24]. The malfunctioning qubits can inhibit the embedding of particularly large problems, that could otherwise be embedded onto a 100% functional QPU [35]. However, to circumvent this issue, methods are available for dividing large problems into smaller sub-problems which are then solvable on the QPU hardware [66, 76].

In a recent review by Coffrin et al. [17], the challenges that current quantum annealers have, such as the limited qubit connectivity, are further elaborated upon. The authors suggest the idea that quantum annealers may become useful as co-processors for classical computing systems, in a similar manner to how many modern computers now have dedicated graphics co-processors. Furthermore, the methods used to benchmark the performance of quantum annealers are reviewed. Coffrin et al. expect that, with improvements, the technology could become valuable for hybrid quantum-classical application to optimization problems.

2.3. Selecting a Quantum Computer for Practical Applications

So far, an overview of the two major quantum computing technologies has been given, being general purpose quantum computers and quantum annealers respectively. For GPQC, topics such as qubits and qubit manipulation were briefly discussed, as well as some of the basic applications for which the GPQC might be useful. Specifically, the HHL algorithm was discussed in some depth, as this algorithm promises a much faster method of solving linear systems of equations. This makes the algorithm extremely relevant to the typical finite-element problems that are solved in engineering applications. However, it was also seen that the HHL algorithm has a number of caveats that limit its direct usability. Practical implementations of the HHL algorithm, such as the work by Zheng et al., have been successful at solving very small linear systems, relying on a small number of qubits. Overall, it has been found that state-of-the-art GPQC have approximately 50 qubits of computing power, but that easily accessible options from IBM and Rigetti offer in the range of 5 to 31 qubits [43, 73].

Compared to GPQC, the QA appears to be much more suitable for solving practical optimization problems. Mostly, this is because the technology has a far greater number of qubits available, with the current D-Wave 2000Q system offering up to 2048 qubits. An interesting application of the QA was shown by Van Vreumingen et al., showing that the technology can be used to perform a shape optimization of a finite-element sphere. This indicates that the technology may also be useful for other typical engineering optimization problems. The drawback is that such optimization problems must be formulated using an Ising or QUBO problem framework, meaning that practical problems might need to be reformulated to be compatible with the QA.

Overall, due to the more advanced state of the QA hardware, as well as the easy availability through an online cloud service, it is from this point forward clear that the QA offers the most promising path towards the practical utilization of quantum computing technology. Since the technology is specifically well suited to solving optimization problems, this will be the focus of the next sections of the literature review. An overview of optimization problems and applications of quantum annealing will be provided.

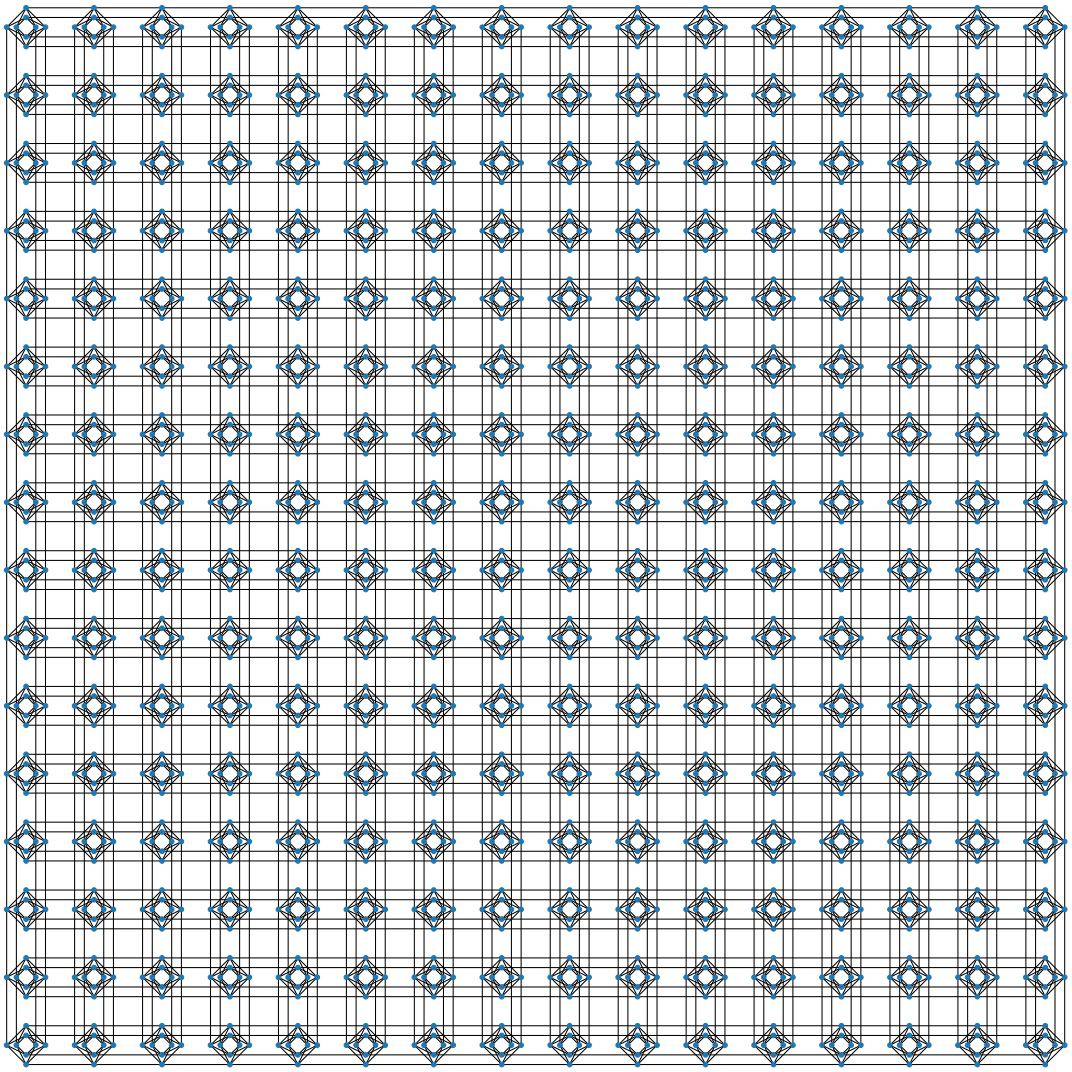


Figure 2.2: Full D-Wave 2000Q Chimera graph.

2.4. Optimization Problems

In aerospace engineering, a common problem is to find optimal designs for airframes, or other load-bearing structures. Usually the target for the optimization is that the weight of the structure should be minimized. Typical constraints on this type of optimization are the maximum stress in the structure, which gives a measure for how close the material is to failure under a given load, and the maximum allowed deflection, which gives a measure for the stiffness of the structure. By optimizing the weight, costs can be saved due to increased fuel-efficiency, or larger profits can be earned due to increased payload capacity. The Airbus Quantum Computing Challenge poses exactly this problem, with respect to the optimization of a wingbox structure [2].

There can be many different variables that play a role in optimization problems. For structural parts, some examples of these variables would be: the material used, the physical shape or design of the part, or the thickness of specific regions. An optimal structural design will aim to make the optimal choice in all of these aspects to achieve the lowest possible structural weight. When the most optimal set of values must be chosen from a set of predefined candidate values, the problem is called a *combinatorial* optimization problem. A recent book on the many different classical methods for optimization problems, including combinatorial optimization problems, was written by Kochenderfer and Wheeler [51].

The typical issue with optimization problems is that, as the number of variables increases, the number of possible solutions, i.e. the size of the solution space, drastically increases. At a certain point, the solution space can get so large that it becomes infeasible to search for an optimal solution within a reasonable amount of time. This is known as *the curse of dimensionality* [60]. Although there are many classical optimization algorithms with different benefits, drawbacks, and computational requirements [51], there are indications that quantum computing may be able to help solve optimization problems more quickly than is classically possible [60]. Since the most promising candidate for solving practical optimization problems is the D-Wave 2000Q QA system, this will be the focus from this point forward.

Quantum annealers are only suitable for solving QUBO or Ising model problems. However, in essence, these problems can be interpreted as general optimization problems, since the QA attempts to find an optimal solution. Usually, the key point is to find a way to cast a practical optimization problem into the correct mathematical form, so that it can be solved by the QA. The method to do so tends to be unique for every different type of problem or application. Thus, in the next section, more applications of the QA that were found in literature will be discussed. A novel optimization algorithm was found in literature [49], and is also discussed. In the context of aerospace engineering and finite-element problems, the most relevant procedure, as given in [85], is described in more detail.

2.4.1. Optimization Applications

A relatively early implementation and experimental test of combinatorial optimization using a QA is given by Djidjev et al. [32]. Specifically, they investigate the maximum clique problem and the graph partitioning problem. The former problem aims to find the maximum group of all-to-all connected vertices in a given graph of vertices and edges. The latter problem aims to divide a graph into two subsets containing equal numbers of vertices while minimizing the number of edges that are cut through. The paper elaborates on some of the programming and solving features that are offered by the D-Wave QA. They conclude that, at the time, a quantum advantage is only seen on problems specifically tailored to the capabilities of the QA. For general problems the QA was not competitive with classical methods.

Quantum annealing has also been applied to financial asset exchange, typically in the context of cryptocurrencies. The asset exchange problem has been analyzed in [37]. An example of a simple asset exchange problem is the following. Given three investors, Investor 1 holds Bitcoin and wishes to trade for exactly 100 Ethereum, while Investor 2 and Investor 3 both hold 60 Ethereum and wish to trade for Bitcoin. A straightforward solution would be to have Investor 2 and 3 both trade 50 Ethereum, but there are many different possible ways to satisfy the problem. The aim of the asset exchange problem is to find the most profitable trade that still satisfies every investor's wishes [37].

Another finance related optimization application of quantum annealing is described by Venturelli and Kondratyev [86]. In this case, a method is described for portfolio optimization, which entails making the most optimal investment choices given a number of assets, their expected returns, and the covariance between assets, such that a desired target return can be achieved. Interestingly, aside from describing a *forward annealing* approach, which is the standard method by which the QA solves problems, Venturelli and Kondratyev also describe a *reverse annealing* approach. Reverse quantum annealing is a method by which, in simple terms, the QA is initialized in a classical candidate solution, after which the reverse annealing step brings some of the qubits back into a quantum superposition state. From this partial quantum state, the

QA proceeds with the standard forward annealing step to find a new solution to the problem. This procedure should allow for the QPU to find solutions that are more optimal than the initially assumed solution, although the technique is extremely novel and has not been the subject of much research. A study by Ohkuwa et al. aimed to establish an analytical framework for the reverse annealing process, such that its performance could be studied [64]. The reverse annealing approach may prove to be a valuable process in iteratively solving optimization problems.

Aside from financial applications, Lucas presents an overview of more traditional optimization problems [56]. Lucas is also known for his earlier work on the Ising formulations of many different optimization problems [55]. However, in the more recent work, Lucas provides more detail on QUBO derivations for the so-called knapsack and number partitioning problems, and further discusses combinatorial optimization problems, such as the graph coloring problem. One issue that is alluded to, however, is that embeddings which require long chains of physical qubits to represent logical variables tend to perform poorly [56]. This issue is also discussed in [70], in which Perdomo-Ortiz et al. discuss the readiness of QA technology for real-world practical industrial applications. The authors indicate that increasing connectivity between qubits, compared to the current Chimera hardware architecture, can have a large impact on the practical applications of QA technology. However, the main point of the study by Perdomo-Ortiz et al. is to function as a baseline for future studies, by providing insight into how QA performance benchmarking should be done. The authors go into both the physics of quantum annealing, as well as the potential optimization applications of the technology. It is concluded that hybrid classical-quantum strategies will likely yield the most useful results for optimization problems. However, there is a need for higher qubit connectivity and higher-order qubit interactions (cubic or even quartic) to solve more complex practical problems.

Two more practical optimization problems which were studied using quantum annealing are related to air traffic management [81] and to traffic flow optimization [62]. In the first case, Stollenwerk et al. [81] use the QA to resolve conflicting aircraft flight-paths. A method is proposed by which this problem is mapped into a QUBO problem, which can be solved on the QA. A simplified version of the problem, where conflicts can only be resolved by applying a time delay at the start of the flights, is solved on the D-Wave QA [81].

The traffic flow optimization problem that was studied by Neukart et al. aims to minimize traffic congestion [62]. Given a number of cars, each of which having a specific origin and destination point, can take one of three proposed routes towards their destination. The optimal solution that minimizes congestion is then the selection of routes for every car such that the number of cars using the same route segments is minimized. The authors used a dataset consisting of 418 cars, traveling to or from the Beijing city center and the airport [62]. This results in a problem that contains too many variables for the QA to directly solve. Therefore, the authors implement a method of dividing the problem into solvable subproblems, and iterate the solving procedure until no further improvement of the general solution is found. The results of the study are promising, showing a clear improvement of the traffic flow. The constraint that was used to enforce that every car can take only one route may be adapted to other contexts, as evidenced in [85], and could be an extremely useful technique for further studies. This constraint is known as the unary constraint and is also briefly discussed by Lucas [56].

An alternative approach to formulating constraints was given by Hen and Spedalieri [42]. The approach in [62] uses a penalty function to enforce a constraint, and is relatively simple to introduce in the QUBO problem formulation. However, this penalty function approach, according to Hen and Spedalieri, comes with numerous disadvantages. Namely, the penalty function necessitates all-to-all qubit connectivity, which in turn makes embedding the problem onto the QPU hardware more difficult. The approach by Hen and Spedalieri aims to construct a so-called driving Hamiltonian, which takes the constraint into account in a formulation that is more naturally usable with the QA, and eliminates the need for penalty functions. This approach allows for more efficient use to be made of the limited hardware resources of the QA. In [41], Hen and Sarandy set up guidelines on how to construct driver Hamiltonian functions.

Another recent study by Ajagekar et al. [3] investigates four different optimization problems. The most difficult problem that was studied was a vehicle routing problem. The problem can be described as having a fleet of vehicles, which depart from a depot, must pass by specific locations to pick up passengers, and finally return to the depot. In this case, the novelty of the study is that the target for the optimization is a ratio of two functions, rather than a single objective function. This ratio is defined as the traveling cost divided by the working time, which is to be optimized for a fleet of vehicles, leading to the most cost-effective set of routes. Using classical methods, large-scale vehicle routing problems could not be solved within 24 hours. The proposed method that uses the QA could solve even the largest-scale problem in less than 10 hours. The results therefore indicate a clear quantum advantage for large-scale problems.

2.4.2. Quantum Assisted Genetic Algorithm

A novel hybrid quantum-classical optimization algorithm is described by King et al. [49]. In the study, a quantum-assisted genetic algorithm is proposed, which additionally makes use of reverse annealing. Genetic algorithms can be used to solve optimization problems iteratively. In general, an initial population of random candidate solutions is generated, after which every candidate is randomly recombined, mutated, and selected according to 'survival of the fittest' principles. In this case, the mutation operation of the genetic algorithm is being performed through reverse annealing. The algorithm was tested using randomly generated Ising model problems, rather than attempting to solve real practical problems. Nevertheless, the results were promising and indicated that the reverse annealing procedure is a viable choice for the mutation step in genetic algorithms. Furthermore, in most cases the theoretical time-to-solution was better than for standard quantum annealing and a number of classical solution methods. However, for this algorithm there is a disparity between the theoretical time-to-solution and the actual wall-clock time needed to solve the problem, due to QPU access overhead, network latency, and other reasons. These issues may partially be solved as QA hardware become more easily accessible in the future [49].

2.4.3. Finite-Element Shape Optimization

The most relevant application of quantum annealing to an optimization problem, in the context of aerospace engineering, is found in the work by Van Vreumingen et al. [85]. It was already briefly reviewed in a previous section, but will now be discussed in more depth. The authors show a method for optimizing the shape of a 3-dimensional finite-element sphere, such that the number of rays, emitted from a point-source and reflected towards a certain plane, is minimized. To achieve this goal, the approach is to iteratively consider different candidate positions of the vertices that define the sphere, and allow the QA to find the optimal arrangement.

The algorithm proposed by Van Vreumingen et al. proceeds as follows. A partial loss function is defined that will calculate, for a single simplex, the fraction of rays reflected towards the plane that must be avoided. This partial loss function requires the three vertices of the simplex to have specific assumed mutations, which are randomly generated. The total loss function for the sphere is then calculated by summing the partial loss functions for every simplex in the discretization, which is the target for minimization. A constraint is added to the total loss function to ensure that every vertex can only exhibit one mutation. As such, the total loss function is defined as:

$$\tilde{\mathcal{L}}\left(S,\mathbf{x}\right) = \mathcal{L}\left(S, c\left(\mathbf{x}\right)\right) + \lambda \sum_{i} \left(\sum_{j=1}^{K} x_{ij} - 1 \right)^2 \tag{2.6}$$

with $\tilde{\mathcal{L}}\left(S,\mathbf{x}\right)$ being the total loss function, which is to be minimized. Furthermore, $\mathcal{L}\left(S, c\left(\mathbf{x}\right)\right)$ is the sum of the partial loss functions of every simplex $s \in S$, for the configuration of mutations $c\left(\mathbf{x}\right)$ with the binary solution vector $\mathbf{x}$. The latter term in the total loss function defines the unary constraint, enforcing that every vertex can have only one mutation. Thus, λ is a penalty coefficient, acting on the binary entries x_{ij} of the solution vector, for every simplex i, and for every assumed mutation $j \in \{1 \dots K\}$.

To run the optimization routine, the initial discretized sphere is generated. Then starting the iteration loop, K possible random mutations per vertex are assumed, with the radius of the mutation decreasing for every iteration of the algorithm so that the solution eventually converges. The total loss function is calculated and is used to generate the appropriate QUBO matrix. The QA uses the QUBO matrix to find the solution bitstring $\mathbf{x}$, which indicates which mutation, if any, gives the most optimal shape of the discretized sphere. This procedure is iterated until the desired result is achieved [85]. Given that a suitable loss function can be formulated, this procedure can potentially be adapted to other finite-element related optimization problems.

2.5. Conclusion

In this literature study, an overview of GPQC and QA technologies was given. Based on an initial investigation into the current state of these technologies and the types of problems these might help to solve, it was decided that the most promising path towards practical applications of quantum computing technology would be through the quantum annealer. This is because the QA offers vastly more qubits, having roughly 2000 qubits as opposed to the roughly 50 qubits present in state-of-the-art GPQC. Aside from the increased number of qubits, the state-of-the-art QA is also conveniently accessible through online means, while for GPQC this is not the case. Since the QA offers more qubits, and is suited to solving optimization problems, this topic has been further researched.

The QA can only solve optimization problems that are defined using a QUBO or Ising model formulation. When presented with such a problem, the QA will attempt to find the solution state for which the Hamiltonian energy is minimized. To prevent the proposed solution from getting stuck in local minimum solutions, the QA relies on quantum tunneling to aid in finding the global minimum solution. There is some evidence to suggest the QA could achieve a quantum speedup compared to classical methods [48, 75, 80], however, for practical problems such a speedup remains an open question.

Nevertheless, numerous studies have used the QA to solve practical problems. In particular, the work by Van Vreumingen et al. [85] shows that the QA might be useful in the context of finite-element problems. Their work shows that the shape of a finite-element sphere can be optimized to minimize the number of rays that, when emitted from a point-source, reflect towards a certain plane. This is not a typical finite-element problem as they are known within aerospace engineering, where the finite-element model is used to calculate the displacements and stresses within load-bearing structures. However, the fact that a sphere was defined using finite-elements, and a shape optimization was performed, indicates that perhaps the QA may also be useful within more typical aerospace optimization problems.

Since the QA shows promise for practical optimization problems, this thesis will focus on this topic. In aerospace engineering, finite-element problems are quite common and there is a demand for optimized lightweight structures, as these help to increase fuel efficiency and reduce operational costs. Thus, the goal is set to investigate a method to optimize simple finite-element structures using the QA. Since quantum computing, in general, is still a novel technology, the simplest type of finite-element structure will be investigated. As such, simple 2-dimensional truss structures are considered, and methods to cast an optimization of such structures into a QUBO form are presented in this thesis. Before this work is shown however, the upcoming chapter will start with an introduction to practical optimization problems, by showing how the QA can be used to solve the Traveling Salesman Problem.

3

Introduction to Practical QUBO Problems

In this chapter, the application of quantum annealing to practical problems will be discussed. As an introduction to the method of formulating QUBO problems, the well-known Traveling Salesman Problem (TSP) will be shown.

3.1. Quadratic Unconstrained Binary Optimization

For any problem that one wishes to solve using a QA, the first step is to recast the problem into either a QUBO or Ising formulation. Throughout this report, the QUBO formulation is preferred, since its binary nature allows for problems to be defined using a single matrix. In turn, this leads to a more intuitive understanding of the nature of QUBO problems. The goal for this chapter is to familiarize the reader with simple QUBO problems and attempt to instill some intuitive understanding of the nature of these problems.

A QUBO problem can be wholly defined through an $N \times N$ matrix $\mathbf{Q}$. The QA attempts to find the optimal bitstring $\mathbf{x}$ of length N that minimizes the Hamiltonian energy H, as shown in Eq. (3.1) [36].

$$\begin{aligned} &\min(H) = \mathbf{x}^T \mathbf{Q} \mathbf{x} \\ &s.t. \quad x_i \in \{0,1\} \, \forall \, i \in \{1,2,\ldots,N\} \end{aligned} \tag{3.1}$$

To set up an optimization problem for use with the QA, the main challenge is therefore to find a way to define an objective function that fits in the square matrix $\mathbf{Q}$. The diagonal entries of the QUBO matrix are the linear terms of the objective function, while all off-diagonal entries define quadratic terms in the objective function. Although not strictly necessary, it is convenient to write $\mathbf{Q}$ as an upper-diagonal matrix, with all entries below the diagonal set equal to zero.

As an example of a small arbitrary QUBO problem, consider the objective function defined in Eq. (3.2).

$$H = 1x_1 + 2x_2 + 3x_3 - 4x_1x_2 + 5x_1x_3 - 6x_2x_3 \tag{3.2}$$

The matrix $\mathbf{Q}$ that corresponds to the objective function in Eq. (3.2) must then be defined as shown in Eq. (3.3). Correspondingly, the bitstring of unknown variables is defined as in Eq. (3.4).

$$\mathbf{Q} = \begin{bmatrix} 1 & -4 & 5 \\ 0 & 2 & -6 \\ 0 & 0 & 3 \end{bmatrix} \tag{3.3}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3.4}$$

The Hamiltonian energy is then calculated by performing $\mathbf{x}^T\mathbf{Q}\mathbf{x}$. Performing this calculation will technically yield $H = 1x_1^2 + 2x_2^2 + 3x_3^2 - 4x_1x_2 + 5x_1x_3 - 6x_2x_3$. However, since the variables in $\mathbf{x}$ are binary, and therefore must be in $\{0,1\}$, it follows that $x_i^2 = x_i$. This simplification is what enables the use of both 'linear' and 'quadratic' terms in the QUBO matrix. Coefficients on the matrix diagonal are multiplied twice by the same variable, yielding a linear contribution to the objective function, and off-diagonal coefficients are multiplied with two different variables, therefore giving a quadratic contribution.

A convenient way to show QUBO matrices, and aid in interpreting them directly, is to include the set of variables along the top and side of the matrix. This allows for quick identification of linear and quadratic terms. An example is shown in Eq. (3.5).

$$\mathbf{Q} = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 \end{matrix} \\ \begin{bmatrix} 1 & -4 & 5 \\ 0 & 2 & -6 \\ 0 & 0 & 3 \end{bmatrix} \end{array} \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} \tag{3.5}$$

In the next section, an introduction to practical applications of QUBO problems is discussed, based on the well-known Traveling Salesman Problem.

3.2. Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a well-known problem that is difficult to solve classically. The TSP can be phrased as a question:

> *Given a number of cities, what is the shortest possible route through all cities that returns to the starting location?*

The TSP is known to be an NP-complete problem [55]. This means that, to find a solution to the problem, it will take a Non-deterministic Polynomial (NP) amount of time. Put simply, it is a problem for which classical solution methods are inefficient, and once the problem has grown beyond a certain scale, it becomes practically impossible to solve. However, it will be shown that the TSP lends itself well to being formulated as a QUBO problem, which means that it can be solved quite straightforwardly using the QA.

A mathematical formulation for the TSP is given by Lucas, and is an extension of the Hamiltonian cycle problem [55]. The Hamiltonian cycle problem asks whether, for a connected graph of nodes, a route exists that visits every node only once, and returns to the origin. The only difference between the Hamiltonian cycle problem and the TSP is that the TSP includes the distances between nodes, and asks for the shortest possible route. An application of the mathematical formulation provided by Lucas is given by Feld et al. [34].

However, for this chapter, it is more useful to set up a QUBO formulation for the TSP using a more intuitive approach. Rather than investigating the exact mathematical definitions, it will be shown that the TSP can also directly be cast into a QUBO matrix. By setting up the TSP from first principles, a foundation will be laid for the basic understanding of QUBO problems, which will be useful for the following chapters in this thesis.

3.2.1. Beginnings of a TSP QUBO

When setting up a QUBO problem, by directly creating a QUBO matrix, the first step is to consider the number of variables needed to define the problem. Furthermore, it is important to consider what the desired output of the optimization should be, and how the solution bitstring should be interpreted. To this end, the basic information for a simple TSP is first given, as shown in the map in Fig. 3.1.

A fully connected set of four cities is used to define the TSP. These cities will be named A, B, C, and D. The route that the salesman will use to travel through these cities can be defined in four steps, by sequentially indicating which city the salesman visits. For example, a route might be (B, A, C, D), indicating that the salesman starts in city B, and travels through A, C, and D. It is a given that the salesman must return to the origin city, city B, so there is no need to explicitly include this in the route (B, A, C, D).

The presence of the salesman in a particular city is to be indicated with a binary variable. If the salesman can be in any of the four cities at each of the four points that define his route, then four sets of four binary variables could be used to define his route. Therefore, the QUBO problem can be defined using a total of 16 variables, resulting in a 16×16 QUBO matrix. The solution bitstring can then be set up as shown in Eq. (3.6). A set of four bits defines the salesman's presence in a particular city, with four sequential sets defining his route. For example, the route (B, A, C, D) would be defined by the bitstring shown in Eq. (3.7).

$$\mathbf{x} = \begin{bmatrix} x_{A1} & x_{B1} & x_{C1} & x_{D1} & x_{A2} & x_{B2} & x_{C2} & x_{D2} & x_{A3} & x_{B3} & x_{C3} & x_{D3} & x_{A4} & x_{B4} & x_{C4} & x_{D4} \end{bmatrix}^T \tag{3.6}$$

$$\mathbf{x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \tag{3.7}$$

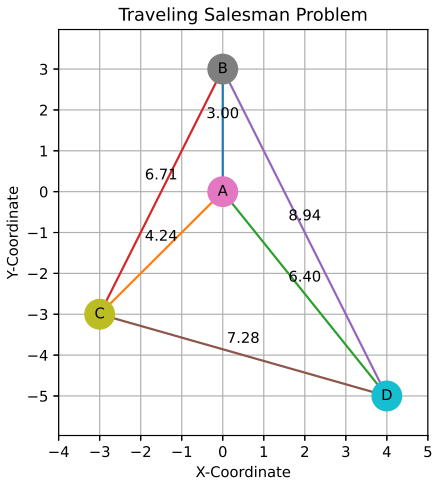


Figure 3.1: Example TSP

It is important to know what the solution bitstring represents when setting up QUBO problems. Now that this has been discussed, and the desired output format has been chosen, the next task is to start defining the TSP by filling in the QUBO matrix.

3.2.2. Distances

The main objective of the TSP is to find the shortest possible cycle that connects all cities. As such, the defining feature of the TSP, the distances between cities, must be added to the QUBO matrix. The next question is *where* these distances should be placed in the QUBO matrix.

Considering that the solution bitstring is interpreted as four sets of four variables, the QUBO matrix as a whole can also be interpreted in 4×4 sized submatrix sections. The submatrices on the diagonal then relate to a choice of a particular city, while the off-diagonal submatrices relate two choices of cities to each other. Since the off-diagonal submatrices relate two choices of cities to each other, these provide the ideal opportunity to include the distances between cities.

The distances will not need to be added to all of the off-diagonal submatrices in the QUBO matrix. Instead, the distance must only be added to the off-diagonal that relates two consecutive choices of cities. For example, the off-diagonal submatrix that relates the first and second cities to each other should contain the information about the distance between these two cities. However, the off-diagonal submatrix that relates the first and third cities to each other does not represent a portion of the route that the salesman will travel. Since, the salesman will be traveling consecutively from the first to second city, from the second to third city, from the third to fourth city, and returning from the fourth to first city. Therefore, off-diagonal submatrices that relate non-consecutive cities cannot contribute useful information about the route. Note that this means that the distances for the last step of the TSP, traveling from the last city back to the origin, must be placed in the upper-right corner of the QUBO matrix. A schematic representation of the full QUBO matrix, divided into submatrices, is shown in Eq. (3.8).

$$\left[\begin{array}{c|c|c|c} \text{Choice 1} & \text{Distance } 1 \rightarrow 2 & 0 & \text{Distance } 4 \rightarrow 1 \\ \hline 0 & \text{Choice 2} & \text{Distance } 2 \rightarrow 3 & 0 \\ \hline 0 & 0 & \text{Choice 3} & \text{Distance } 3 \rightarrow 4 \\ \hline 0 & 0 & 0 & \text{Choice 4} \end{array}\right] \tag{3.8}$$

The distances between cities will therefore be defined using a 4×4 matrix. This distance matrix must then be added to the QUBO matrix at the four relevant off-diagonal positions, as indicated in Eq. (3.8). The distance matrix, which includes the distances between all possible combinations of cities, can then be defined as shown in Eq. (3.9). Note, the diagonal entries in this matrix represent the distance from a city to itself, and will therefore be zero. Furthermore, the matrix will be symmetric, since, for example, the distance from City A to City B is the same as the distance from City B to City A. Filling in the distance matrix for the example TSP, as was shown in Fig. 3.1, gives the distance matrix in Eq. (3.10).

$$\mathbf{D} = \begin{array}{c} \begin{matrix} A & B & C & D \end{matrix} \\ \begin{bmatrix} d_{AA} & d_{AB} & d_{AC} & d_{AD} \\ d_{BA} & d_{BB} & d_{BC} & d_{BD} \\ d_{CA} & d_{CB} & d_{CC} & d_{CD} \\ d_{DA} & d_{DB} & d_{DC} & d_{DD} \end{bmatrix} \end{array} \begin{matrix} A \\ B \\ C \\ D \end{matrix} \tag{3.9}$$

$$\mathbf{D} = \begin{array}{c} \begin{matrix} A & B & C & D \end{matrix} \\ \begin{bmatrix} 0 & 3.00 & 4.24 & 6.40 \\ 3.00 & 0 & 6.71 & 8.94 \\ 4.24 & 6.71 & 0 & 7.28 \\ 6.40 & 8.94 & 7.28 & 0 \end{bmatrix} \end{array} \begin{matrix} A \\ B \\ C \\ D \end{matrix} \tag{3.10}$$

Adding the distance matrix to the QUBO matrix in the four correct positions yields the QUBO matrix shown in Eq. (3.11), which defines the distances between cities, for each step in the route for the TSP.

$$\mathbf{Q} = \left[\begin{array}{cccc|cccc|cccc|cccc} 0 & 0 & 0 & 0 & 0 & 3.00 & 4.24 & 6.40 & 0 & 0 & 0 & 0 & 0 & 3.00 & 4.24 & 6.40 \\ & 0 & 0 & 0 & 3.00 & 0 & 6.71 & 8.94 & 0 & 0 & 0 & 0 & 3.00 & 0 & 6.71 & 8.94 \\ & & 0 & 0 & 4.24 & 6.71 & 0 & 7.28 & 0 & 0 & 0 & 0 & 4.24 & 6.71 & 0 & 7.28 \\ & & & 0 & 6.40 & 8.94 & 7.28 & 0 & 0 & 0 & 0 & 0 & 6.40 & 8.94 & 7.28 & 0 \\ \hline & & & & 0 & 0 & 0 & 0 & 0 & 3.00 & 4.24 & 6.40 & 0 & 0 & 0 & 0 \\ & & & & & 0 & 0 & 0 & 3.00 & 0 & 6.71 & 8.94 & 0 & 0 & 0 & 0 \\ & & & & & & 0 & 0 & 4.24 & 6.71 & 0 & 7.28 & 0 & 0 & 0 & 0 \\ & & & & & & & 0 & 6.40 & 8.94 & 7.28 & 0 & 0 & 0 & 0 & 0 \\ \hline & & & & & & & & 0 & 0 & 0 & 0 & 0 & 3.00 & 4.24 & 6.40 \\ & & & & & & & & & 0 & 0 & 0 & 3.00 & 0 & 6.71 & 8.94 \\ & & & & & & & & & & 0 & 0 & 4.24 & 6.71 & 0 & 7.28 \\ & & & & & & & & & & & 0 & 6.40 & 8.94 & 7.28 & 0 \\ \hline & & & & & & & & & & & & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & & & 0 & 0 & 0 \\ & & & & & & & & & & & & & & 0 & 0 \\ & & & & & & & & & & & & & & & 0 \end{array}\right] \tag{3.11}$$

3.2.3. Constraints

Although the QUBO matrix for the TSP as given in Eq. (3.11) defines the distances to all cities, along every part of the route, the QA would still not successfully yield a valid solution to the problem. This is because there are not yet any constraints in place that help the QA to find valid solutions. In this section, the constraints necessary for creating a functional QUBO formulation of the TSP will be discussed and shown.

One can observe that the QUBO matrix in Eq. (3.11) only contains positive terms. This means that the Hamiltonian energy can only have values greater than, or equal to zero. The minimum energy solution would in this case therefore have exactly zero Hamiltonian energy. However, it can also be seen that many different solutions could yield zero-energy solutions. To see this, it is only necessary to identify pairs of variables that point towards zero-valued coefficients in the QUBO matrix. For example, using the arbitrary bitstring from Eq. (3.12) and performing $\mathbf{x}^T\mathbf{Q}\mathbf{x}$ can be seen to yield a zero-energy solution. Using this bitstring, the relevant terms that 'contribute' to the zero-energy solution are bolded and underlined in the QUBO matrix in Eq. (3.13).

$$\mathbf{x} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{3.12}$$

$$
\begin{array}{cccc|cccc|cccc|cccc|c}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & \\
\underline{\mathbf{0}} & 0 & \underline{\mathbf{0}} & 0 & 0 & 3.00 & 4.24 & 6.40 & 0 & \underline{\mathbf{0}} & 0 & \underline{\mathbf{0}} & 0 & 3.00 & 4.24 & 6.40 & 1 \\
 & 0 & 0 & 0 & 3.00 & 0 & 6.71 & 8.94 & 0 & 0 & 0 & 0 & 3.00 & 0 & 6.71 & 8.94 & 0 \\
 & & \underline{\mathbf{0}} & 0 & 4.24 & 6.71 & 0 & 7.28 & 0 & \underline{\mathbf{0}} & 0 & \underline{\mathbf{0}} & 4.24 & 6.71 & 0 & 7.28 & 1 \\
 & & & 0 & 6.40 & 8.94 & 7.28 & 0 & 0 & 0 & 0 & 0 & 6.40 & 8.94 & 7.28 & 0 & 0 \\
\hline
 & & & & 0 & 0 & 0 & 0 & 0 & 3.00 & 4.24 & 6.40 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & 0 & 0 & 0 & 3.00 & 0 & 6.71 & 8.94 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & 0 & 0 & 4.24 & 6.71 & 0 & 7.28 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & 0 & 6.40 & 8.94 & 7.28 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
 & & & & & & & & 0 & 0 & 0 & 0 & 0 & 3.00 & 4.24 & 6.40 & 0 \\
 & & & & & & & & & \underline{\mathbf{0}} & 0 & \underline{\mathbf{0}} & 3.00 & 0 & 6.71 & 8.94 & 1 \\
 & & & & & & & & & & 0 & 0 & 4.24 & 6.71 & 0 & 7.28 & 0 \\
 & & & & & & & & & & & \underline{\mathbf{0}} & 6.40 & 8.94 & 7.28 & 0 & 1 \\
\hline
 & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & & 0 & 0 & 0 \\
 & & & & & & & & & & & & & & & 0 & 0
\end{array}
\tag{3.13}
$$

The arbitrary zero-energy solution bitstring from Eq. (3.12) is actually an invalid result. Since, for the first and third step of the route two cities are simultaneously chosen, while for the second and fourth steps in the route no choice is made at all. This type of behavior is undesirable, as the lowest energy solution should be a valid solution, where exactly one city is chosen for each stop along the route. This problem can be amended by using a penalty function that causes the Hamiltonian energy of invalid solutions to become unfavorable compared to valid solutions. The first penalty function that will be added to the TSP QUBO is one that will ensure exactly one city is chosen for every stop in the route.

Unary Constraint

The penalty function that helps the QA select exactly one variable from a given set is sometimes known as the unary constraint. The unary constraint and its mathematical background are discussed in more detail in Section 4.2.2 of this thesis. However, the essence of the unary constraint, in matrix form, is straightforward to understand. For this particular TSP, one of the four cities must be chosen, for each stop along the route. This means that for each set of four binary variables, only one may have a value of 1, while the others must have a value of 0. The penalty matrix for the unary constraint that would promote this behavior is shown in Eq. (3.14). With the strength of the unary constraint set to a value of λ, note the structure of this penalty matrix: $-\lambda$ is written on the diagonal, while all off-diagonal terms have a value of 2λ.

$$
\mathbf{U} = \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \left[\begin{array}{cccc} -\lambda & 2\lambda & 2\lambda & 2\lambda \\ 0 & -\lambda & 2\lambda & 2\lambda \\ 0 & 0 & -\lambda & 2\lambda \\ 0 & 0 & 0 & -\lambda \end{array}\right] \end{array} \begin{array}{c} A \\ B \\ C \\ D \end{array}
\tag{3.14}
$$

The lowest energy state for this penalty matrix, with a Hamiltonian energy of $-\lambda$, is found when exactly one of the four variables is equal to 1, with the remaining variables being 0. Given that this unary constraint submatrix acts on four binary variables, it has a total of $2^4 = 16$ unique possible solution states. These are indicated in Table 3.1, including the corresponding Hamiltonian energy. It can be seen that the non-valid (NV) solution states lead to higher Hamiltonian energies than the valid (V) solution states, making the non-valid solutions less favorable. Thus, this 4×4 submatrix can be added to the main TSP QUBO matrix along the diagonal, to ensure that at each point along the route only one of the four cities is selected. This then yields the TSP QUBO matrix shown in Eq. (3.15).

Validity	NV	V	V	NV	V	NV	NV	NV	V	NV	NV	NV	NV	NV	NV	NV
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
H	0	-1	-1	0	-1	0	0	3	-1	0	0	3	0	3	3	8

Table 3.1: Table containing every solution of the unary constraint acting on four variables, with a constraint strength of $\lambda = 1$.

$$\mathbf{Q} = \left[\begin{array}{cccc|cccc|cccc|cccc}
-\lambda & 2\lambda & 2\lambda & 2\lambda & 0 & 3.00 & 4.24 & 6.40 & 0 & 0 & 0 & 0 & 0 & 3.00 & 4.24 & 6.40 \\
 & -\lambda & 2\lambda & 2\lambda & 3.00 & 0 & 6.71 & 8.94 & 0 & 0 & 0 & 0 & 3.00 & 0 & 6.71 & 8.94 \\
 & & -\lambda & 2\lambda & 4.24 & 6.71 & 0 & 7.28 & 0 & 0 & 0 & 0 & 4.24 & 6.71 & 0 & 7.28 \\
 & & & -\lambda & 6.40 & 8.94 & 7.28 & 0 & 0 & 0 & 0 & 0 & 6.40 & 8.94 & 7.28 & 0 \\
\hline
 & & & & -\lambda & 2\lambda & 2\lambda & 2\lambda & 0 & 3.00 & 4.24 & 6.40 & 0 & 0 & 0 & 0 \\
 & & & & & -\lambda & 2\lambda & 2\lambda & 3.00 & 0 & 6.71 & 8.94 & 0 & 0 & 0 & 0 \\
 & & & & & & -\lambda & 2\lambda & 4.24 & 6.71 & 0 & 7.28 & 0 & 0 & 0 & 0 \\
 & & & & & & & -\lambda & 6.40 & 8.94 & 7.28 & 0 & 0 & 0 & 0 & 0 \\
\hline
 & & & & & & & & -\lambda & 2\lambda & 2\lambda & 2\lambda & 0 & 3.00 & 4.24 & 6.40 \\
 & & & & & & & & & -\lambda & 2\lambda & 2\lambda & 3.00 & 0 & 6.71 & 8.94 \\
 & & & & & & & & & & -\lambda & 2\lambda & 4.24 & 6.71 & 0 & 7.28 \\
 & & & & & & & & & & & -\lambda & 6.40 & 8.94 & 7.28 & 0 \\
\hline
 & & & & & & & & & & & & -\lambda & 2\lambda & 2\lambda & 2\lambda \\
 & & & & & & & & & & & & & -\lambda & 2\lambda & 2\lambda \\
 & & & & & & & & & & & & & & -\lambda & 2\lambda \\
 & & & & & & & & & & & & & & & -\lambda
\end{array}\right] \tag{3.15}$$

Repeated Visit Constraint

With the QUBO matrix given in Eq. (3.15), the distances between cities have been defined, and thanks to the unary constraint the QA should choose only one city for each point along the route of the traveling salesman. However, one problem remains: nothing is preventing the traveling salesman from just staying in the same city all the time, and not traveling to any of the other cities. Phrased differently, a constraint is needed that prevents the traveling salesman from visiting the same city multiple times along his route.

Recall that distances are defined using off-diagonal 4×4 submatrices. When the distances between the cities were defined, with the submatrix shown in Eq. (3.10), the distance from a city to itself was left at a value of zero. However, these zero entries on the submatrix diagonal can be changed to prevent the traveling salesman from visiting the same city multiple times. Setting the repeated visitation penalty strength to a value of γ, the penalty submatrix would take the form as shown in Eq. (3.16). This repeated visitation submatrix must be added to all off-diagonal submatrices in the main QUBO matrix, since a repeated visitation must be prevented at all points in the route. Therefore, adding the repeated visitation constraint to the main TSP QUBO matrix, the final QUBO matrix shown in Eq. (3.17) is found.

$$\mathbf{V} = \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \left[\begin{array}{cccc} \gamma & 0 & 0 & 0 \\ 0 & \gamma & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & \gamma \end{array}\right] \end{array} \begin{array}{c} A \\ B \\ C \\ D \end{array} \tag{3.16}$$

$$\mathbf{Q} = \left[\begin{array}{cccc|cccc|cccc|cccc}
-\lambda & 2\lambda & 2\lambda & 2\lambda & \gamma & 3.00 & 4.24 & 6.40 & \gamma & 0 & 0 & 0 & \gamma & 3.00 & 4.24 & 6.40 \\
 & -\lambda & 2\lambda & 2\lambda & 3.00 & \gamma & 6.71 & 8.94 & 0 & \gamma & 0 & 0 & 3.00 & \gamma & 6.71 & 8.94 \\
 & & -\lambda & 2\lambda & 4.24 & 6.71 & \gamma & 7.28 & 0 & 0 & \gamma & 0 & 4.24 & 6.71 & \gamma & 7.28 \\
 & & & -\lambda & 6.40 & 8.94 & 7.28 & \gamma & 0 & 0 & 0 & \gamma & 6.40 & 8.94 & 7.28 & \gamma \\
\hline
 & & & & -\lambda & 2\lambda & 2\lambda & 2\lambda & \gamma & 3.00 & 4.24 & 6.40 & \gamma & 0 & 0 & 0 \\
 & & & & & -\lambda & 2\lambda & 2\lambda & 3.00 & \gamma & 6.71 & 8.94 & 0 & \gamma & 0 & 0 \\
 & & & & & & -\lambda & 2\lambda & 4.24 & 6.71 & \gamma & 7.28 & 0 & 0 & \gamma & 0 \\
 & & & & & & & -\lambda & 6.40 & 8.94 & 7.28 & \gamma & 0 & 0 & 0 & \gamma \\
\hline
 & & & & & & & & -\lambda & 2\lambda & 2\lambda & 2\lambda & \gamma & 3.00 & 4.24 & 6.40 \\
 & & & & & & & & & -\lambda & 2\lambda & 2\lambda & 3.00 & \gamma & 6.71 & 8.94 \\
 & & & & & & & & & & -\lambda & 2\lambda & 4.24 & 6.71 & \gamma & 7.28 \\
 & & & & & & & & & & & -\lambda & 6.40 & 8.94 & 7.28 & \gamma \\
\hline
 & & & & & & & & & & & & -\lambda & 2\lambda & 2\lambda & 2\lambda \\
 & & & & & & & & & & & & & -\lambda & 2\lambda & 2\lambda \\
 & & & & & & & & & & & & & & -\lambda & 2\lambda \\
 & & & & & & & & & & & & & & & -\lambda
\end{array}\right] \tag{3.17}$$

The last step before the TSP QUBO matrix is complete is to choose appropriate values for the strength of the unary and repeated visit constraints. Since these two constraints are considered equally important [34], the values of λ and γ will be chosen equal to each other. The constraint strength must be chosen to be high enough that the constraints are always obeyed. However, choosing constraint strength values that are excessively high can cause the distances for the TSP to become relatively less significant. This can make it more difficult for the QA to find the optimal solution. Therefore, fine-tuning the constraint strength can be a trial-and-error process. Typically the constraint strength must be higher than the highest valued entry in the QUBO matrix. An initial guess for the constraint strength of $\lambda = \gamma = 20$, which is approximately twice the maximum road length, was seen to provide satisfactory results.

3.2.4. Embedding the TSP

Once the final QUBO matrix has been produced, the problem is nearly ready to be submitted to the QA. The last remaining step is to find an embedding that will fit the problem described by the QUBO matrix from Eq. (3.17) onto the hardware of the QPU. By counting the number of off-diagonal non-zero entries in, for example, the top row of the QUBO matrix, it can be seen that the first binary problem variable must form connections with 12 other problem variables. Since the quantum annealing hardware only supports up to six connections per qubit, this leads to the troublesome situation where the problem will not directly fit on the QPU. To remedy this issue, an embedding must be found that uses chains of physical qubits to represent the individual logical problem variables. By using physical qubit chains to represent single logical problem variables, the total connectivity of the embedded logical variables can be increased to meet the demands of the problem at hand. For this small TSP such an embedding can be found easily, and in a fraction of a second, by using the *minorminer* Python tool provided by D-Wave, which is openly available online [19].

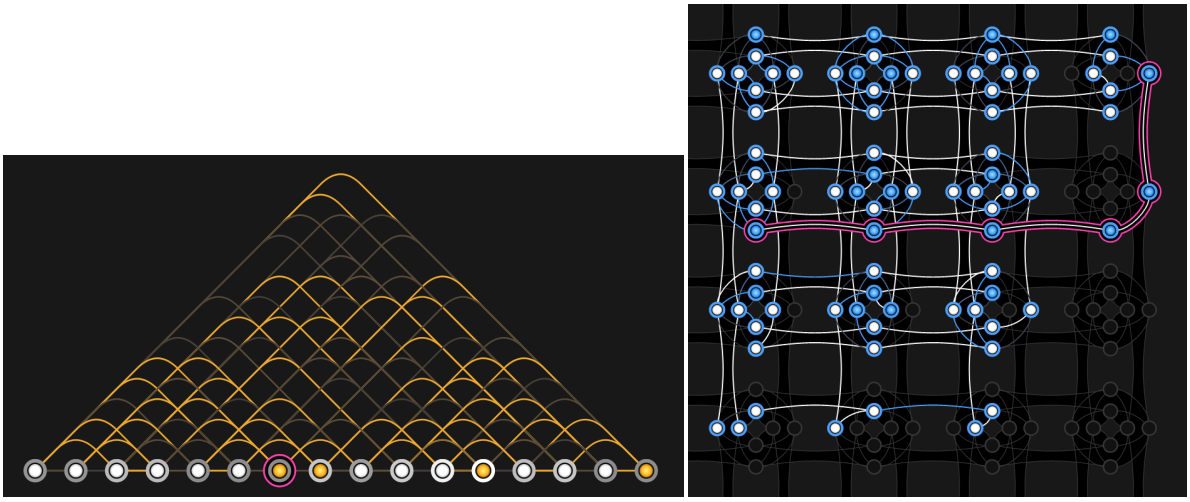
A schematic of the logical variable connectivity structure of the TSP is shown Fig. 3.2a. This schematic was produced from the results of a quantum annealing attempt, using the D-Wave Inspector tool [25]. Each circle in the lower portion of the schematic represents one of the logical problem variables, of which there are 16 in total. The circles are filled in with either white or orange coloring, depending on the solution that QA found, with white and orange corresponding to 0-bits and 1-bits respectively. This solution will be discussed later. The variables are (unfortunately) presented in a seemingly arbitrary ordering. The lines connecting logical variables represent the non-zero off-diagonal terms in the QUBO matrix, where the color intensity of the line (dark or bright orange) corresponds to the magnitude of the QUBO matrix term. It can be seen that nearly all variables are connected, but there are still a number of gaps in the connectivity structure.

The physical embedding that was found for the logical problem structure is shown in Fig. 3.2b. This schematic shows the physical layout of the QPU, with each of the circles in the schematic representing physical qubits. The color that these circles are filled with relates to the solution that the QA found to the problem, which will be discussed later, with white and blue corresponding to the Ising states -1 and +1 respectively. Even though a QUBO problem was submitted, the software shows the results in the schematic of the physical embedding in an Ising form. The physical qubits are connected with either blue or white lines, with the color intensity corresponding to the strength of the coupling. Negative couplings are white, and help coupled qubits fall into similar final states. There are no negative off-diagonal terms in the QUBO matrix, so it can

be concluded that all white lines in this schematic belong to chained qubits, which are collectively acting as individual logical variables. Positive couplings are shown as blue lines between qubits, and steer coupled qubits toward opposite states. The blue lines therefore correspond to the positive off-diagonal values in the QUBO matrix.

When the QA solves problems it is possible that qubit chains are 'broken'. This means that not every qubit in the chain has the same final solution state. Broken qubit chains are not desirable, since the solution state for the logical variable that they represent then becomes ambiguous. Thus, the *chain strength* parameter of the QPU must be set to a value that ensures that chain breaks do not occur. In this case, the chain strength was set to a value of $2\lambda = 40$. This value was chosen since it is the same as the maximum value in the QUBO matrix.

A single logical problem variable has been highlighted in purple in Fig. 3.2a. The embedded physical chain of qubits that represent this logical variable is also highlighted in purple in Fig. 3.2b. It is clear that the physical embedding of this small TSP is already quite complicated, and uses far more physical qubits than the number of logical problem variables might initially suggest. To be exact, while the logical problem only has 16 variables, the embedding uses 81 physical qubits. However, having finally also set up an embedding for the TSP, the problem can indeed be solved by the QA.



(a) Logical problem connectivity structure.

(b) Physical embedding.

Figure 3.2: A highlighted logical problem variable and its physically embedded counterpart. Figures produced using the D-Wave Inspector tool [25].

3.2.5. Results

Submitting the TSP to the QA, using an embedding that ensures the problem fits onto the QPU hardware, a solution to the problem is found. The QA finds the solution bitstring given in Eq. (3.18). This solution represents the route (C, D, A, B), shown in Fig. 3.3. The route has a total length of 23.39 units. Although the proposed route starts and ends in city C, traveling the cycle counterclockwise, alternate solutions can also be found that consist of the same cycle, but start with different cities or travel the cycle in the clockwise direction.

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T \quad (3.18)$$

To confirm the result of the QA, a simple exhaustive analysis is also performed, calculating the total length of every possible route. This analysis confirms that the QA has indeed found the minimum length cycle, and that there are eight variations of this cycle in total, each with different starting positions and travel directions. The Python code that was used to produce the quantum annealing and exhaustive analysis results is available online [93].

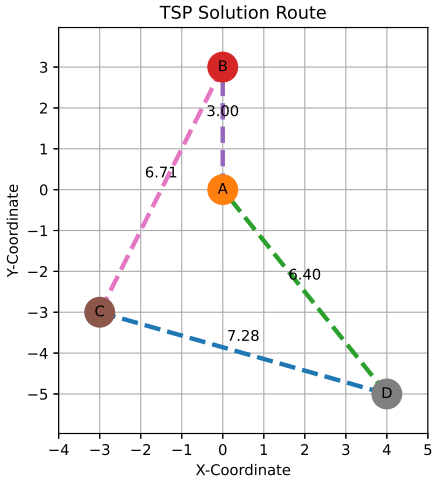


Figure 3.3: Optimal route for the TSP.

3.2.6. Final Comments

To close this chapter, some final comments are offered regarding the Traveling Salesman Problem and the QUBO formulation that was presented.

It has been mentioned that a mathematical foundation for the TSP QUBO has been laid out by Lucas, which has been implemented by Feld et al. Yet, in this chapter, a different formulation was provided based purely on an intuitive understanding of QUBO problems. The reason for providing this alternative formulation was to instill a similar intuitive understanding of QUBO problems in a reader who might be relatively new to this topic. Nevertheless, comparing the formulations provided in this chapter and the one implemented by Feld et al. reveals that the final QUBO matrices have many similarities. The main difference between the two methods is that the solution bitstring is interpreted differently, which in turn causes the data in the QUBO matrix to be structured differently.

The approach taken in the intuitive formulation leads to a solution bitstring that is interpreted as four sequential choices of different cities. In other words, the bitstring consists of groups of 4 variables that each represent a different city, and the bitstring is interpreted sequentially. This leads to a solution bitstring as shown in Eq. (3.19). However, the solution bitstring used by Feld et al. is ordered in the manner shown in Eq. (3.20). In their implementation, the bitstring consists of groups of 4 variables that each represent the same city, and the ordering of the cities is determined within these groups. Overall, the differences between the two methods are mostly superficial, and both methods lead to functional QUBO formulations of TSP problems. In the opinion of this author, the intuitive method described in this chapter yields a QUBO matrix that is easier to understand.

$$\mathbf{x} = \begin{bmatrix} x_{A1} & x_{B1} & x_{C1} & x_{D1} & x_{A2} & x_{B2} & x_{C2} & x_{D2} & x_{A3} & x_{B3} & x_{C3} & x_{D3} & x_{A4} & x_{B4} & x_{C4} & x_{D4} \end{bmatrix}^T \tag{3.19}$$

$$\mathbf{x} = \begin{bmatrix} x_{A1} & x_{A2} & x_{A3} & x_{A4} & x_{B1} & x_{B2} & x_{B3} & x_{B4} & x_{C1} & x_{C2} & x_{C3} & x_{C4} & x_{D1} & x_{D2} & x_{D3} & x_{D4} \end{bmatrix}^T \tag{3.20}$$

One aspect that was excluded from the formulation, for simplicity, is a constraint that allows for certain road sections to be considered invalid. For this simple TSP, with only four cities, it was assumed that all cities are connected to each other via roads. However, this may not realistically be the case, as certain roads might simply not exist, or are perhaps blocked by traffic. If this is the case, such roads can be penalized by simply artificially increasing their length. If there are shorter valid routes available, the QA will try to find these instead. The work by Feld et al. provides some additional mathematical detail, although this penalization was also not relevant to their fully connected network of cities.

4

Truss Sizing Optimization: Direct QUBO Method

Truss structures are considered one of the simplest types of structures that an engineer may want to analyze and optimize. Typically, a truss structure is optimized to have minimal total mass, while being compliant with various stiffness or strength requirements. In this chapter, the truss structure optimization is discussed, and a method of casting this problem into the QUBO formulation is proposed.

4.1. Overview

One of the most basic types of structures that engineers learn to analyze are truss structures. Various optimization problems can be imagined for truss systems. Typically, an objective will be to minimize the total mass of the truss system, whilst obeying constraints on the allowable stress, displacement, or stability of the structure. In general, there are three different types of optimization frameworks that can be applied, namely, sizing optimization, shape optimization, and topology optimization. In the context of a truss structure optimization, a sizing optimization will aim to uncover the optimal solution by allowing the truss cross-sections to be altered. A shape optimization will aim to optimize the objective by moving the position of the nodes upon which trusses are defined. Topology optimization would aim to find the optimal set of trusses by allowing certain members to disappear altogether [77].

Classically, the truss optimization problem is well-studied. Especially the truss sizing optimization has received much attention, with many different approaches being formulated. One distinction between methods is that some rely on continuous design variables, while others attempt to find an optimum by considering a set of discrete design variables. An overview of classical truss optimization methods, relying on discrete design variables been provided by Stolpe [82].

In the case of quantum annealing, the only variables that the annealer can use are binary variables. In classical computing, sets of binary variables can be used to represent continuous design variables using floating-point numbers. However, common floating-point number schemes require 32 or 64 bits to encode a single number. Due to the limited number of qubits available in the QA, it would be very impractical to implement a method that attempts to use continuous variables. Given that the state-of-the-art QA only has roughly 2000 qubits, and that these qubits are only sparsely connected to each other, it would be impossible to encode more than one or two actual floating-point numbers. Since, the floating-point numbers require full qubit connectivity [74], and the maximum sized fully connected problem could have only 64 variables at most on the D-Wave 2000Q QA [38]. A more efficient use of qubit resources would therefore be to formulate the problem by using binary variables that, in turn, relate to a set of discrete design variables. Certain classical methods also take this approach [82]. Seemingly, a good place to start with solving a practical truss optimization problem using a QA would therefore be to do a sizing optimization with discrete variables for the allowed cross-sectional areas for every truss.

4.2. Direct QUBO Formulation

In this section, the first attempt at formulating a truss sizing optimization method for the QA is described. This process will involve the definition of design variables, setting up an optimization objective, and adding constraints. Through these steps, the QUBO matrix that defines the optimization problem will be assembled directly.

4.2.1. Design Variables and Objective Function

When setting up a QUBO problem, it is critical to define the design variables in terms of the binary variables that the QA will use. Consider a sizing optimization for a system of N trusses, with each truss having a discrete choice of C different cross-sectional areas. Then defining $c \in \{1,2,\dots C\}$ and $n \in \{1,2,\dots N\}$, a set of possible cross-sections for truss n can be defined as shown in Eq. (4.1). In this manner, every truss in the truss system can have its own uniquely defined set of C discrete choices.

$$A_{n,set} = \{A_{n,1}, A_{n,2}, \dots, A_{n,C}\} \tag{4.1}$$

Correspondingly, to define the design variable, i.e. the cross-sectional area of truss n, we also need a set of qubits that each correspond to one of the possible choices of cross-sectional area, giving:

$$\begin{aligned} q_{n,set} &= \{q_{n,1}, q_{n,2}, \dots, q_{n,C}\} \\ \text{With:} \quad q_{n,c} &\in \{0,1\}\, \forall c \in \{1,2,\dots C\} \end{aligned} \tag{4.2}$$

Multiplying the entries in these sets, and performing a summation will allow for an expression of the total cross-sectional area of truss n to be defined.

$$A_n = \sum_{c=1}^{C} q_{n,c} A_{n,c} \tag{4.3}$$

In the case that, for truss n, only one of the qubits in $q_{n,set}$ is equal to 1, and the others are equal to 0, then this binary variable would correspond directly to a particular choice in cross-sectional area. Furthermore, if this is the case, the total truss system mass can be calculated by:

$$M = \sum_{n=1}^{N} \rho_n L_n \left(\sum_{c=1}^{C} q_{n,c} A_{n,c} \right) \tag{4.4}$$

In this expression, L_n and ρ_n are, respectively, the length and the material density of truss n. The total mass of the system, M, can then be used as the objective function for minimization. This objective function can also be written in a matrix format, for use within the QUBO problem formulation. The individual mass terms of every truss will then be located on the matrix diagonal, and correspond with the different choices in cross-sectional area.

As an example, allowing three different choices for the cross-sectional area, giving $c \in \{1,2,3\}$, the mass objective function matrix for a system with trusses $n \in \{1,2,\dots\}$ would generally look like the form given in Eq. (4.5). Note that the $q_{1,1}, q_{1,2}, \dots$ indicated above and to the side of the matrix are simply to help visualize to which qubit variable that particular row or column pertains, and have no mathematical meaning in this case.

$$Q_M = \begin{array}{c} \begin{matrix} q_{1,1} & q_{1,2} & q_{1,3} & q_{2,1} & q_{2,2} & q_{2,3} & \cdots \end{matrix} \\ \begin{bmatrix} \rho_1 L_1 A_{1,1} & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \rho_1 L_1 A_{1,2} & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \rho_1 L_1 A_{1,3} & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \rho_2 L_2 A_{2,1} & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \rho_2 L_2 A_{2,2} & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & \rho_2 L_2 A_{2,3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{matrix} q_{1,1} \\ q_{1,2} \\ q_{1,3} \\ q_{2,1} \\ q_{2,2} \\ q_{2,3} \\ \vdots \end{matrix} \tag{4.5}$$

In practice, the QA is limited in the precision and magnitude of the terms in the QUBO matrix that it can take into account. Specifically, the terms on the matrix diagonal, which in the physical QA correspond to linear qubit biases, must be scaled to fit within the range $[-2,2]$[1]. Out of convenience, and to fit the mass terms into the valid qubit bias range, the values in the mass objective function matrix can all be scaled such that the maximum becomes a dimensionless 'mass' term with a value of 1. This can be achieved simply by dividing all terms in the matrix by the maximum value from the set of all masses for all trusses. Thus the scaled mass objective QUBO matrix would be found as shown in Eq. (4.6).

[1]The valid range of qubit biases can be obtained directly by submitting a query to the D-Wave QA via: `DWaveSampler().properties['h_range']`

$$Q_{M^*} = \frac{1}{\max(Q_M)} \begin{array}{c} \begin{matrix} q_{1,1} & q_{1,2} & q_{1,3} & q_{2,1} & q_{2,2} & q_{2,3} & \cdots \end{matrix} \\ \begin{bmatrix} \rho_1 L_1 A_{1,1} & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \rho_1 L_1 A_{1,2} & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \rho_1 L_1 A_{1,3} & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \rho_2 L_2 A_{2,1} & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \rho_2 L_2 A_{2,2} & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & \rho_2 L_2 A_{2,3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{matrix} q_{1,1} \\ q_{1,2} \\ q_{1,3} \\ q_{2,1} \\ q_{2,2} \\ q_{2,3} \\ \vdots \end{matrix} \tag{4.6}$$

An abbreviated notation for the scaled mass terms will be assumed. Since all mass terms are located simply on the diagonal of the mass objective function QUBO matrix, and each term corresponds directly to the qubit $q_{n,c}$, the scaled mass terms on the diagonal of the matrix will be written simply as $M^*_{n,c}$.

4.2.2. Unary Constraint

With just the objective function, the setup of the problem is not yet complete. It was already briefly mentioned that, for every truss, only one of the possible cross-sections should be chosen. To accomplish this, a constraint must be added for every truss to ensure that exactly one cross-sectional area is chosen. However, since QUBO problems are, per definition, unconstrained, this instead involves the setup of a penalty function. The goal for the penalty function is then to penalize solutions that do not comply with the requirement of choosing only one cross-sectional area.

The constraint that must be satisfied, for every truss n in the system, can mathematically be expressed as:

$$\sum_{c=1}^{C} q_{n,c} = 1 \tag{4.7}$$

Since only one of the qubits on which this constraint acts must take a value of 1, while the others must all equal 0, this constraint is sometimes referred to as the *unary constraint*. Further elaboration on the unary constraint is given in [56]. In the context of the truss optimization problem, it enforces that only one cross-section is chosen per truss. However, in the form shown in Eq. (4.7), the constraint cannot be applied in the QUBO problem framework. This is because the constraint is currently written as an equality constraint, which per definition is incompatible with quadratic *unconstrained* binary optimization problems. For the constraint to become compatible with QUBO problems it must be rewritten as a minimization problem. A common method is to rewrite the equality constraint as a penalty function, using a 'squared-error' approach [50, 88]. This approach is also used, for example, in the works by Van Vreumingen et al. and Neukart et al. [62, 85]. Thus, the constraint can be rewritten as a minimization problem as shown in Eq. (4.8).

$$\begin{gathered} \left(\sum_{c=1}^{C} q_{n,c}\right) - 1 = 0 \\ \left(\left(\sum_{c=1}^{C} q_{n,c}\right) - 1\right)^2 = 0 \end{gathered} \tag{4.8}$$

Now, adding in a penalty scaling factor λ, the penalty function for the unary constraint becomes:

$$H_U = \lambda \left(\left(\sum_{c=1}^{C} q_{n,c}\right) - 1\right)^2 \tag{4.9}$$

For solutions that comply with the unary constraint, the penalty function from Eq. (4.9) will have a minimum value of $H_U = 0$. To show how to implement the unary constraint penalty function, an example is given. Imagine for every truss a choice out of three possible cross-sectional areas must be made. In that case, $C = 3$, and the penalty function can be expanded as shown in Eq. (4.10).

$$
\begin{aligned}
H_U =& \lambda \left(\left(\sum_{c=1}^{3} q_{n,c} \right) - 1 \right)^2 \\
H_U =& \lambda \left(q_{n,1} + q_{n,2} + q_{n,3} - 1 \right) \left(q_{n,1} + q_{n,2} + q_{n,3} - 1 \right) \\
H_U =& \lambda \left(q_{n,1}^2 + q_{n,1} q_{n,2} + q_{n,1} q_{n,3} - q_{n,1} \right. \\
& + q_{n,1} q_{n,2} + q_{n,2}^2 + q_{n,2} q_{n,3} - q_{n,2} \\
& + q_{n,1} q_{n,3} + q_{n,2} q_{n,3} + q_{n,3}^2 - q_{n,3} \\
& \left. - q_{n,1} - q_{n,2} - q_{n,3} + 1 \right)
\end{aligned} \tag{4.10}
$$

Knowing that $q_{n,c} \in \{0,1\}$ the expression can be simplified, since $q_{n,c}^2 = q_{n,c}$. Thus:

$$
H_U = \lambda \left(2q_{n,1} q_{n,2} + 2q_{n,2} q_{n,3} + 2q_{n,1} q_{n,3} - q_{n,1} - q_{n,2} - q_{n,3} + 1 \right) \tag{4.11}
$$

Lastly, the final constant term can be dropped, since it is independent of the qubit variables, and does not affect the minimization problem. Doing so makes the unary constraint penalty function compatible with the QUBO problem framework. Since, the penalty function can now be written as a pure summation of linear and quadratic terms, as shown in Eq. (4.12).

$$
H_U = \lambda \left(2q_{n,1} q_{n,2} + 2q_{n,1} q_{n,3} + 2q_{n,2} q_{n,3} - q_{n,1} - q_{n,2} - q_{n,3} \right) \tag{4.12}
$$

Ideally, the penalty function from Eq. (4.12) will prevent the QA from yielding invalid results. This means that, with this penalty function, the problematic situation can be avoided where the QA selects more than one, or none of the proposed cross-sectional areas for any of the trusses in the structure. For convenient implementation with QUBO problems, the penalty function can be rewritten into a matrix form. The unary constraint penalty function matrix Q_U will then have terms of $-\lambda$ on the diagonal, and the off-diagonals will have a value of 2λ.

$$
Q_U = \begin{bmatrix} -\lambda & 2\lambda & 2\lambda \\ 0 & -\lambda & 2\lambda \\ 0 & 0 & -\lambda \end{bmatrix} \tag{4.13}
$$

In this case, this penalty function matrix has dimensions 3×3, because it was initially assumed that there were three possible choices for the cross-sectional area of every truss. However, in the general case, with C possible choices, the unary constraint penalty matrix becomes a $C \times C$ matrix, with values of $-\lambda$ on the diagonal, and 2λ for the off-diagonal terms. In this form, the penalty matrix can straightforwardly be added to the main objective function QUBO matrix, for every truss in the truss optimization problem.

Giving a practical example, for a system of N trusses with $n \in \{1,2,\ldots\}$, and having three different choices in cross-sectional area, giving $C = 3$ and $c \in \{1,2,3\}$, the QUBO matrix would take form shown in Eq. (4.14). This matrix includes the scaled mass terms $M_{n,c}^*$ from the mass objective function, as well as the λ terms from the unary constraint, which is taken into account individually for every truss.

$$
Q_{M^*+U} = \begin{array}{c} \begin{matrix} q_{1,1} & q_{1,2} & q_{1,3} & q_{2,1} & q_{2,2} & q_{2,3} & \cdots \end{matrix} \\ \begin{bmatrix} M_{1,1}^* - \lambda & 2\lambda & 2\lambda & 0 & 0 & 0 & \cdots \\ 0 & M_{1,2}^* - \lambda & 2\lambda & 0 & 0 & 0 & \cdots \\ 0 & 0 & M_{1,3}^* - \lambda & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & M_{2,1}^* - \lambda & 2\lambda & 2\lambda & \cdots \\ 0 & 0 & 0 & 0 & M_{2,2}^* - \lambda & 2\lambda & \cdots \\ 0 & 0 & 0 & 0 & 0 & M_{2,3}^* - \lambda & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{matrix} \\ q_{1,1} \\ q_{1,2} \\ q_{1,3} \\ q_{2,1} \\ q_{2,2} \\ q_{2,3} \\ \vdots \end{matrix} \tag{4.14}
$$

4.2.3. Stress Constraint: Preliminary Information

The last constraint necessary for a useful sizing optimization of a truss structure would be a constraint on one of the structural responses when the structure is subjected to a particular load. This could be a constraint on the allowed displacement of the structure, the maximum allowed stress, or structural stability. However, for a

simple truss problem based on a linear finite-element method, an initial starting point would be to constrain the maximum stress.

Given a maximum allowable stress, the stress in each truss can be used to calculate the *reserve factor*. The reserve factor (RF) gives a measure of how close a structural member is to failure, and is calculated simply by dividing the allowable stress by the applied stress in the member, as shown in Eq. (4.15).

$$RF = \frac{\sigma_{allowable}}{\sigma_{applied}} \tag{4.15}$$

The RF indicates structural failure when its value goes below 1. Structures for which all reserve factors are above 1 can be considered safe. However, if the RF is much higher than 1, it essentially indicates that the structure is overdesigned, and there is an opportunity to save weight.

For truss structures it is assumed that trusses can only carry axial loads, being either tensile or compressive in nature. Furthermore, most materials have different allowables for tensile and compressive stresses. Therefore, if possible, a constraint on the reserve factor that takes separate allowables into account for tensile and compressive stresses would be ideal. However, a simplification is possible if the same value is assumed for both cases.

Nevertheless, the question remains, how would one implement a constraint on the truss reserve factors in a QUBO problem formulation? This is a deceptively difficult question to answer. Ideally, a mathematical expression for the reserve factor, written entirely in terms of binary qubit variables is desired. The expression should contain only linear and quadratic terms, such that it can straightforwardly be written in a matrix form, similar to that of the unary constraint. Such an approach is discussed in Chapter 5. However, the upcoming section shows a simpler and more direct method of implementing an RF constraint into the QUBO formulation.

4.2.4. Stress Constraint: RF Dependent Preference in an Optimization Scheme

The reserve factor, based on the stress the truss is experiencing, is calculated using the results of a finite-element analysis. However, it can be quite time-consuming to complete the process of setting up a finite-element problem, finding the solution, and post-processing the results to get a list of RFs. In an optimization procedure, which is often iterative in nature, it would therefore be desirable to perform as few FEM analyses as possible. This may be one way to reduce the total amount of time needed to find an optimal design for the truss structure. This section elaborates on a method of constraining the truss RFs, within an iterative optimization scheme, in a way that relies on only one FEM analysis per iteration. This method is also chronologically the first that was implemented and tested.

Given an iterative optimization scheme, the procedure is started by setting up and classically solving a finite-element problem for a system of trusses. The output of this analysis is a reserve factor value for every truss in the initial configuration. Then, it is assumed that there are three different discrete choices for the cross-sectional area of every truss. Namely, a smaller cross-sectional area, a larger area, or simply the same area as the truss already has. Thus, with $C = 3$, the following set for the possible choices of cross-sectional area, for truss n, can be defined:

$$A_{n,set} = \{A_{n,1}, A_{n,2}, A_{n,3}\} \tag{4.16}$$

The set of corresponding qubits then also becomes:

$$\begin{aligned} q_{n,set} &= \{q_{n,1}, q_{n,2}, q_{n,3}\} \\ \text{With:} \quad q_{n,c} &\in \{0,1\}\ \forall c \in \{1,2,3\} \end{aligned} \tag{4.17}$$

It is convenient to define the set of allowed choices for the cross-sectional area based on the currently assumed cross-sectional area, with the smaller and larger options differing by a constant amount dA. The set of possible choices for the cross-sectional area therefore becomes:

$$A_{n,set} = \{A_n - dA, A_n, A_n + dA\} \tag{4.18}$$

Based on the known RF for every truss, a guess can be made as to what the most logical choice would be for the next iteration of the optimization procedure. Given a truss n, with area A_n, area increment dA, reserve factor RF_n, and a predefined margin for the reserve factor RF_{margin}, the scheme in Eq. (4.19) is defined:

$$\begin{aligned} &\text{If } RF_n < 1 && \rightarrow \text{prefer } A_n + dA \\ &\text{If } 1 \leq RF_n \leq 1 + RF_{margin} && \rightarrow \text{prefer } A_n \\ &\text{If } RF_n > 1 + RF_{margin} && \rightarrow \text{prefer } A_n - dA \end{aligned} \tag{4.19}$$

By following the steps in this scheme, it is expected that the truss cross-sectional areas can be steered towards the configuration where all trusses approach $RF = 1$, thereby complying with the stress constraint. A basic QUBO problem has already been set up that defines the truss system weight as the objective, and utilizes the unary constraint to force the QA to choose one of the suggested cross-sectional areas. Because the QA naturally tends to seek solutions with a low Hamiltonian energy, the RF constraint preference for certain solutions can be added to the QUBO matrix by adding a negative term $-\gamma$ to specific matrix entries. The exact position in the QUBO matrix to which the RF preference term is added is then based on the three decision-making rules stated above.

As an example, suppose the reserve factor for the first truss is less than 1, while for the second truss it is much greater than 1. This therefore gives that $RF_1 < 1$ and $RF_2 > 1 + RF_{margin}$. According to the scheme presented above, a preference for the $A_1 + dA$ and $A_2 - dA$ choices should be added. This, in turn, means that in the QUBO matrix, the negative term $-\gamma$ can be added to the terms that correspond with these choices, namely the linear terms for qubits $q_{1,3}$ and $q_{2,1}$. This is shown in Eq. (4.20). The magnitude of this preference component, relative to all other terms in the QUBO matrix represents how strongly the RF constraint preference is enforced.

$$Q_{M^*+U+RF} = \begin{array}{c} \begin{matrix} q_{1,1} & q_{1,2} & q_{1,3} & q_{2,1} & q_{2,2} & q_{2,3} & \cdots \end{matrix} \\ \begin{bmatrix} M^*_{1,1} - \lambda & 2\lambda & 2\lambda & 0 & 0 & 0 & \cdots \\ 0 & M^*_{1,2} - \lambda & 2\lambda & 0 & 0 & 0 & \cdots \\ 0 & 0 & M^*_{1,3} - \lambda - \gamma & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & M^*_{2,1} - \lambda - \gamma & 2\lambda & 2\lambda & \cdots \\ 0 & 0 & 0 & 0 & M^*_{2,2} - \lambda & 2\lambda & \cdots \\ 0 & 0 & 0 & 0 & 0 & M^*_{2,3} - \lambda & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{matrix} q_{1,1} \\ q_{1,2} \\ q_{1,3} \\ q_{2,1} \\ q_{2,2} \\ q_{2,3} \\ \vdots \end{matrix} \tag{4.20}$$

In the previous example, it was seen that $RF_1 < 1$ and $RF_2 > 1 + RF_{margin}$, yielding the preference for the choices $A_1 + dA$ and $A_2 - dA$. This meant that a preference was added to the linear terms in the QUBO matrix that correspond to the qubits $q_{1,3}$ and $q_{2,1}$. However, these two qubits can communicate with each other through the off-diagonal quadratic coupling term. Thus, the constraint can be enforced more strongly if a preference is also added to the coupling term $q_{1,3}q_{2,1}$. This would then yield the QUBO matrix as shown in Eq. (4.21).

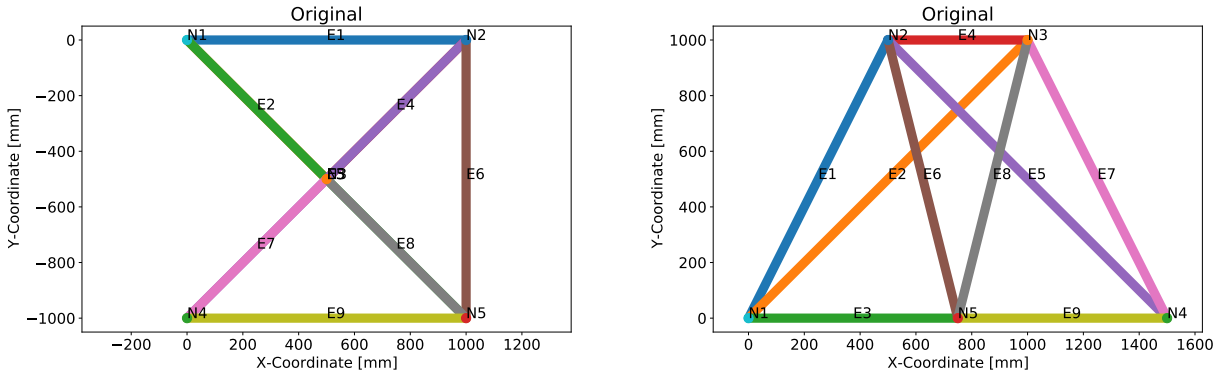
$$Q_{M^*+U+RF} = \begin{array}{c} \begin{matrix} q_{1,1} & q_{1,2} & q_{1,3} & q_{2,1} & q_{2,2} & q_{2,3} & \cdots \end{matrix} \\ \begin{bmatrix} M^*_{1,1} - \lambda & 2\lambda & 2\lambda & 0 & 0 & 0 & \cdots \\ 0 & M^*_{1,2} - \lambda & 2\lambda & 0 & 0 & 0 & \cdots \\ 0 & 0 & M^*_{1,3} - \lambda - \gamma & -\gamma & 0 & 0 & \cdots \\ 0 & 0 & 0 & M^*_{2,1} - \lambda - \gamma & 2\lambda & 2\lambda & \cdots \\ 0 & 0 & 0 & 0 & M^*_{2,2} - \lambda & 2\lambda & \cdots \\ 0 & 0 & 0 & 0 & 0 & M^*_{2,3} - \lambda & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{matrix} q_{1,1} \\ q_{1,2} \\ q_{1,3} \\ q_{2,1} \\ q_{2,2} \\ q_{2,3} \\ \vdots \end{matrix} \tag{4.21}$$

By adding these preference terms to the QUBO matrix, an RF constraint formulation is achieved. The formulation is not based on analytical relations from the finite-element analysis and depends on the results of only one finite-element analysis to fill in the QUBO matrix. By filling in the QUBO matrix, based on the rules for adding a preference term to certain qubit biases, the QA can find the optimal choice of truss cross-sectional areas that minimizes the objective function while preferring solutions that steer all truss RFs towards a value of 1. It is expected that iteratively solving the FEM problem, setting up the QUBO, and finding the optimal choice of cross-sectional areas, while adaptively controlling the allowed change in cross-sectional

area dA, a weight-optimal truss structure design can be found for which all RFs are near 1. Such an optimization routine is described in the next section.

4.3. Testing of Optimization Procedure

To test the procedures outlined in the previous section, sample problems are necessary. To this end, the truss-structures shown in Figs. 4.1a and 4.1b are defined.



(a) Box truss system.

(b) Bridge truss system

Figure 4.1: Sample truss optimization problems.

For both truss systems, the following points are relevant:

- Pinned support boundary conditions are applied to nodes N1 and N4.
- A vertical downward (↓) load of 20 kN is applied at node N5.
- The truss system is fully connected, with the exception of between boundary condition nodes. This means that every node is connected to every other node with a truss, but there is no truss in between the boundary condition nodes.
- The initial cross-sectional area for every truss is set to 2500 mm^2.
- The material properties assumed are for a fictitious material with a tensile limit stress of $\sigma_t = 10$ MPa and a compressive limit stress of $\sigma_c = 5$ MPa. The Young's modulus is set to a value of 200 GPa.

The procedure that is followed to optimize the truss structures is:

1. Set up the finite-element problem with a specific selection of cross-sectional areas.
2. Classically solve the finite-element problem to obtain the RFs for every truss.
3. Set up the QUBO for the optimization, allowing a change in cross-sectional area of dA.
4. Solve the QUBO problem to obtain the optimal selection of cross-sectional areas.
5. Set the optimal cross-sectional areas as the new starting point for a new iteration.
6. Repeat steps 1 through 5. If oscillating results are observed, i.e. when a truss area oscillates between smaller and larger choices, reduce the step size dA, until the desired accuracy is obtained.

The Python code that performs this procedure is available online [92]. The code allows for individual control of the dA variable on a per-truss basis. Furthermore, when many iterations of the procedure are performed, oscillating results and dA step size reduction are also automatically detected. The optimization process stops when sequential iterations produce identical results, or when the maximum allowed number of iterations has been reached. The maximum number of iterations can be reached, for example, when results oscillate at the smallest allowable scale of dA. A failsafe was built in to prevent cross-sectional areas from

reaching negative or zero values, by simply setting the area to 1 mm^2 whenever this situation arises. This was necessary to prevent singular matrix errors from occurring in the finite-element analysis.

To solve the QUBO problem itself, as represented by step 4 in the iterative scheme, two different options are available. The QUBO problem can be solved using either quantum annealing, or by using D-Wave's implementation of the classical simulated annealing (SA) method [19]. The choice between either method can be made simply by changing one Boolean variable (True/False) in the Python code. Throughout the development of the Python code, SA has proven to be a convenient classical solution method. The main benefit is that it allows for QUBO problems to be solved without relying on the quantum computing hardware. Therefore, SA will be used as a classical reference, against which the performance of the QA can be compared.

After the iterative optimization is complete, a classical post-processing procedure is also performed to simplify the result of the optimization. If the optimization procedure yields trusses with very small cross-sectional areas, such trusses are removed from the structure. Given that the starting point for each truss is to have an area of 2500 mm^2, the cutoff area is set at 50 mm^2, which is a factor 50 reduction in area compared to the initial configuration. Once the small trusses have been removed from the truss system, the post-processing also checks for overlapping trusses and whether any redundant nodes are present. In this context, redundant nodes are those nodes that only connect to two trusses, with those two trusses being exactly collinear. The redundant nodes are removed, with the two collinear trusses being simplified to a single truss. Finally, if any exactly overlapping trusses are present in the system, these trusses are merged into a single truss, having the summed cross-sectional area of its constituent parts. It will be seen that these post-processing steps lead to a drastically simplified version of the optimized Box truss system.

4.4. Results

In this section, the results will be shown that are produced for the two sample problems. In both cases, the variation in cross-sectional area dA initially starts at a value of 500 mm^2, but is sequentially allowed to change to 100 mm^2, 50 mm^2, and 10 mm^2. For both the unary constraint and the RF constraint, a strength value of 1 was used, giving the constraints the same order of magnitude as the mass objective function. Additionally, for the RF constraint, an RF margin of 0.1 was used. This means that the final optimal result will likely be found when the RFs of all trusses in the system are between 1 and 1.1. The Python codes used for these analyses and the raw results datasheet are available online [92].

4.4.1. Box Truss System

The initial design of the box truss system is shown in Fig. 4.2, with the exact cross-sectional areas and RFs as indicated in Table 4.1. The pinned boundary conditions act on nodes N1 and N4, with a downward vertical load of 20 kN acting in N5.

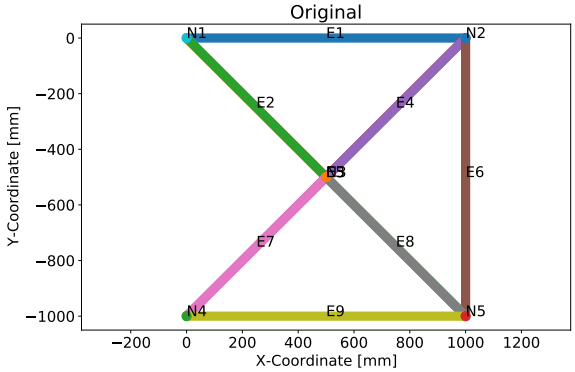


Figure 4.2: Box truss system initial design.

Element	Area [mm^2]	RF [-]
E1	2500	3.0176
E2	2500	3.0178
E3	2500	3.0176
E4	2500	2.1340
E5	2500	2.1340
E6	2500	3.0176
E7	2500	2.1339
E8	2500	3.0173
E9	2500	1.0670

Table 4.1: Table of truss cross-sectional areas and RFs.

Simulated Annealing

The truss system was optimized using 30 iterations of simulated annealing. The evolution of the cross-sectional areas and the RFs over every iteration are shown in the plots in Fig. 4.3. It can be seen that the cross-sectional area of certain trusses tends towards zero area, which causes significant oscillations in the

RFs for these trusses. Furthermore, note that some of the plotted lines may not be clearly visible due to exactly overlapping with the plots for other elements.

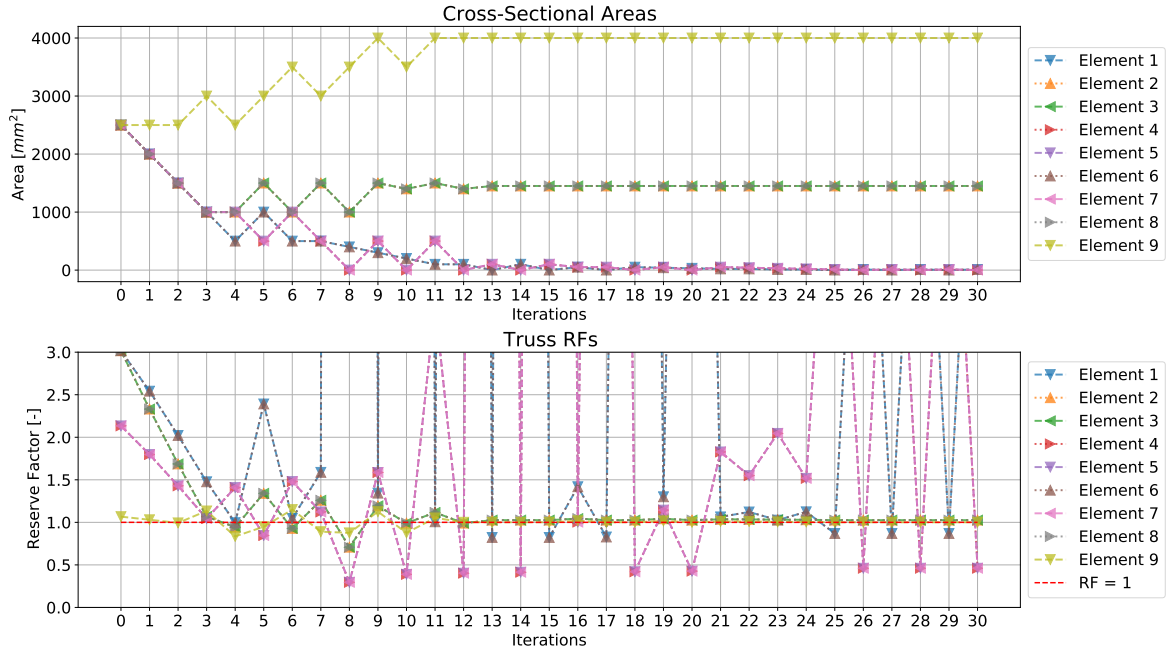


Figure 4.3: Evolution of the cross-sectional area and the RFs for the box truss structure using SA.

The final iteration of the SA optimization routine yielded the cross-sectional areas and RFs as shown in Table 4.2, graphically shown in Fig. 4.4.

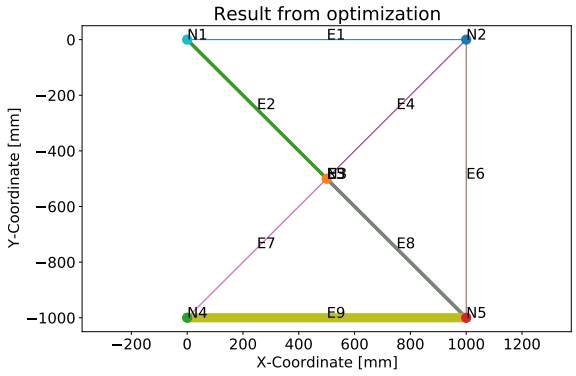


Figure 4.4: Box structure optimization result based on SA.

Element	Area [mm^2]	RF [-]
E1	11	7.1626
E2	1450	1.0261
E3	1450	1.0260
E4	1	0.4609
E5	1	0.4609
E6	11	7.1690
E7	1	0.4609
E8	1450	1.0260
E9	4000	1.0011

Table 4.2: Results from optimization of box truss system using SA.

However, as can be seen, several trusses have become negligible in size compared to the others. Thus, some classical post-processing can be performed to simplify the structure. As a simplification, all trusses that have been given a cross-sectional area of less than 50 mm^2 are removed. Furthermore, for this structure there are some overlapping trusses that can be merged, taking care to sum the cross-sectional areas of the merged trusses. This yields the simplified structure with the cross-sectional areas and RFs as shown in Table 4.3, and visible in Fig. 4.5.

This entire analysis procedure was performed three times using SA and produced identical final results each time. On average the analysis took 36.511 seconds, with a standard deviation of 0.344 seconds. After post-processing, the final result is a simplified truss-structure design for which cross-sectional areas have been found that minimize the weight of the structure and bring the reserve factors close to a value of 1.

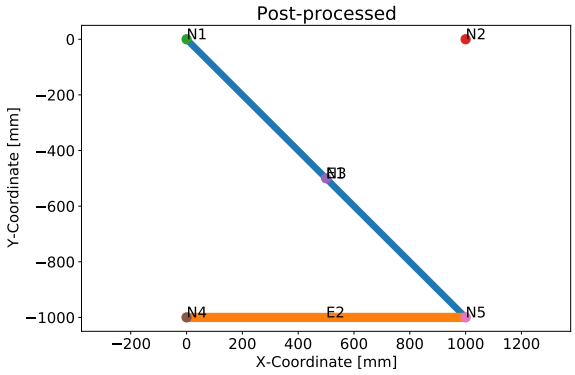


Figure 4.5: Post-processed box structure optimization result based on SA.

Element	Area [mm^2]	RF [-]
E1	2900	1.0252
E2	4000	1.0003

Table 4.3: Post-processed optimized box truss system cross-sectional areas and RFs.

Quantum Annealing

Using the SA analysis as a reference, the problem can also be submitted to the QA to investigate whether similar results are obtained, and compare the amount of time necessary to complete the analysis. Starting from the same initial structure, again 30 iterations of the optimization routine are performed. For the QA a setting of 16 reads per problem submission is used. This means that each time a QUBO problem is submitted, it is solved 16 times, and the lowest energy solution that has been found will be the one used for the next iteration of the analysis. For each read, the default annealing time of 20 microseconds is used.

Performing the full optimization procedure three times, it was seen that each time the same final result is obtained. The evolution of the cross-sectional areas and the RFs throughout the analysis are shown in Fig. 4.6. Note that since some of the cross-sectional areas tend towards zero, this causes large fluctuations to occur in the RFs for those trusses. The cross-sectional areas and corresponding RFs of the final iteration are shown in Table 4.4, and are shown graphically in Fig. 4.7.

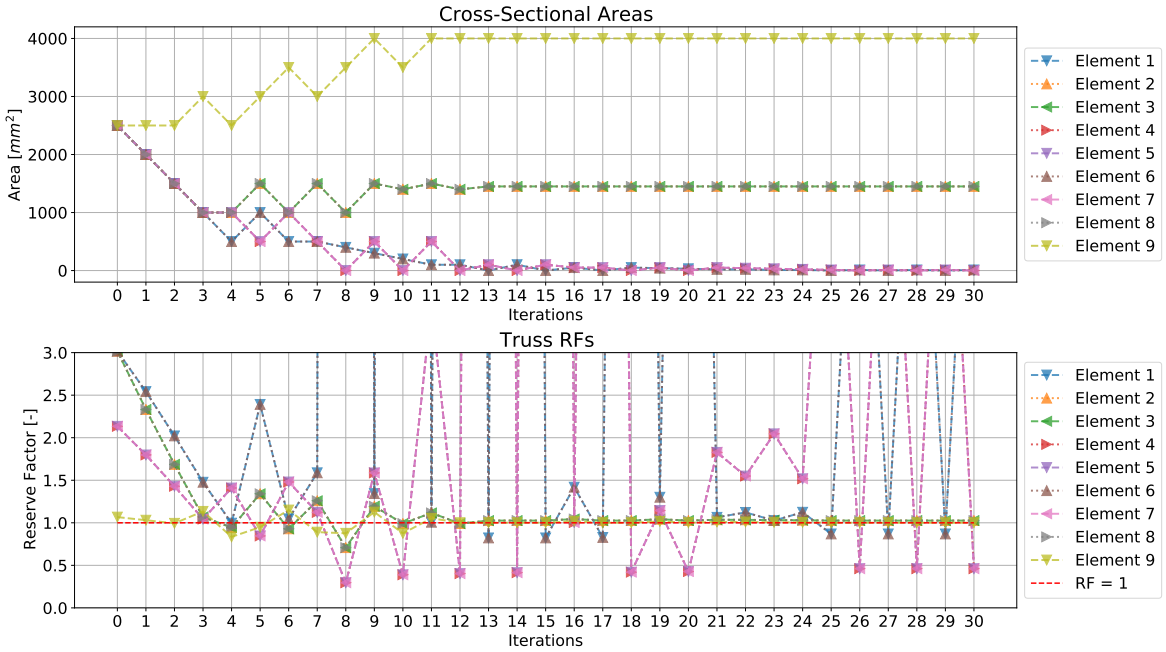


Figure 4.6: Evolution of the cross-sectional area and the RFs for the box truss structure using quantum annealing.

After post-processing, by removing trusses with a cross-sectional area smaller than 50 mm^2, and merging overlapping trusses, the final optimized structure is obtained as shown in Fig. 4.8, with the cross-sectional areas and RFs as specified in Table 4.5. The total amount of time needed to achieve these final results was,

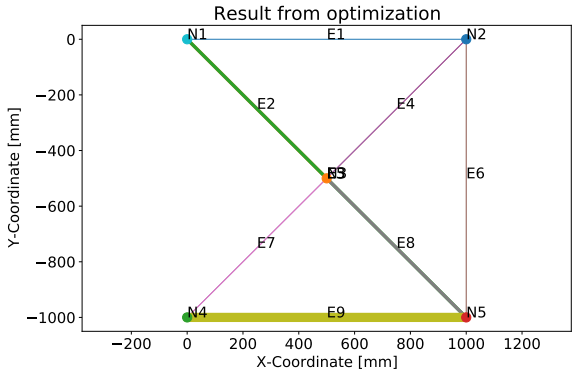


Figure 4.7: Box structure optimization result based on quantum annealing.

Element	Area [mm^2]	RF [-]
E1	11	7.1626
E2	1450	1.0261
E3	1450	1.0260
E4	1	0.4609
E5	1	0.4609
E6	11	7.1690
E7	1	0.4609
E8	1450	1.0260
E9	4000	1.0011

Table 4.4: Results from optimization of box truss system using quantum annealing.

on average, 138.469 seconds, with a standard deviation of 0.264 seconds. However, this amount of time is dominated by various sources of overhead, such as the programming methods used, as well as network delay in submitting the problem to D-Wave, potential queue time before the problem is solved, and network delay for returning the result to the local computer. Detailed information on the timing of the processes involved in submitting problems to the D-Wave QA is provided in D-Wave's timing user manual [28]. The total amount of actual QPU access time needed to perform all iterations was only 0.4382 seconds on average, with a standard deviation of $8.387 \cdot 10^{-5}$ seconds.

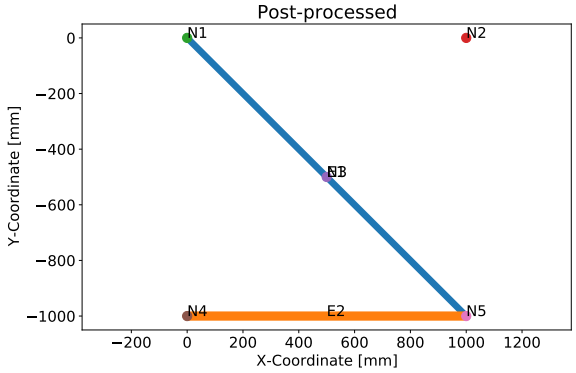


Figure 4.8: Post-processed box structure optimization result based on quantum annealing.

Element	Area [mm^2]	RF [-]
E1	2900	1.0252
E2	4000	1.0003

Table 4.5: Post-processed optimized box truss system cross-sectional areas and RFs.

Box Truss System Discussion

It is observed that the final results from the quantum annealing analyses are the exact same as those from the simulated annealing analyses. Furthermore, the methods produced the same results every time the analysis was performed. This indicates that both methods are capable of consistently solving the problem and finding optimal solutions. The only difference between the methods is that the total amount of time needed to solve the optimization problem through quantum annealing was much longer than through SA. However, neglecting the overhead associated with quantum annealing, the actual amount of QPU access time needed to solve the problem is extremely short. If the various sources of overhead could be reduced, for example by having unlimited access to local quantum annealing hardware, the method could become quite competitive to use in practice.

4.4.2. Bridge Truss System

For the bridge truss system a similar approach is taken to that used for the box truss system. First, the structure will be analyzed three times using simulated annealing, to obtain classical reference solutions. Then, an additional three analyses are performed using quantum annealing, and the results from both methods will be compared. The original bridge truss system design and the initial cross-sectional areas and RFs are shown in Fig. 4.9 and Table 4.6. Note that the pinned boundary conditions act in the nodes N1 and N4, with a 20 kN downward vertical load acting in node N5.

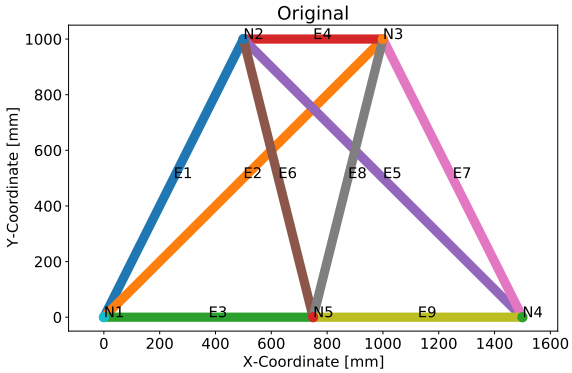


Figure 4.9: Initial design of bridge truss structure.

Element	Area [mm^2]	RF [-]
E1	2500	1.7216
E2	2500	2.5211
E3	2500	33632.6804
E4	2500	5.5775
E5	2500	2.5211
E6	2500	2.4254
E7	2500	1.7216
E8	2500	2.4254
E9	2500	33632.6804

Table 4.6: Bridge truss system cross-sectional areas and RFs.

Simulated Annealing

Although a maximum number of 30 iterations was set for the bridge truss system analysis, the optimization consistently converged before reaching the maximum number of iterations. In this case, the analysis is deemed to have converged because the optimization routine finds the same exact solution twice in a row. Each of the three analyses resulted in the exact same final solution, reaching this solution after 11 iterations. The evolution of the truss cross-sectional areas throughout these iterations, as well as the truss RFs are shown in Fig. 4.10. The final iteration produced the results as shown in Fig. 4.11 and Table 4.7.

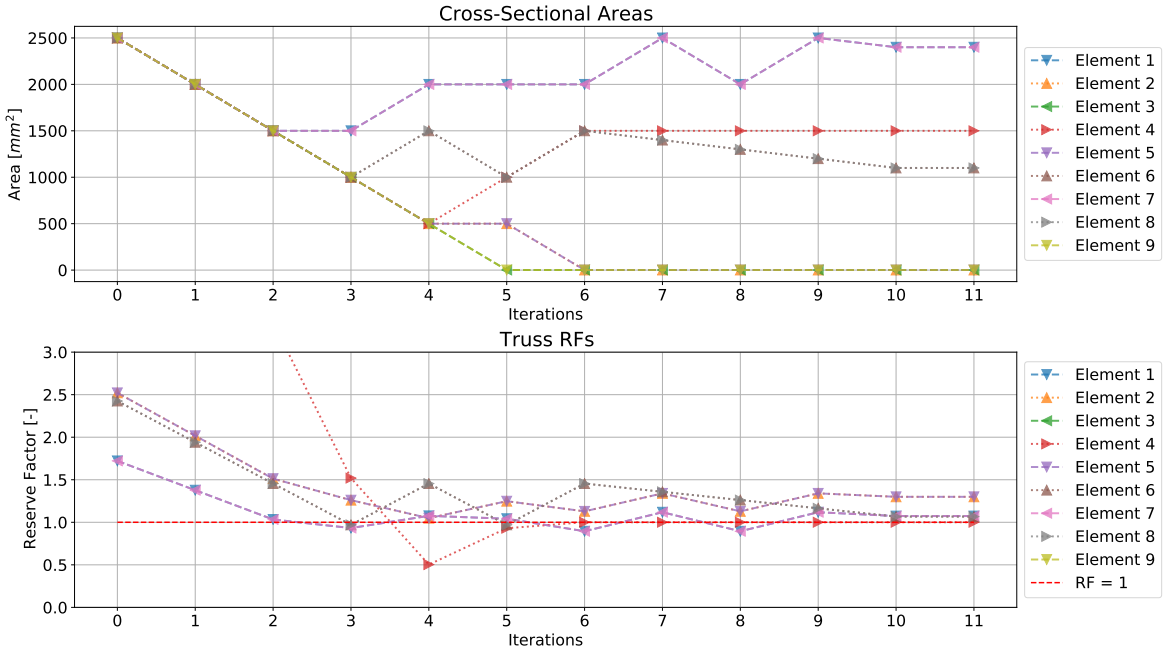


Figure 4.10: Evolution of the cross-sectional area and the RFs for the bridge truss structure using SA.

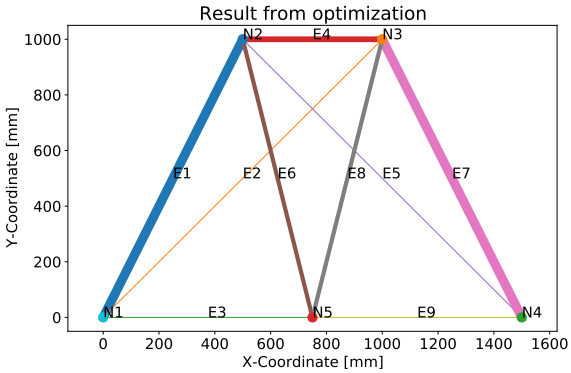


Figure 4.11: Final result from SA optimization of bridge truss structure.

Element	Area [mm^2]	RF [-]
E1	2400	1.0736
E2	1	1.2995
E3	1	8053.3562
E4	1500	1.0005
E5	1	1.2995
E6	1100	1.0672
E7	2400	1.0736
E8	1100	1.0672
E9	1	8053.3562

Table 4.7: Truss cross-sectional areas and RFs from SA optimization of bridge truss structure.

The solution was post-processed to remove trusses with a final cross-sectional area smaller than 50 mm^2. This then resulted in the final simplified design as shown in Fig. 4.12, with the cross-sectional areas and RFs as shown in Table 4.8.

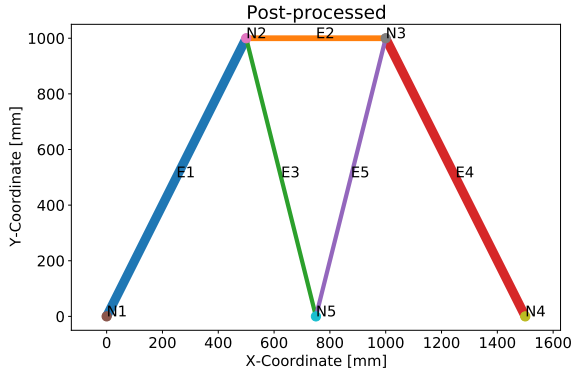


Figure 4.12: Post-processed result from SA optimization of bridge truss structure.

Element	Area [mm^2]	RF [-]
E1	2400	1.0733
E2	1500	1.0000
E3	1100	1.0672
E4	2400	1.0733
E5	1100	1.0672

Table 4.8: Post-processed truss cross-sectional areas and RFs for bridge truss structure.

Overall, the three SA analyses that were performed for the bridge truss system lasted 13.999 seconds on average, with a standard deviation of 0.465 seconds.

Quantum Annealing

Similar to the SA analyses, the bridge truss system is also analyzed three times using quantum annealing. The results can then be compared to those found using SA. It was found that by using quantum annealing, with a setting of 16 reads per iteration and a default annealing time of 20 microseconds, the same final results were obtained for each of the three analyses. The analyses each converged after 11 iterations. The evolution of the truss cross-sectional areas and the RFs throughout these iterations are shown in Fig. 4.13. The final selection of cross-sectional areas and the corresponding RFs are shown in Table 4.9, with the graphical representation of the bridge truss system shown in Fig. 4.14.

Post-processing of the result from the quantum annealing optimization is performed by removing the trusses that have small cross-sectional areas with an area of less than 50 mm^2. This yielded the simplified solution as shown in Fig. 4.15, with the cross-sectional areas and corresponding RFs detailed in Table 4.10.

The quantum annealing analyses took on average 51.475 seconds with a standard deviation of 0.184 seconds. However, especially for quantum annealing, this amount of time is dominated by various sources of overhead. The sum of the QPU access time needed to perform all iterations for each of the analyses was on average a mere 0.161 seconds, with a standard deviation of $8.055 \cdot 10^{-6}$ seconds.

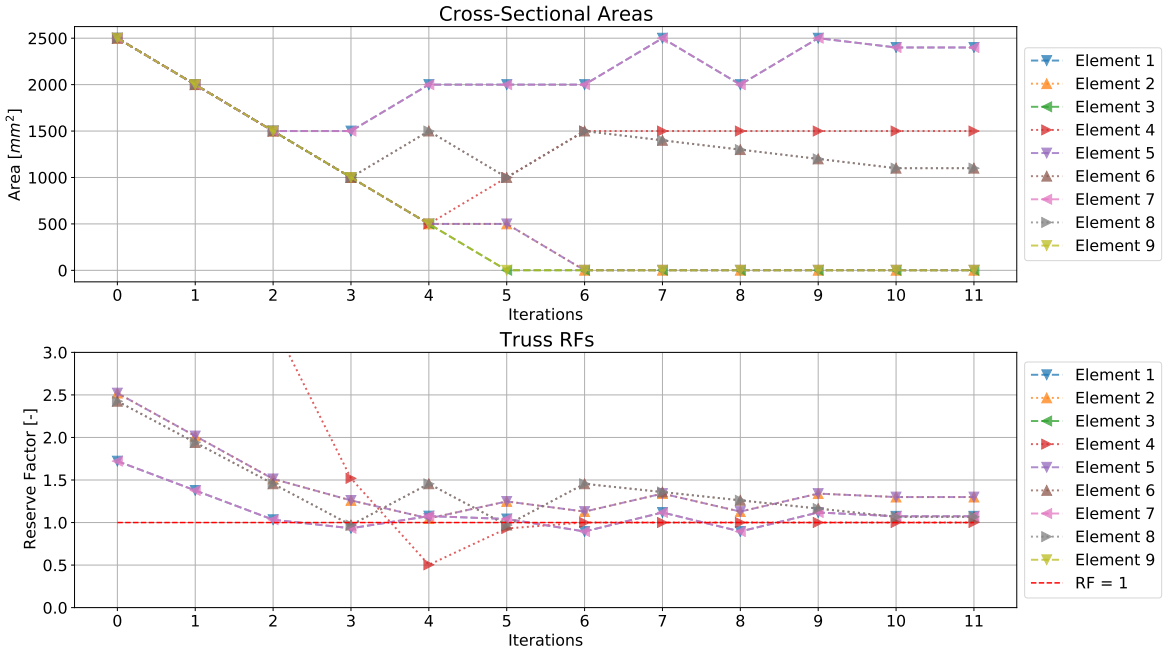


Figure 4.13: Evolution of the cross-sectional area and the RFs for the bridge truss structure using quantum annealing.

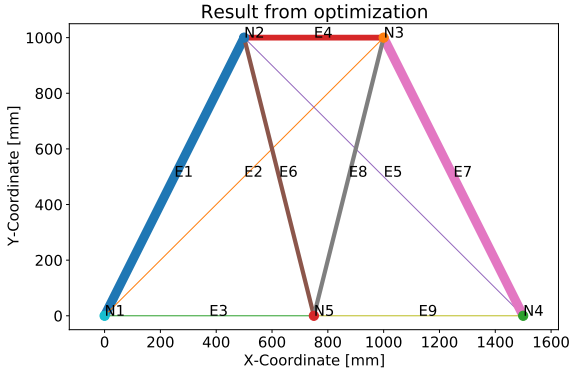


Figure 4.14: Final result from quantum annealing optimization of bridge truss structure.

Element	Area [mm^2]	RF [-]
E1	2400	1.0736
E2	1	1.2995
E3	1	8053.3562
E4	1500	1.0005
E5	1	1.2995
E6	1100	1.0672
E7	2400	1.0736
E8	1100	1.0672
E9	1	8053.3562

Table 4.9: Truss cross-sectional areas and RFs from quantum annealing optimization of bridge truss structure.

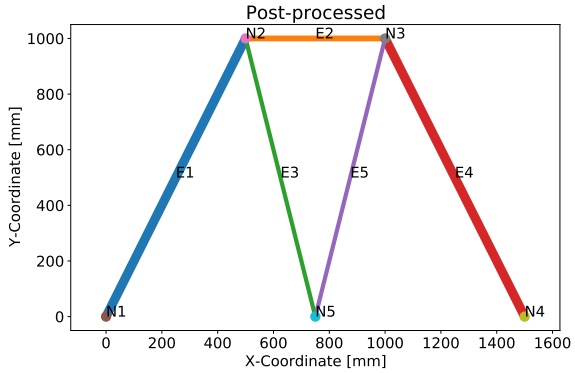


Figure 4.15: Post-processed result from quantum annealing optimization of bridge truss structure.

Element	Area [mm^2]	RF [-]
E1	2400	1.0733
E2	1500	1.0000
E3	1100	1.0672
E4	2400	1.0733
E5	1100	1.0672

Table 4.10: Post-processed truss cross-sectional areas and RFs for bridge truss structure.

Bridge Truss System Discussion

From the results that were obtained for the bridge truss system it is evident that using simulated annealing and quantum annealing both lead to the exact same final solution each time the problem is solved. This means that both methods are able to reliably produce optimal results without making errors. Because of this, it also indicates that it is likely feasible to solve larger problems than the ones currently tested. Note that, since a purely vertical load was applied, the horizontal members connected to the loaded point were essentially zero-force members. This is why the optimization reduced these members to having negligible cross-sectional areas. Realistically, if the truss joints were allowed to pivot, the optimized and post-processed result would be unstable, since any amount of horizontal load would cause the structure to collapse.

Naturally, the two solving methods that were applied differ in the amount of time necessary to find the final solution. Using SA the final solution was found in about 14 seconds on average, while for quantum annealing it took an average of 51.5 seconds. In the case of quantum annealing, the actual average amount of QPU access time needed was only 0.161 seconds, meaning that the total of 51.5 seconds consists of almost exclusively overhead. Therefore, with unfettered access to QPU hardware, quantum annealing may be an extremely fast way to solve this type of optimization problem.

Despite the seemingly positive results, throughout the testing process, a feeling of uncertainty about the methods presented in this chapter arose. The optimization method, as shown, can provide optimal truss structure designs after performing several iterations. However, one might critically ask: *'What is the quantum annealer really adding to this process?'*. In the upcoming section this question is addressed.

4.5. Triviality of the QUBO Formulation and Classical Reproduction

Of the work that was done to formulate the truss system optimization QUBO, as presented in this chapter, the most time and effort was spent finding a way to add a stress constraint to the QUBO problem. This was necessary, as the goal for the optimization is to find the most lightweight design, while still complying with basic limit stress constraints. At the time of development, this was very much a pen-and-paper process, relying on a natural understanding of the workings of QUBO problems, and being the topic of many discussions with thesis supervisor Dr. Chen. However, despite the effort, the stress constraint as it is currently being applied appears to result in a trivial optimization problem.

Looking at how the optimization process is performed through every iteration shows that the QA always obeys the 'preference' that is assigned as part of the stress constraint. This preference is determined through a single classical FEM analysis of the current selection of cross-sectional areas. Then, based on the RFs, a preference is assigned for how the truss cross-sectional areas should be changed. For example, if the current cross-sectional area of a specific truss yields an $RF < 1$, then a preference is assigned to choose an increased cross-sectional area in the next iteration. However, if the QA always exclusively obeys this preference, then it is trivial to predict what the outcome of the optimization will be for every iteration. By knowing exactly what the truss RFs are for the current selection of cross-sectional areas, the selection of new cross-sectional areas for the next iteration will simply follow from the rules that govern the setting of the preference. As such, the QA does not really *do* anything interesting. It is simply directly following the assigned preferences, and returning an answer that could be guessed without going through the effort of setting up a QUBO matrix and solving the problem. In essence, the method used for constraining the truss RFs yields a completely trivial problem, with the optimization routine as a whole simply describing a basic decision-making strategy.

The above statements can be supported experimentally. A completely classical decision-making scheme can be set up, relying solely on the truss RFs to make a selection of new cross-sectional areas. The process emulates the behavior of the optimization that was done through the QA, but completely bypasses the need to set up and solve a QUBO problem. The code for this purely classical decision-making scheme is also available online [92]. Allowing this decision-making routine to run 30 iterations of both the box truss system and the bridge truss system yields the results shown in Figs. 4.16 and 4.17 and Tables 4.11 and 4.12.

The classical decision-making scheme leads to the exact same solutions as those achieved through quantum annealing and simulated annealing. However, in this case, the final results were obtained much more quickly. Based on the results from three separate runs, the box truss system took on average 0.3305 seconds to optimize, with a standard deviation of 0.0245 seconds. For the bridge truss system, the converged solution was found in 0.2441 seconds on average, with a standard deviation of only 0.0033 seconds. This is at least an order of magnitude faster than using either simulated annealing or quantum annealing.

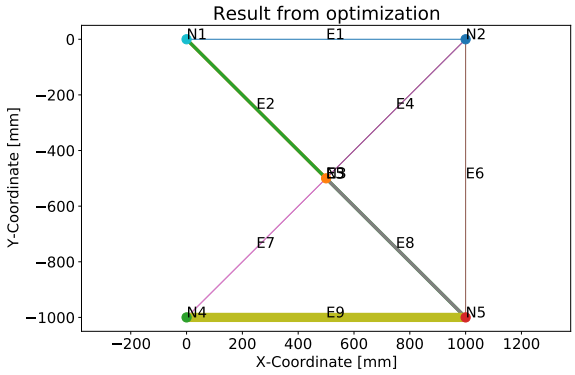


Figure 4.16: Raw final result for box truss system from classical decision-making strategy.

Element	Area [mm^2]	RF [-]
E1	11	7.1626
E2	1450	1.0261
E3	1450	1.0260
E4	1	0.4609
E5	1	0.4609
E6	11	7.1690
E7	1	0.4609
E8	1450	1.0260
E9	4000	1.0011

Table 4.11: Final areas and RFs for box truss system from classical decision-making strategy.

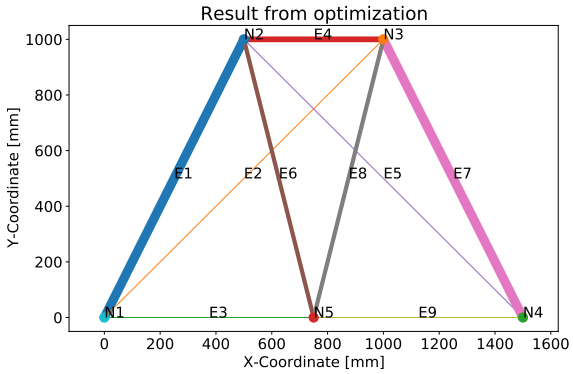


Figure 4.17: Raw final result for bridge truss system from classical decision-making strategy.

Element	Area [mm^2]	RF [-]
E1	2400	1.0736
E2	1	1.2995
E3	1	8053.3562
E4	1500	1.0005
E5	1	1.2995
E6	1100	1.0672
E7	2400	1.0736
E8	1100	1.0672
E9	1	8053.3562

Table 4.12: Final areas and RFs for bridge truss system from classical decision-making strategy.

Because the results can be emulated exactly, while completely bypassing any of the quantum computing-related aspects, this means that the quantum annealing technology is not adding any value to the optimization method presented in this chapter. It only serves as a medium to execute a simple decision-making scheme, and thus the method is considered trivial.

At this point then, two options exist: either the methods in this chapter are changed, extended, or adapted so that they are no longer trivial, or a completely different method must be created for setting up truss-optimization problems for use with the QA. Possible extensions are discussed in Section 4.6, while a completely different approach to setting up the truss optimization problem is discussed in Chapter 5.

4.6. Possible Extensions

In this section some of the attempted or conceptualized extensions to the truss optimization method discussed in this chapter are commented on.

Tuning of Constraint Strength For the testing of the current stress constraint method, a constraint strength value of 1 was used. It was seen that this caused the stress constraint to always be obeyed, which is what led to the conclusion that the method produced trivial results. A method of reducing the triviality of the results might be to reduce the strength of the stress constraints to such a degree, that the constraint is no longer exclusively obeyed. In turn, this would mean that the mass objective function becomes relatively more important. This may lead to situations where the QA prefers to save weight versus picking cross-sections that comply with strength requirements.

Arguably, reducing the strength value of the stress constraints would make the output of the QA less predictable, and therefore probably less trivial. However, it is questionable whether this tactic would lead to

truss system designs that comply with stress requirements. This is why the constraint strength was kept at a value high enough to enforce a hard stress constraint. Thus, the possibility that lower constraint strength would lead to less trivial results was not investigated in depth. This could be investigated without needing to otherwise change or extend any of the currently implemented methods.

Inclusion of Secondary Effects One of the ideas to reduce the triviality that the stress constraint causes was to find a way to include secondary stress effects in the QUBO. What is meant by this is clarified by asking: *How does the stress in truss A change, when the cross-sectional area of truss B is altered?*. More mathematically, the partial derivatives of the truss stress with respect to the cross-sectional area of every truss were investigated. The primary stress derivative would then be the derivative of the stress in truss A with respect to the area of truss A. The secondary stress derivatives are then the derivative of the stress in truss A, with respect to the cross-sectional area of all other trusses in the system.

Using the software Maple, a simple truss structure was investigated. By leaving the truss cross-sectional areas as symbolic variables, and then solving the FEM problem, symbolic expressions for the truss stresses and RFs could be generated. Having these expressions allowed for the program to then calculate the partial derivatives with respect to the individual truss cross-sectional areas. However, investigating the magnitudes of these derivatives, it was seen that the secondary stress derivatives were much smaller than the primary derivatives. When evaluating all symbolic derivatives, using a cross-sectional area of 2500 mm^2 for all trusses, it was seen that the secondary stress derivatives are approximately 1000 times less significant than the primary derivatives. From this observation it was concluded that it would not be particularly useful to integrate the secondary stress effects into the truss optimization QUBO. The amount of additional information needed to include the secondary effects would also likely make this quite a challenging task to integrate into the setup of the QUBO problem. The Maple worksheet, as well as the Abaqus FEM model that was used to verify the results are available online [92].

Discrete Derivatives An idea for a possible extension for the truss system optimization QUBO is to do more than just one FEM analysis before setting up the QUBO matrix. For example, imagine a system with N number of trusses, with every truss having a specified initial choice of cross-sectional area. It would be possible to perform a FEM analysis on the initial configuration to get the reference RFs for every truss. Then, an additional N number of FEM analyses could be performed, where for each analysis one of the cross-sectional areas is slightly changed. This would then yield a new set of RFs, which can be used in conjunction with the initial reference RFs to determine the discrete derivatives of the RFs due to changing the cross-sectional area. To get all discrete derivatives, a total of $N+1$ FEM analyses would therefore need to be performed.

The improvement in this case again lies in the attempted inclusion of secondary effects, but as was previously discussed, this is likely not very interesting, as these effects are very small. Furthermore, by finding the discrete derivatives, as opposed to finding derivatives symbolically, many more FEM analyses need to be performed. This is fast for small problems, however, becomes much more time-consuming for larger problems. The fact that $N+1$ FEM analyses would be needed to calculate all discrete derivatives means even more so that this idea to include secondary effects would be an inefficient process. The idea was not further investigated.

Inclusion of Stiffness Constraints One last idea to reduce the triviality caused by the stress constraint in its current form is to add an additional constraint on the stiffness of the truss system. An investigation into the truss system stiffness matrix could reveal properties that would help with identifying optimal load paths, or finding the truss elements that contribute most to the deflection of the loaded point. Knowing such properties may in turn lead to an extension of the QUBO setup that includes stiffness constraints, which reduces the triviality of the method compared to only using a stress constraint.

The possibility for a stiffness constraint was briefly investigated. The method relied on the addition of a 'preference' term to the QUBO matrix, similar to that used in the stress constraint method. In this case, the preference was added to increase the stiffness (choose a larger cross-sectional area), for the trusses that directly contribute the most stiffness to the loaded point, in the direction of the load. However, since the manual addition of preference terms to the QUBO matrix was eventually found to lead to trivial optimization problems, it was also deemed to be an unsuitable method for stiffness constraints. Ultimately, these ideas for reducing the triviality of the direct QUBO method for the truss sizing optimization problem were abandoned in favor of setting up an entirely new methodology, as presented in Chapter 5.

5

Truss Sizing Optimization: Symbolic Finite-Element Method

Another method of setting up the truss sizing optimization QUBO is by solving the finite-element problem fully in terms of qubit variables. This may allow for a minimization objective function to be found that describes the sizing optimization problem from a more analytical perspective. When minimized this objective function should also lead to the ideal choice of truss cross-sections. This symbolic method of setting up the truss sizing optimization problem is described in this chapter. The work is divided into three phases: a preparation phase, a QUBO setup phase, and a QUBO solving phase. Throughout the process, various challenges were encountered that needed to be overcome to eventually yield a suitable QUBO formulation for the truss sizing optimization problem.

5.1. Phase 1: Preparation

In this section, the basic definitions that are needed for setting up the QUBO problem are given, as well as an overview of the general plan that will be followed to set up the QUBO formulation of the truss sizing optimization problem. The first challenge that was encountered, being the inversion of symbolically defined matrices, is also part of the preparatory work.

5.1.1. QUBO Basics and Plan

Since the goal of the optimization is to make the ideal selection of truss cross-sectional areas, the cross-sectional areas are defined in terms of binary variables. This is done in a similar way as for the mass objective function in Chapter 4. Thus, the cross-sectional area of truss n, given a set of C possible discrete choices, is written as:

$$A_n = \sum_{c=1}^{C} q_{n,c} A_{n,c} \tag{5.1}$$

for which $A_{n,c}$ are in the set of cross-sectional areas that can be selected for truss n, and $q_{n,c}$ are the corresponding binary qubit variables.

The main idea behind the symbolic finite-element method of formulating a sizing optimization QUBO is to utilize Eq. (5.1) for the cross-sectional area of every truss, and set up and solve the linear FEM problem entirely symbolically. If the binary qubit variables can be kept as unknowns throughout the solving procedure, symbolic expressions for the truss stresses and reserve factors can hopefully be set up and used to formulate a QUBO problem. By using these expressions, the QA can then potentially be used to make the optimal selection of cross-sectional areas such that every truss in the system is as close as possible to the limit stress, giving a reserve factor as close as possible to 1. This also implies that the minimum-weight configuration has been found, since further reducing cross-sectional areas to save weight would lead to non-compliance with material strength requirements. Thus, the focus will be on using the symbolic expressions for truss stresses and reserve factors to obtain an objective function that can be used for the QUBO problem.

From this basic idea, the symbolic problem-solving process can be described by the following steps:

1. Given sets of possible choices of cross-sectional area for every truss, the actual cross-sectional area can be written in terms of qubit variables, as shown in Eq. (5.1).

2. Using the expression for the truss cross-sectional area, the element stiffness matrices of the trusses can also be written in terms of the qubit variables.
3. The symbolic global stiffness matrix of the entire system of trusses can be assembled from each of the element stiffness matrices.
4. Proceeding as normal with the FEM analysis, boundary conditions must be taken into account, and a vector of applied loads must be known. By inverting the symbolic global stiffness matrix, and multiplying this inverse matrix with the load vector, a symbolic vector of nodal displacements can be obtained.
5. Using the symbolic vector of nodal displacements, and the known initial length of every truss, symbolic expressions for the truss strain can be set up.
6. By multiplying the symbolic expression of the truss strain with the Young's modulus, a symbolic expression for the truss stress is obtained.
7. Lastly, a symbolic expression for the truss RF can be found by dividing the assumed limit stress by the symbolic truss stress expression.

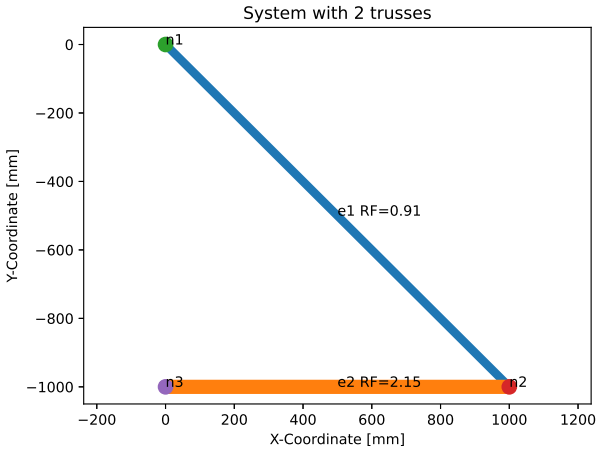
Once symbolic expressions for the truss reserve factors have been obtained, these expressions may be used to construct an objective function for which the minimum solution encodes the optimal choice of cross-sectional area for every truss in the structure. For such an objective function to be usable with the D-Wave QA, it must be compatible with the QUBO problem formulation. Thus, the objective function should be written as a pure sum of linear and quadratic qubit terms. However, the outcome of the aforementioned symbolic problem-solving steps may not be in exactly this format. As such, once symbolic expressions for the reserve factors for every truss have been found, these expressions may need to be further manipulated to produce an objective function that fits exactly within the QUBO problem framework. This process was thoroughly investigated using three small sample problems.

5.1.2. Sample Problems

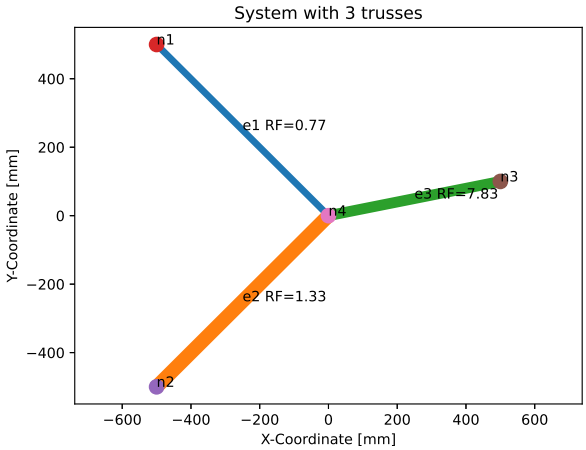
The sample problems used to investigate the symbolic QUBO method are shown in Figs. 5.1a to 5.1c. As can be seen, the simple truss structures consist of two, three, and four truss elements, forming problems that are expected to be incrementally more challenging to solve. For these sample problems it is assumed that there are three possible choices of cross-sectional area for every truss. This means that the number of binary qubit variables needed to define the problems is three times the number of trusses in the problem, giving the two-, three-, and four-truss problems a total of 6, 9, and 12 qubit variables.

The exact definitions for the three sample problems are given in Table 5.1, including the boundary conditions and loads acting on the truss systems. The sign convention has the positive x-direction pointing horizontally to the right ($\rightarrow$) and has the positive y-direction pointing vertically upwards ($\uparrow$). Having the necessary design parameters for the truss systems, the allowable choices for the truss cross-sectional areas are also defined. These are given in Table 5.2. For reference, when choosing the mid-sized (option 2) cross-sectional area for every truss, each of the trusses will have the reserve factors as indicated in Table 5.3. These reserve factors are also indicated in Figs. 5.1a to 5.1c.

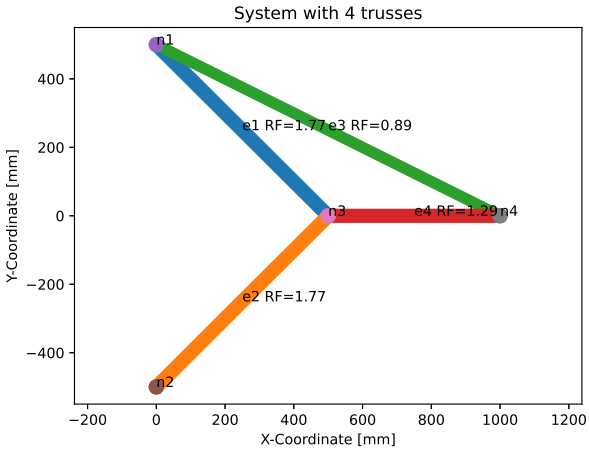
The following sections describe the implementation, and the challenges encountered during the investigation into these sample problems, in the attempt at setting up the truss sizing optimization QUBO problem.



(a) Two-truss problem.



(b) Three-truss problem.



(c) Four-truss problem.

Figure 5.1: Sample truss optimization problems.

	Two-Truss System			Three-Truss System			Four-Truss System		
Material	Young's Modulus	200000	MPa	Young's Modulus	200000	MPa	Young's Modulus	200000	MPa
	Tens. Limit	100	MPa	Tens. Limit	100	MPa	Tens. Limit	100	MPa
	Compr. Limit	100	MPa	Compr. Limit	100	MPa	Compr. Limit	100	MPa
Nodes	Name	X [mm]	Y [mm]	Name	X [mm]	Y [mm]	Name	X [mm]	Y [mm]
	N1	0	0	N1	-500	500	N1	0	500
	N2	1000	-1000	N2	-500	-500	N2	0	-500
	N3	0	-1000	N3	500	100	N3	500	0
	-	-	-	N4	0	0	N4	1000	0
Elements	Name	Start Node	End Node	Name	Start Node	End Node	Name	Start Node	End Node
	E1	N1	N2	E1	N1	N4	E1	N1	N3
	E2	N2	N3	E2	N2	N4	E2	N2	N3
	-	-	-	E3	N3	N4	E3	N1	N4
	-	-	-	-	-	-	E4	N3	N4
Load	Acting on:	Fx [kN]	Fy [kN]	Acting on:	Fx [kN]	Fy [kN]	Acting on:	Fx [kN]	Fy [kN]
	N2	0	-70	N4	0	-100	N4	0	-100
BCs	Acting on:	dx [mm]	dy [mm]	Acting on:	dx [mm]	dy [mm]	Acting on:	dx [mm]	dy [mm]
	N1	0	0	N1	0	0	N1	0	0
	N3	0	0	N2	0	0	N2	0	0
	-	-	-	N3	0	0	-	-	-

Table 5.1: Truss system material, design, and load parameters.

Two-Truss System			
For element:	Option 1 [mm^2]	Option 2 [mm^2]	Option 3 [mm^2]
E1	800	900	1000
E2	1400	1500	1600
Three-Truss System			
For element:	Option 1 [mm^2]	Option 2 [mm^2]	Option 3 [mm^2]
E1	400	500	600
E2	950	1050	1150
E3	700	800	900
Four-Truss System			
For element:	Option 1 [mm^2]	Option 2 [mm^2]	Option 3 [mm^2]
E1	2400	2500	2600
E2	2400	2500	2600
E3	1900	2000	2100
E4	2400	2500	2600

Table 5.2: Truss system allowable choice of cross-sectional areas.

Two-Truss System			Three-Truss System			Four-Truss System		
For element:	Area [mm^2]	RF	For element:	Area [mm^2]	RF	For element:	Area [mm^2]	RF
E1	900	0.9086	E1	500	0.7692	E1	2500	1.7680
E2	1500	2.1511	E2	1050	1.3252	E2	2500	1.7680
-	-	-	E3	800	7.8331	E3	2000	0.8910
-	-	-	-	-	-	E4	2500	1.2859

Table 5.3: Truss system reference reserve factors.

5.1.3. Challenge: Symbolic Matrix Inversion and Setup of Symbolic Expressions

Using Python, a script was written that can set up a truss finite-element problem symbolically, assuming known discrete choices for the cross-sectional area of every truss. However, it became immediately apparent that it is quite difficult to 'solve' the finite-element problem while using a global stiffness matrix that contains many unknown variables. Even for these small problems, it became intractable to invert the global stiffness matrix for the four-truss problem. Thus, step 4 from the approach described above, could not be performed. The solving procedure stalls when trying to find the inverse of the symbolic global stiffness matrix.

However, the Python package that was used for manipulating symbolic expressions, Sympy, might be inefficient compared to other software. Thus, as an alternative, the Matlab software package was used with the hope that its implementation for solving symbolic matrix equations might be more efficient. Within Matlab, matrix equations of the familiar $[\mathbf{A}]\,\mathbf{x} = \mathbf{b}$ form can be solved for the vector of unknowns $\mathbf{x}$ by utilizing the so-called backslash operator. As such, the vector of unknowns can be found by implementing $\mathbf{x} = [\mathbf{A}]\,\backslash\mathbf{b}$. This circumvents the need to explicitly calculate the inverse of the matrix $[\mathbf{A}]$, being $[\mathbf{A}]^{-1}$, and means that the vector of unknowns does not have to be calculated by $\mathbf{x} = [\mathbf{A}]^{-1}\,\mathbf{b}$.

Thus, to attempt to solve the symbolic finite-element problem, the symbolic global stiffness matrix was implemented into the Matlab software package. Knowing the symbolic global stiffness matrix $[\mathbf{K}]$ and the nodal force vector $\mathbf{f}$, the vector of nodal displacements $\mathbf{u}$ could be found by implementing $\mathbf{u} = [\mathbf{K}]\,\backslash\mathbf{f}$. It was found that indeed the Matlab implementation for solving linear systems of equations was much more efficient and was able to find symbolic expressions for the nodal displacement vector.

At the point that the symbolic expressions for the nodal displacements were found, it became clear why the previous step in the process was so troublesome to calculate. The expressions for the nodal displacements are extremely long, even for such simple finite-element problems. However, now that these expressions have been found, they can be used to find expressions for the truss strains, stresses and RFs.

Initially, the truss strain was calculated by finding the symbolic expression for the total displaced length of the truss L_{disp}, and then using the known initial length of the truss L_0, to perform the strain calculation shown in Eq. (5.2). However, it was found that this leads to a symbolic expression for the truss strain that contains many absolute functions. The appearance of these absolute functions was problematic as these prevent the symbolic expression from being written purely as a sum of terms, which is eventually necessary for the QUBO problem formulation. By using the Green-Lagrange strain formulation, the stretching of the trusses is squared. This prevents the absolute functions from appearing in the symbolic strain expressions. Thus, the symbolic expressions for the strain are found by calculating the strains using the expression shown in Eq. (5.3). Under the infinitesimal deformation assumption, when $L_0 \approx L_{disp}$, Eq. (5.2) and Eq. (5.3) will be equivalent.

$$\epsilon = \frac{L_{disp} - L_0}{L_0} \tag{5.2}$$

$$\epsilon = \frac{1}{2}\left(\frac{L_{disp}^2}{L_0^2} - 1\right) \tag{5.3}$$

The expressions for truss stresses and RFs then follow simply from Eq. (5.4) and Eq. (5.5), in which the material Young's modulus E and limit stress σ_{limit} are needed.

$$\sigma = E\epsilon \tag{5.4}$$

$$RF = \frac{\sigma_{limit}}{\sigma} \tag{5.5}$$

The expressions for the displacements, strains, stresses, and RFs can be verified by filling in the values of the qubit variables that correspond to choosing a particular area. By doing so, it was seen that indeed the expected values are obtained, giving the same results as a standard finite-element analysis. As such, up until this point, the preparatory procedure for setting up symbolic expressions for the truss strains, stresses and RFs is working as intended, and is giving the correct results. For reference, the symbolic expressions for the truss stresses of each of the sample problems are available online [91].

As an example, for the two-truss problem, the expressions found for the stresses in both truss members can be evaluated. To do this a certain binary solution to the problem must be assumed. In this case, by choosing the solution $[0,1,0,0,1,0]$, this corresponds to choices of 900 mm^2 for the first truss and 1500 mm^2 for second truss. Using these values, the expression for the first truss evaluates to a stress of 110.09 MPa, and the expression for the second truss yields a value of -46.48 MPa. When verified using Abaqus, the commercial finite-element software, almost exactly the same solution is obtained, as can be seen from Fig. 5.2. This simple verification FEM model is also available online [91]. The slight differences are likely due to rounding errors in the setup of the symbolic expressions.

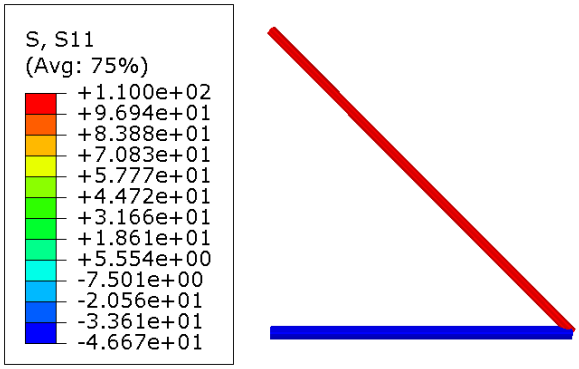


Figure 5.2: Commercial FEM analysis of two-truss problem.

Now that a method has been implemented that can correctly write the stresses and RFs in terms of qubit variables, the next step is to set up an objective function that, when minimized, should yield the most optimal choice of cross-sectional area for every truss in the structure. However, this process uncovered another challenge, which is discussed in the upcoming section.

5.2. Phase 2: Setup of the QUBO Problem

Phase 1 concluded with having found a way to set up symbolic expressions for the truss stresses and RFs for three simple FEM problems. Now, phase 2 will focus on using these expressions to set up truss optimization QUBO problems. This means that a way must be found to use these expressions to set up an objective function that describes the behavior of the full truss systems, in a QUBO-compatible format. Since quite a number of topics are discussed as a part of this phase, the following list gives a brief overview of the upcoming topics.

1. In Section 5.2.1 it will be discussed how potential candidate objective functions can be evaluated.
2. In Section 5.2.2 the expected optimal outcomes of the three sample problems are discussed. These will help to verify the results from the investigation into the objective function setup methods.
3. In Section 5.2.3 the challenge of setting up an objective function is discussed, eventually finding a functional method.
4. In Section 5.2.4 it will be seen that, although a functional objective function can be found, it is written in a QUBO-incompatible fractional format. Some methods are discussed to rewrite the objective function into a more compatible non-fractional format.
5. In Section 5.2.5 some practical implementation aspects will be discussed, pertaining to the final steps needed to write the truss sizing optimization objective function as a true QUBO problem.

Thus, it will first be discussed how a theoretical objective function can be evaluated.

5.2.1. Objective Function Evaluation Method

When symbolic expressions for the truss stresses and RFs are set up through the process described in phase 1, these tend to be long and complicated in nature. Nevertheless, the function values of these expressions can be calculated if a certain solution for the binary variables is assumed. One simple way to evaluate the nature of these expressions would be to sequentially assume every possible solution of the binary function variables, and calculate the output of the symbolic expression in every possible case. This type of approach is called a *brute-force* approach, because it relies on brute computational power to simply perform function evaluations for every possible solution to the problem. When every possible solution to the problem has been calculated, it is straightforward to find which of the solutions gives the best possible result. Due to the simplicity of the brute-force approach, this is the preferred method to gain insight into the behavior of the various objective functions that will be set up in the upcoming section of the report.

One drawback of brute-force analyses is that it can be quite time-consuming to find every possible solution to a problem, if the problem is quite large in size. In the case of the truss-optimization problems, the two-, three- and four-truss problems are defined using a total of 6, 9, and 12 binary qubit variables respectively. In turn this means that the problems have $2^6 = 64$, $2^9 = 512$, and $2^{12} = 4096$ different possible solutions respectively. The number of possible solutions scales exponentially with the size of the problem, increasing by a factor $2^3 = 8$ for each additional truss in the system. This naturally leads to significant time requirements to solve larger problems, and may eventually become intractable.

However, the effects of the exponential growth of the number of possible solutions to the truss optimization problem can be reduced. This is because it is known beforehand which solutions are valid, and which solutions are invalid. Namely, those solutions where exactly one cross-sectional area is selected for every truss in the truss structure will be considered to be valid. Solutions for which multiple areas are selected per truss, or none are selected at all, are invalid and are undesirable. Such invalid solutions can be avoided by the QA by implementing the unary constraint, which has already been seen in both Chapters 3 and 4, and will also be implemented for this symbolic finite-element QUBO method at a later stage. Thus, assuming that invalid solutions will be avoided by the QA, the brute-force verification of the different objective functions can also be performed using solely the set valid solutions to each problem. This drastically reduces the number of possible problem solutions. If N is the number of trusses in the truss system, then the full set of all possible solutions scales by 2^{3N}, while the set of only valid solutions would scale by 3^N. Although this still represents exponential growth, it is much less extreme, and leads to a total of $3^2 = 9$, $3^3 = 27$, and $3^4 = 81$ valid solutions for the two-, three- and four-truss problems respectively. Thus, the various methods of setting up truss system objective functions will be assessed by brute-force evaluation of the valid solutions to the optimization problems. In this way, the global minimum solution to these objective functions is guaranteed to be found, and the overall behavior of the objective functions can be assessed.

5.2.2. Expected Optimization Outcomes

The methods for setting up the objective functions will be evaluated using brute-force as was discussed. In this way the minimum solution is guaranteed to be found. However, one point of attention remains. Namely, if a certain minimum solution is found, how does one know if this solution makes physical sense? The answer to this question can be found by performing a number of additional classical FEM analyses on the sample truss systems, thus finding an expected optimum solution to the optimization problem. These expected

solutions will be briefly discussed, and will serve as a guide to determine if objective functions are behaving as expected.

Consider again the results initially shown in Table 5.3, reiterated in Table 5.4 for convenience. The table shows the RFs present in each of the sample truss systems when the mid-sized cross-sectional area is chosen for every truss in the system. These results are based on simply performing a standard FEM analysis of the truss-structures. Because the mid-sized cross-sectional area is chosen for every truss in the system, there are both smaller and larger options available for each of the trusses. Based on this information, an educated guess can be made as to which solution might be expected from a truss-sizing optimization, given the allowable choices.

Two-Truss System			Three-Truss System			Four-Truss System		
For element:	Area [mm^2]	RF	For element:	Area [mm^2]	RF	For element:	Area [mm^2]	RF
E1	900	0.9086	E1	500	0.7692	E1	2500	1.7680
E2	1500	2.1511	E2	1050	1.3252	E2	2500	1.7680
-	-	-	E3	800	7.8331	E3	2000	0.8910
-	-	-	-	-	-	E4	2500	1.2859

Table 5.4: Truss system reference reserve factors.

Recall that the most lightweight truss structure that still complies with the material stress requirements will have $RF = 1$ for every truss in the structure. Thus, looking at the RFs for the two-truss system, it is seen that the RF for the first element is too low, while for the second element it is too high. Thus, given that both smaller and larger options are available for both elements, it could be expected that the optimal result would be to use a larger cross-sectional area for the first element, and a smaller one for the second element. Since, the RF is expected to behave roughly proportionally to the change in cross-sectional area. In other words, it is expected that increasing the cross-sectional area of a truss leads to a decrease of the truss stresses, in turn leading to an increase of the RF. Applying this logic to the two-truss problem, this would mean that the first element would optimally have an area of 1000 mm^2, and the second element would use an area of 1400 mm^2. Another FEM analysis can be performed using these choices to see if this selection brings the truss RFs closer to the optimal value of 1. Indeed, it is seen that using these choices, RFs of 1.0096 and 2.0062 are found for the first and second truss elements, both being closer to the optimal value. As such, the expected result for the two-truss system, in terms of binary variables, is $[0,0,1,1,0,0]$.

In a similar vein, the expected results for the three- and four-truss problems can also be determined. By first observing the RFs from Table 5.4, in which all trusses used their mid-sized area choices, a certain change can be expected which may bring each truss closer to the optimal situation where the RFs are close to a value of 1. For the three-truss system this means that the expected outcome would be a larger cross-sectional area for the first truss, and smaller cross-sectional areas for the second and third trusses. Verifying with a FEM analysis, using $A_1 = 600$ mm^2, $A_2 = 950$ mm^2, and $A_3 = 700$ mm^2, the RFs are found to be $RF_1 = 0.8586$, $RF_2 = 1.3206$, and $RF_3 = 48.2891$ respectively. In terms of the binary problem variables, this solution would be represented as $[0,0,1,1,0,0,1,0,0]$. It can be seen that in this configuration, the reserve factor for the third truss becomes much higher compared to the original configuration of cross-sectional areas. Specifically, in the original configuration the third truss has $RF_3 = 7.8331$, which increases to $RF_3 = 48.2891$ upon making the expected change to the cross-sectional areas. This goes against the intuition where an increase in cross-sectional area is expected to lower truss stresses and increase RFs. Thus, this indicates that the problem might be difficult to solve. The cause of this behavior is likely due to force redistribution within the truss structure, due to the significant change in cross-sectional area of the first and second trusses.

Finally, for the four-truss system, observing the RFs in the initial configuration, it is expected that the optimum choices would be to increase the cross-sectional area of the third truss element, while decreasing the area of the first, second, and fourth elements. Thus, this is again checked with a standard FEM analysis, using the areas $A_1 = 2400$ mm^2, $A_2 = 2400$ mm^2, $A_3 = 2100$ mm^2, and $A_4 = 2400$ mm^2. Doing so results in finding $RF_1 = 1.6973$, $RF_2 = 1.6973$, $RF_3 = 0.9354$, and $RF_4 = 1.2323$. This solution would correspond to $[1,0,0,1,0,0,0,0,1,1,0,0]$ in terms of the binary problem variables. It can be seen that for this solution indeed each of the trusses gets closer to the optimum value of $RF = 1$, compared to the original configuration.

5.2.3. Challenge: Setting up an Objective Function

As of yet, every truss has its own individual symbolic expressions for the stresses and RFs. Furthermore, it is now known that such symbolic expressions can be evaluated by brute-force to investigate their behavior for valid solutions to the optimization problem, and the expected optimal results have also been discussed.

Thus, it is now time to find a single expression that describes the entire truss system, and which can be used to define the truss sizing optimization problem. This will be the objective function for the optimization.

The most optimal design, for this sizing optimization, is the truss structure design that is the most lightweight, while still complying with the material strength allowables. In essence, if every truss in the system is brought as close as possible to the allowed limit stress, giving a reserve factor close to 1, this would also describe the minimum weight configuration. Since, if any more material were to be removed from any of the trusses, the reserve factors would fall below a value of 1, meaning strength requirements are not being met. The following sections detail the chronological development of a suitable objective function, which proved to be quite challenging. The first attempts at setting up an objective function are related to the RF, while the final attempt directly uses the expressions for the truss stresses. Throughout this process, the MATLAB software package was used to perform the manipulations of the various symbolic expressions. The code is available online [91].

Initial Objective Function

The first method that was used to set up an objective function utilized the expressions that were found for the truss RFs. It was seen that the symbolic expressions for the reserve factors are of a fractional form. The numerator and denominator of these expressions both contain very many high-order qubit terms. Aside from the fact that terms of higher than quadratic order are not allowed in QUBO problems, this fractional expression for the reserve factor cannot be used directly in the QUBO formulation. This is because in QUBO problem formulations only pure sums of linear and quadratic qubit terms can be taken into account. However, since the QA is inherently designed for solving minimization problems, this fractional expression can possibly be rewritten such that it becomes a target for a minimization problem.

It is known that the most optimal value of the reserve factor will be 1. This means that, if the reserve factor is written as a fraction, the numerator and the denominator should be equal to each other. In this case, it should therefore also be the case that the numerator minus the denominator should yield a result of zero, in the optimal case. However, assuming that the currently known symbolic expression for the truss reserve factor actually describes a sub-optimal case, this will result in an error term. An optimization problem can then be set up for which the goal is to minimize this error, by simply squaring the expression. This ensures that the optimum solution will be the one where the squared error is closest to zero, giving an objective function for the minimization. The steps to this thought process are shown in Eq. (5.6).

$$\begin{aligned} \text{If: } RF = \frac{N}{D} &= 1 \\ \text{Then: } &\rightarrow N = D \\ \text{Optimally: } &\rightarrow N - D = 0 \\ \text{Sub-optimally: } &\rightarrow N - D = \epsilon \\ \text{Optimizing: } &\rightarrow \min\left(\epsilon^2\right) \Rightarrow \min\left((N-D)^2\right) \end{aligned} \tag{5.6}$$

If this objective function for the minimization is set up for every truss individually, it may be reasonable to expect the sum of each of these expressions to encode the minimization problem for the entire truss system. Since, the objective would therefore be to find the minimum sum of all individual truss errors. As such, using this method, the objective function for the whole system of trusses becomes:

$$T = \sum_{n=1}^{N} (N_n - D_n)^2 \tag{5.7}$$

Testing and Results This objective function was implemented and tested using MATLAB [91]. Using a simple brute-force analysis, using the set of valid answers to the optimization problem, the results for the two-, three-, and four-truss problems are shown in Figs. 5.3 to 5.5. In each case, the minimum solution is indicated with a small red circle.

For the two-truss problem, the minimum solution is found to be $[0,1,0,1,0,0]$, which indicates the selection of the mid-sized cross-section for the first truss, and the smallest possible cross-section for the second truss. For the three- and four-truss problems, the minimum solutions are found to be $[1,0,0,1,0,0,1,0,0]$ and $[1,0,0,1,0,0,1,0,0,1,0,0]$ respectively. Both of these solutions indicate that the smallest possible cross-section should be chosen for all trusses in the truss systems. These results are contrary to what was expected, based on the arguments provided in Section 5.2.2. As an example, for the four-truss problem, the expected

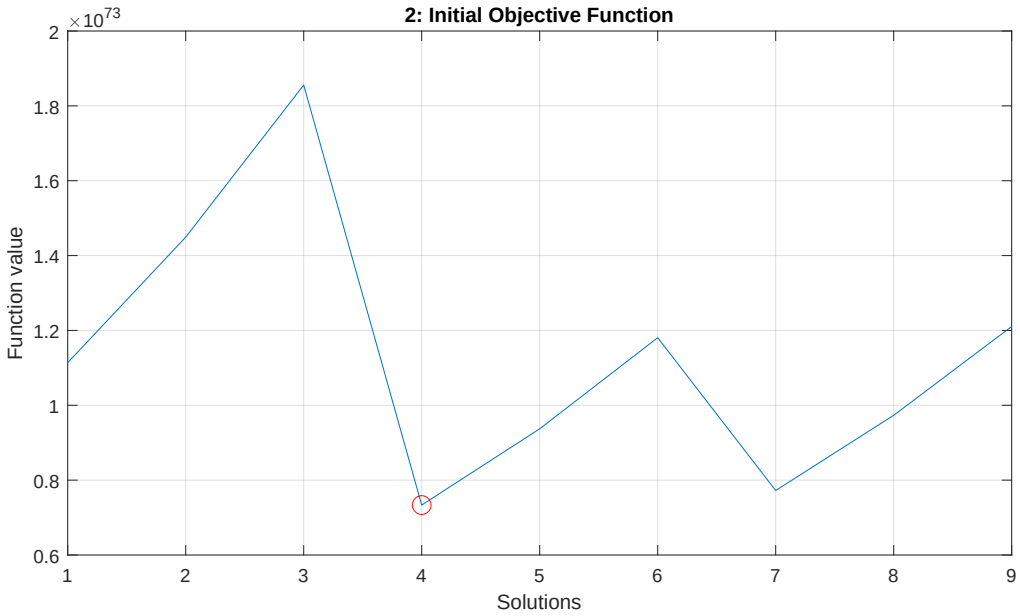


Figure 5.3: Initial objective function for the two-truss problem. Global minimum is found to be solution number 4, corresponding to $[0,1,0,1,0,0]$.

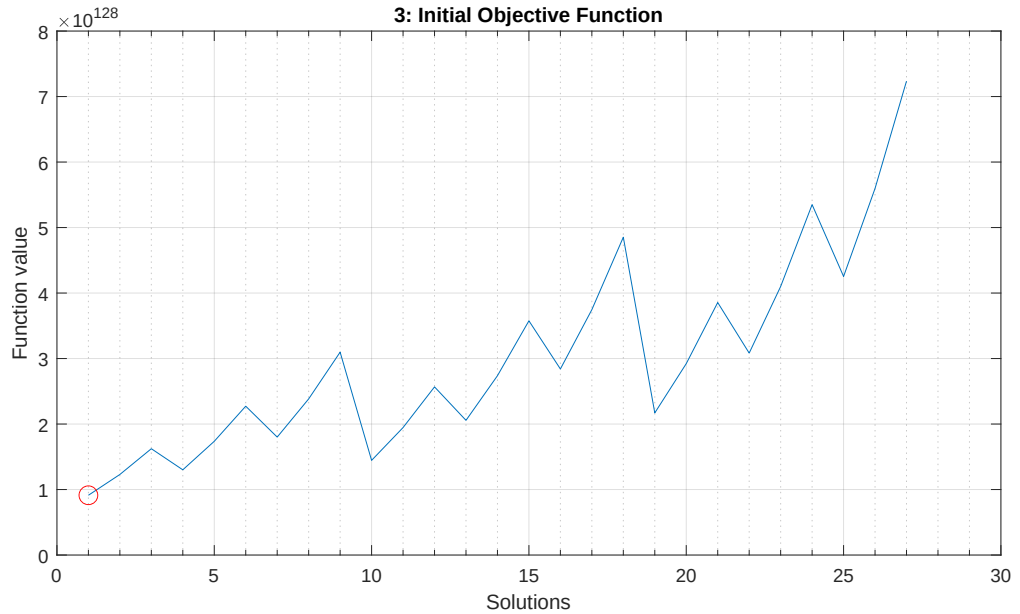


Figure 5.4: Initial objective function for the three-truss problem. Global minimum is found to be solution number 1, corresponding to $[1,0,0,1,0,0,1,0,0]$.

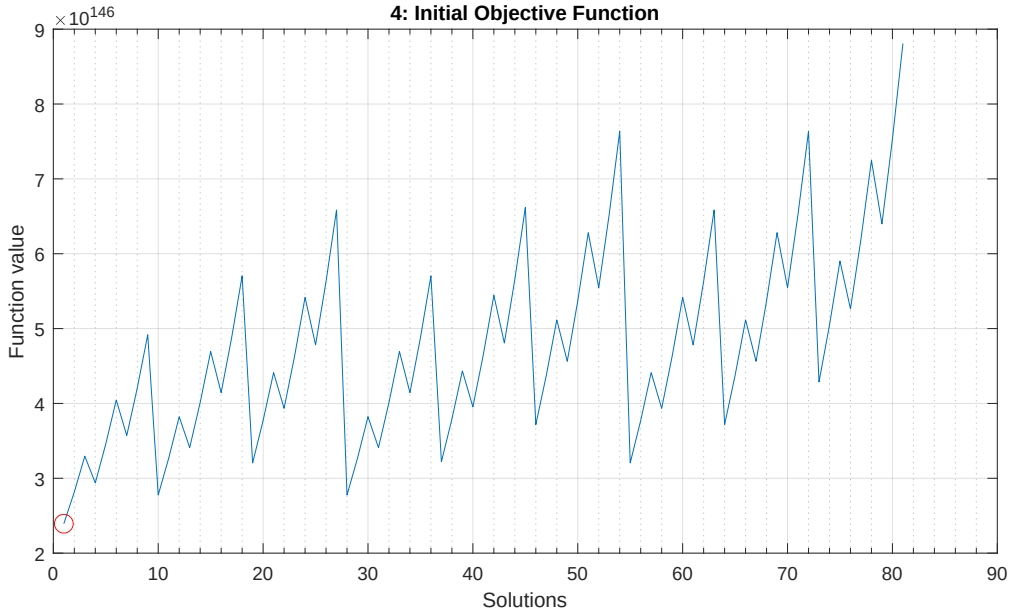


Figure 5.5: Initial objective function for the four-truss problem. Global minimum is found to be solution number 1, corresponding to [1,0,0,1,0,0,1,0,0,1,0,0].

optimum is that the third truss is would use the largest possible cross-section, rather than the smallest of the given choices. Hence, an error may be present in the proposed method.

After some investigation, a source of erroneous behavior was found. The oversight in this method is that the large symbolic expressions for the reserve factors rely on the expressions for the stresses. In turn, the symbolic stress expressions evaluate to negative values for trusses that are in compression. This means that the RF also becomes a negative number for compressive trusses, which in turn means that the calculation to set up the objective function stops working properly. When the RF is calculated by $RF = N/D$ and the term D is a negative number, this means that the objective function is calculated using the sum of N and D, rather than calculating the difference between N and D. This means that the objective function would work inversely for compressive trusses. Naturally, the next step in the development process was to correct this error.

Improved Objective Function

It was previously found that the objective function was not working properly with compressive trusses, due to the stress in these trusses being negative. One idea to solve this issue might be to apply an absolute function to the expressions for the truss stresses, such that they always return positive values. However, this would be problematic, as mathematical functions cannot be applied to the terms in QUBO problem formulations. Since, by definition, QUBO problems can only consist of a summation of linear and quadratic terms. Mathematical functions, such as the absolute function, would therefore prevent the objective function from being written in a QUBO-compatible form.

However, there is a workaround for this issue of not being allowed to use the absolute function for the truss stresses. Namely, by squaring the expressions for the stresses, a new quantity is obtained that will always be positive. Then, the squared stress can be used to find an expression for the squared reserve factor. The expression for the squared reserve factor could then be used to set up a new objective function, for which the minimum solution is sought. The squared reserve factor, for every truss n in the truss system, will have the form shown in Eq. (5.8).

$$RF_n^2 = \frac{N_n^2}{D_n^2} \tag{5.8}$$

Similar to the previous case, the expression for the squared reserve factor is of a fractional form. Furthermore, the optimal solution for the squared reserve factor will still have a value of 1, which means that the numerator and denominator should still be equal to each other. As such, the objective function for a single truss is set up as:

$$T_n = (N_n^2 - D_n^2)^2 \tag{5.9}$$

with the objective function for the complete system of trusses then being found by taking the sum for all trusses.

$$T = \sum_{n=1}^{N} (N_n^2 - D_n^2)^2 \tag{5.10}$$

This reformulated objective function is again tested with the sample problems, using the brute-force analysis method, to find out if the minimum solution makes sense given the problem that is defined.

Testing and Results The improved objective function is again tested via brute-force analysis [91]. The results of these analyses are shown in Figs. 5.6 to 5.8, with the global minimum solutions indicated with a small red circle. For the two-truss problem, the optimum is found at solution number 7, which in turn corresponds to the binary solution $[0,0,1,1,0,0]$. For both the three-, and four-truss problems, both optimum solutions are found at solution number 1, corresponding to $[1,0,0,1,0,0,1,0,0]$ and $[1,0,0,1,0,0,1,0,0,1,0,0]$.

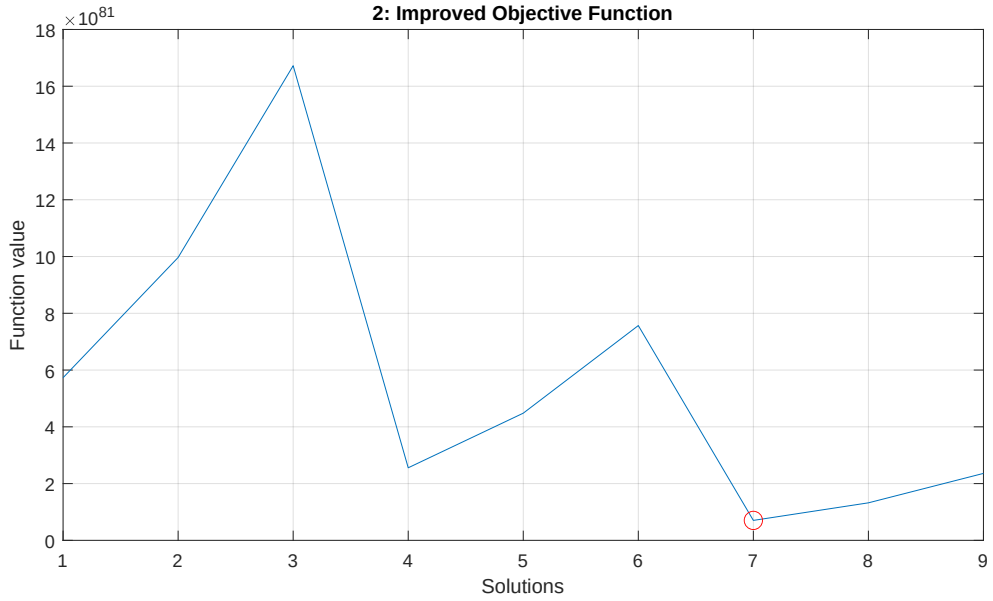


Figure 5.6: Improved objective function for the two-truss problem. Global minimum is found to be solution number 7, corresponding to $[0,0,1,1,0,0]$.

Using the improved objective function method, the expected solution for the two-truss problem is indeed returned. Since, the binary solution $[0,0,1,1,0,0]$ indicates that the first truss element should choose the largest possible cross-section, while the second truss element should optimally take the smallest cross-section. However, it is seen that, for the three- and four-truss problems, the improved objective function method is still not providing the expected results, indicating that the smallest cross-sections should be chosen for all trusses in these truss systems.

Even though the improved objective function is now providing the expected result for the two-truss problem, the current method is still presumed to be flawed. Since, for the more complicated three- and four-truss problems, the expected results are not obtained. The reason for this might be that the method includes a step to mitigate a fractional form. In the improved objective function method, the difference between N^2 and D^2 is considered, rather than considering the ratio N^2/D^2 which defines the squared reserve factor. A solution that minimizes the difference between N^2 and D^2 is not necessarily the same solution for which the ratio between N^2 and D^2 is minimal. Hence, this may be the source of the unexpected behavior produced by this objective function method.

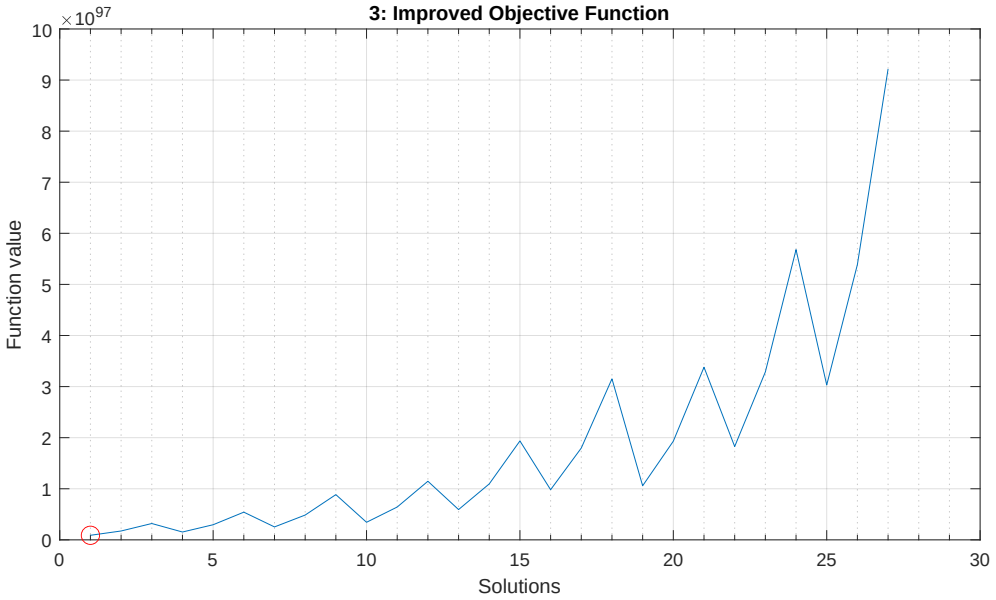


Figure 5.7: Improved objective function for the three-truss problem. Global minimum is found to be solution number 1, corresponding to $[1,0,0,1,0,0,1,0,0]$.

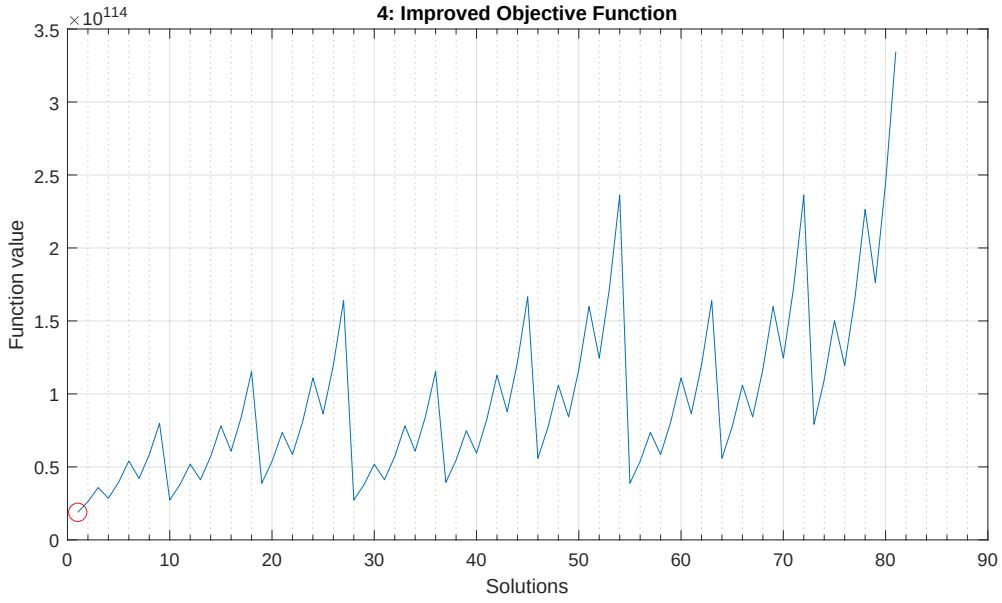


Figure 5.8: Improved objective function for the four-truss problem. Global minimum is found to be solution number 1, corresponding to $[1,0,0,1,0,0,1,0,0,1,0,0]$.

The objective function setup methods thus far have included steps to mitigate a fractional form for the objective functions, but have not been successful in reliably finding the expected minimum solutions via brute-force analysis. An alternative option, which may find the expected solutions more reliably, could be to set up a fractional objective function. However, by having a fractional objective function, the function would be much less compatible with the QUBO problem formulation in which only a summation of linear and quadratic terms is allowed. Nevertheless, a fractional objective function could still be evaluated using brute-force analysis to investigate if it produces the expected results more reliably. Thus, in the next section, the setup of a fractional objective function will be investigated via brute-force. If it is found that such an objective function indeed provides the expected results via brute-force, then an additional method must be found by which fractional objective functions can be made compatible with the QUBO problem format. Otherwise, it will not be feasible to implement this problem on the QA.

Fractional Objective Function

The idea that reserve factors should be equal to 1 was previously used to allow for the objective function to be written in a non-fractional form. However, the testing of these methods showed that they were not reliably finding the expected optimum solution. It is therefore necessary to investigate the theoretical results if the objective function is intentionally kept in a fractional form. In this case, there is no longer a need to rely on an expression for the reserve factor. Rather, an objective function can be set up that directly considers the difference between the allowable stress and the actual stress in the truss. Because the expressions for the truss stresses are fractional in nature, the final objective function will also become a fraction using this method.

Setting up the fractional objective function, it is again important to consider that the truss stresses evaluate to negative numbers for trusses in compression. As such, it is necessary to square the expression for the truss stress. For consistent units this also means that the maximum allowable stress must be squared. For a single truss n, this leads to the fractional target objective function shown in Eq. (5.11).

$$T_n = \left(\sigma_{limit}^2 - \sigma_n^2\right)^2 \tag{5.11}$$

With Eq. (5.11), the minimum solution should encode the choice of truss cross-sectional area that minimizes the absolute difference between the limit stress and the truss stress. In turn, this choice implicitly brings the RF as close as possible to 1, and the weight of the truss is therefore minimized. Furthermore, if Eq. (5.11) describes the difference between the limit stress and the current stress in a single truss, then the goal for the entire truss system would be to minimize the sum of these differences for every truss in the structure. Thus, to set up an objective function that describes the entire truss structure, the summation of the target expressions of every individual truss is taken, which leads to Eq. (5.12).

$$T = \sum_{n=1}^{N} T_n = \sum_{n=1}^{N} \left(\sigma_{limit}^2 - \sigma_n^2\right)^2 \tag{5.12}$$

With Eq. (5.12) a general method is obtained that can be used to set up a fractional objective function for each of the sample problems. These objective functions can then be evaluated using a brute-force analysis, to find if it gives the expected results in a reliable manner. Unfortunately, it is impossible to implement this fractional objective function for use directly with the QA, since it is incompatible with the QUBO formulation.

Testing and Results As with the previous attempts at formulating an objective function, this new fractional objective function is also investigated using a brute-force method. The results for the three sample problems are shown in Figs. 5.9 to 5.11, and were produced using the MATLAB code which is available online [91]. For the two-truss problem, the minimum is found at solution number 7, corresponding to $[0,0,1,1,0,0]$. The three-truss problem has the global minimum at solution number 21, corresponding to the binary solution $[0,0,1,1,0,0,0,0,1]$. Finally, the four-truss problem has the minimum at solution number 7, which in this case corresponds to $[1,0,0,1,0,0,0,0,1,1,0,0]$.

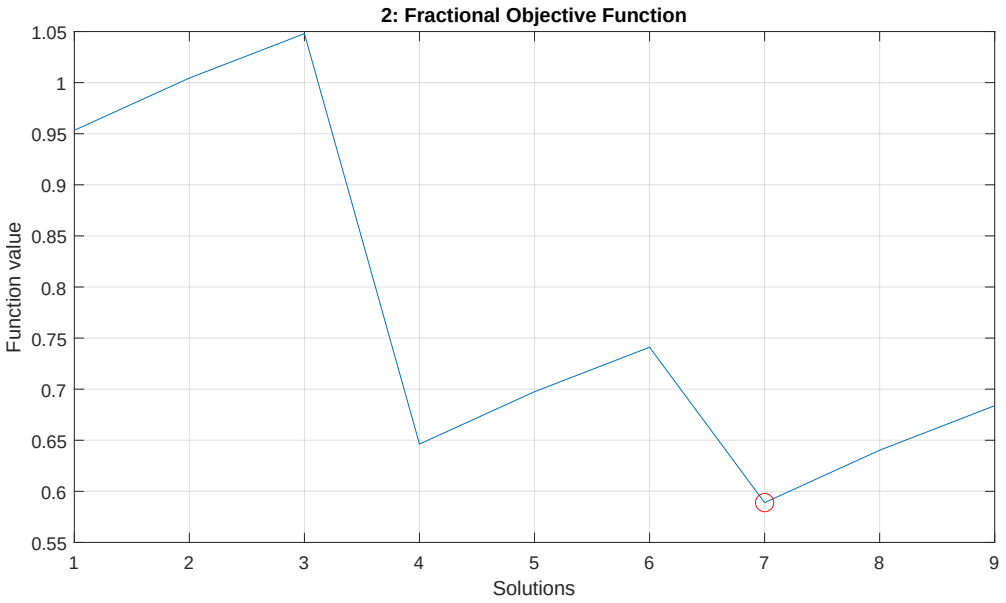


Figure 5.9: Fractional objective function for the two-truss problem. Global minimum is found to be solution number 7, corresponding to $[0,0,1,1,0,0]$.

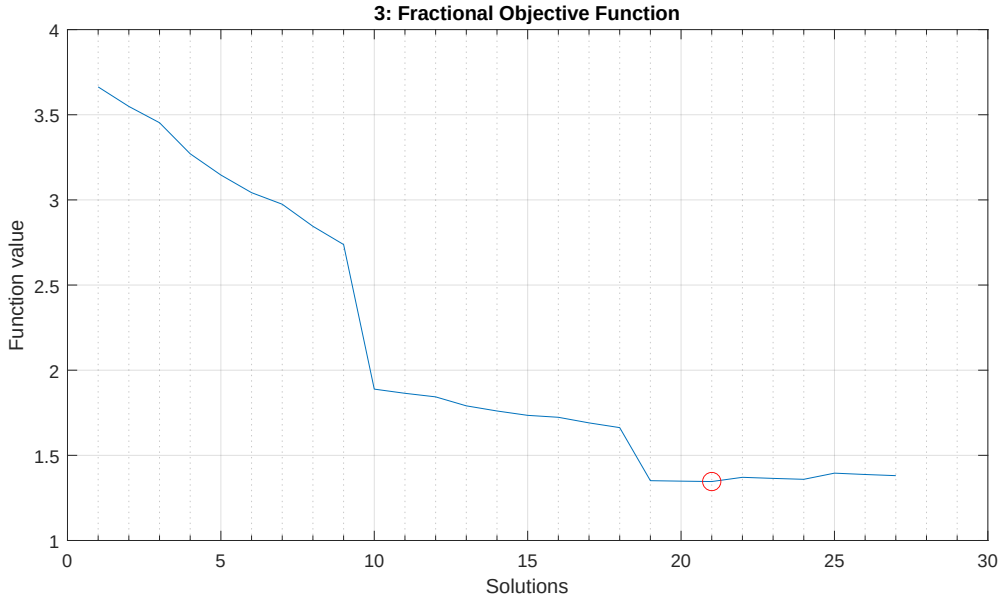


Figure 5.10: Fractional objective function for the three-truss problem. Global minimum is found to be solution number 21, corresponding to $[0,0,1,1,0,0,0,0,1]$.

From these results, it can be seen that the fractional objective function provides the expected outcomes for both the two- and four-truss problems. This indicates that the fractional objective function method may be useful for solving these truss optimization problems. However, for the three-truss problem, the fractional objective function does not entirely yield the expected result. In this case, the result $[0,0,1,1,0,0,0,0,1]$ indicates that the third truss should optimally use the largest cross-section, rather than the initially expected smallest cross-section. The results for the first and second trusses in the system do agree with the expectation from Section 5.2.2. From Fig. 5.10 it can also be seen that solutions 19 and 20 are extremely close to the optimum, having only slightly higher function values compared to the optimum solution. To be precise, the function values are $T(\mathbf{q}_{19}) = 1.3512$, $T(\mathbf{q}_{20}) = 1.3485$, and $T(\mathbf{q}_{21}) = 1.3461$. These three solutions only have different cross-sectional areas for the third truss, all having the same choices for the first two trusses. Thus, for the three-truss problem, it evidently does not make much difference for the value of the objective function what cross-section is chosen for the third truss, as long as the expected choices are made for the first and

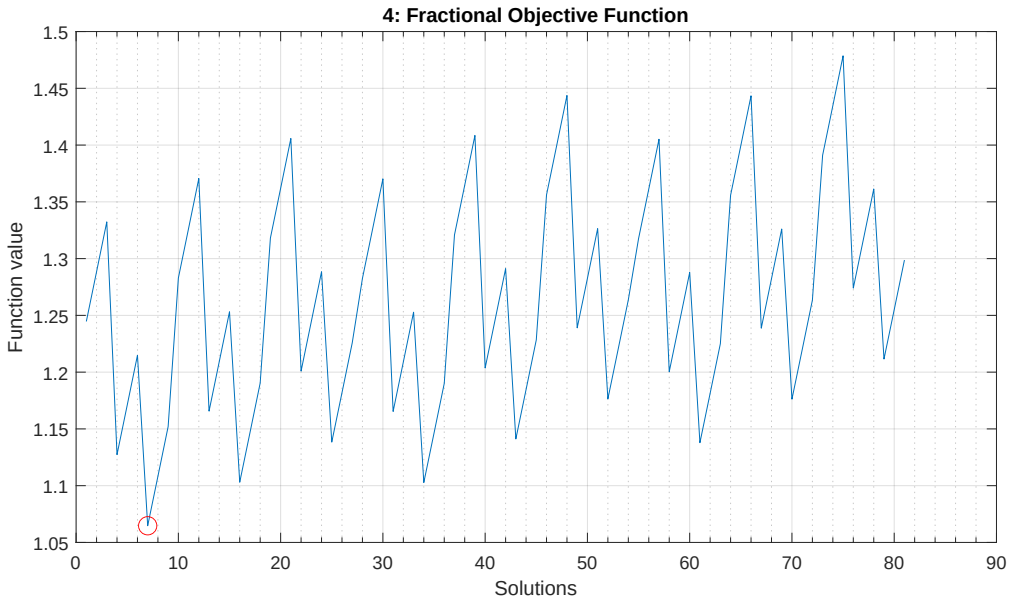


Figure 5.11: Fractional objective function for the four-truss problem. Global minimum is found to be solution number 7, corresponding to $[1,0,0,1,0,0,0,0,1,1,0,0]$.

second trusses.

Initially, in Section 5.2.2, it was already seen that the three-truss problem could cause difficulties. Under the assumed expected solution, it was seen that the RF for the third truss would increase from 7.8331 to a value of 48.2891. This goes against the intuition that decreasing the truss cross-sectional area would lead to higher truss stresses, and thus lower the RF. Instead, for the third truss in the three-truss system, the opposite behavior is seen. This is likely due to the design of the truss structure, having the third truss in a nearly horizontal orientation. The load acting on the central node is acting vertically downward. Thus, the first and second trusses will carry the majority of the load, since they are better aligned with the loading direction compared to the nearly-horizontal third truss. However, if the stiffness of the first two trusses is significantly changed, this can lead to a force redistribution within the structure. This can decrease the load that is carried by the third truss, in turn meaning that the stresses decrease and the RF increases. Since the stiffness of the first and second trusses is significantly changed, the force redistribution within the structure is expected to be the reason for the counter-intuitive behavior of the third truss.

Overall, the fractional objective function works as expected for the two- and four-truss problems. Furthermore, for the three-truss problem, it is seen that the choice of cross-sectional area of the third truss has little effect on the value of the objective function. For the first and second trusses in the three-truss system the fractional objective function works as expected. The deviation from the expected result in this case is caused by the fact that the problem itself is quite difficult, and is not necessarily due to mistakes in the formulation of the objective function. Thus, even though some improvements to the methodology may still be possible, the fractional objective function using the current method works well enough to merit further investigation. Because of the fact that the fractional objective function cannot be used directly with the QA, since it is incompatible with the QUBO problem format, the next section of the thesis discusses this challenge.

5.2.4. Challenge: Fractional Objective Functions

In the previous section, it was seen that an objective function can be set up for which the minimum solution gives a selection of truss cross-sectional areas that bring every truss closest to the assumed limit stress. However, the problem with this specific objective function formulation is that it is fractional in nature, which is not compatible with the QUBO problem framework and therefore not directly usable with the QA. In the earlier attempts at setting up an objective function, methods were tried to set up non-fractional objective functions. Unfortunately, these methods were shown to be ineffective at yielding the expected results. As such, the fractional objective function presents another challenge to be overcome along the path to using a QA for the truss sizing optimization problem. In this section, methods for translating the fractional objective function to an equivalent non-fractional form are investigated.

Original Conversion Method

Assuming a basic fractional objective function of the form:

$$T_f = \frac{N}{D} \tag{5.13}$$

where the subscript f indicates the fractional form, there might be several ways one could consider converting the function to an equivalent non-fractional form. The previous attempts at converting to a non-fractional objective accomplished the conversion by:

$$T_{nf} = (N-D)^2 \tag{5.14}$$

in which the subscript *nf* indicates 'non-fractional'. This conversion was thought to be feasible, since optimally $T_f = 1$ should hold. Because this means that $N = D$, taking $(N-D)^2$ was thought to lead to an optimization problem. However, the minimum solution of T_{nf} is not necessarily the same as the minimum solution of T_f. As a brief example, imagine a certain fractional objective function has two possible solutions:

$$T_{f,1} = \frac{11}{10} = 1.1 \qquad \text{and} \qquad T_{f,2} = \frac{101}{100} = 1.01 \tag{5.15}$$

In this case, the minimal solution of the two available options would be the second, $T_{f,2} = \frac{101}{100}$. However, conversion into a non-fraction using the method of Eq. (5.14) does not yield any difference between both solutions. Since, performing the conversion, $T_{nf,1} = (11-10)^2 = 1$ and $T_{nf,2} = (101-100)^2 = 1$. This method is therefore problematic, as it does not preserve the same minimum solution, and therefore fails to set up an equivalent non-fractional objective fuction.

Taylor Series Approximation

Another method that was attempted for converting the fractional objective into a non-fraction was through using a Taylor series approximation. Aside from helping to write the objective as a non-fraction, the Taylor series approximation may also help to reduce the highest order of the terms appearing in the objective function. Originally the fractional objective function has many high-order terms, where many of the binary qubit variables are multiplied together. Using the Taylor series approximation, the highest order of terms allowed in the approximation becomes a user-defined parameter. Given a fractional objective function:

$$T_f(\mathbf{q}) = \frac{N(\mathbf{q})}{D(\mathbf{q})} \tag{5.16}$$

in which the vector of binary qubit variables is written as $\mathbf{q}$. This fractional objective function has high-order functions of all binary qubit variables in both the numerator and the denominator. For the Taylor series expansion it is also necessary to assume a vector of values for the qubit variables, around which the approximation will be valid. This vector of assumed values is written as $\mathbf{a}$. The Taylor series expansion for this multi-variable function is then given as:

$$T_{nf}(\mathbf{q}) \approx T_f(\mathbf{a}) + (\mathbf{q}-\mathbf{a})\frac{\partial T_f(\mathbf{a})}{\partial \mathbf{q}} + \frac{1}{2}(\mathbf{q}-\mathbf{a})^2 \frac{\partial^2 T_f(\mathbf{a})}{\partial \mathbf{q}^2} + \ldots \tag{5.17}$$

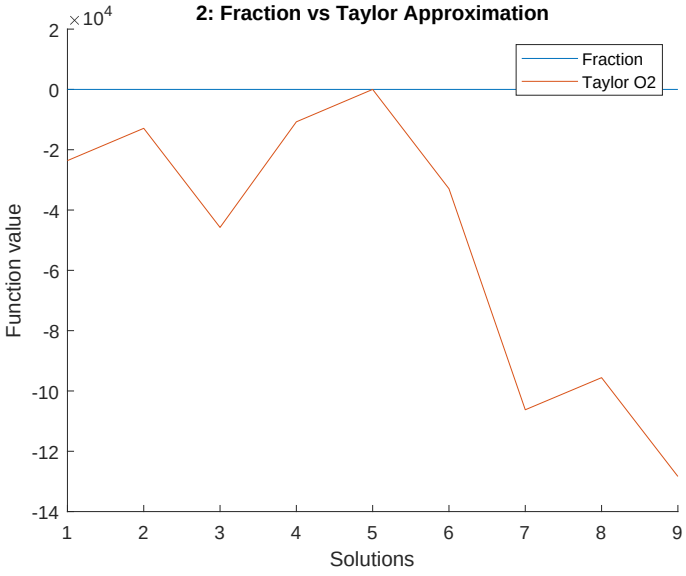
In this case, the expansion is written up to the second order, but higher-order expansions are possible. The constant term $T_f(\mathbf{a})$ could actually be dropped, since it contributes the same amount to every possible solution, and therefore does not influence which solution is minimal. By finding the second-order expansion the objective would be written as a summation of linear and quadratic terms. This in turn means that a second-order Taylor approximation of the fractional objective function would automatically make the approximation compatible with the QUBO problem framework.

For the assumed solution $\mathbf{a}$ of the Taylor approximation, several possibilities exist. Either, a particular solution to the problem can be chosen, such as the solution for which all trusses utilize a medium-sized cross-section. Alternatively though, despite the variables $\mathbf{q}$ being binary, every variable can be assumed to be in a 'superposition' state, i.e. having a value of $\frac{1}{2}$. Both of these methods were implemented and tested, investigating the resulting approximated objective function through brute-force analysis of every possible valid solution to the problem.

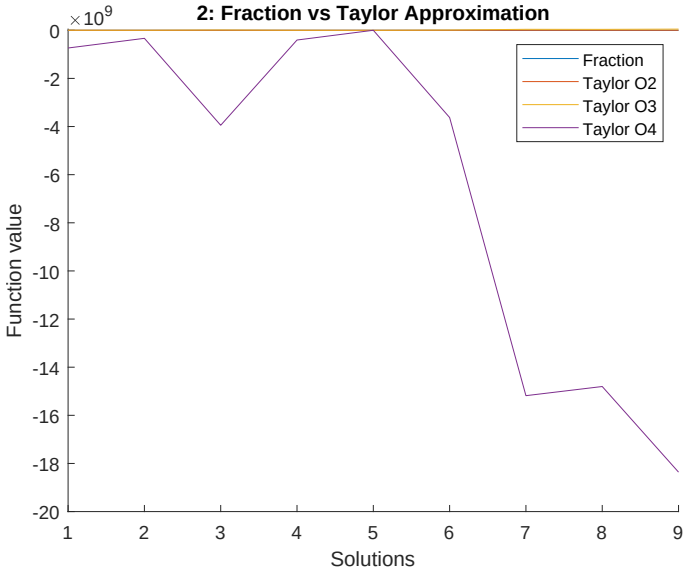
Testing and Results The Taylor series approximation method was tested for the simplest sample problem that has been defined, namely the two-truss structure. In this case, the problem is defined such that there is a choice of three different cross-sectional areas for each truss, meaning that in total there are six different qubit variables that must be taken into account.

Through some initial testing it quickly became apparent that for this particular problem it is quite time-consuming to find the fifth- and sixth-order Taylor approximations. Since, the number of symbolic derivatives that must be calculated and evaluated increases exponentially with every additional order of approximation. Therefore, only results of second-, third-, and fourth-order Taylor series approximations are investigated and compared to the original fractional objective function.

The following plots show the brute-force solutions of the Taylor approximations and the original fractional objective function. The first plot, given in Fig. 5.12a, shows the second-order Taylor approximation results, using the mid-sized solution as the solution around which the approximation is calculated. The exact cross-sectional areas for the mid-sized solution can be found in Table 5.2, using the areas found in the 'Option 2' column, for every truss in the two-truss problem.



(a) Second-order Taylor approximation.



(b) Taylor approximations up to fourth order.

Figure 5.12: Taylor approximations of the fractional objective function of the two-truss problem around the mid-sized solution.

It can be seen that, performing the approximation around the assumed mid-sized solution, the second-order Taylor approximation does an extremely poor job at approximating the original fractional objective function. The order of magnitude on which the approximated function varies is much greater than that of the original function, making the original look like a simple horizontal line. The approximation is only close to the true value for solution number 5, which is exactly the mid-sized solution around which the approximation should be valid. The approximation therefore seems to be working correctly, but fails to capture the behavior of the fractional objective function for every other solution, aside from the initially assumed solution. Increasing the order of the approximation only exacerbates the errors, as can be seen from Fig. 5.12b.

Clearly then, choosing the mid-sized solution for the Taylor approximation is not yielding usable results. The approximated functions yield erroneous solutions for all but the initially assumed solution, and fail to correctly approximate the shape of the original fractional objective function, giving different minimum solutions.

The second option for the assumed solution is to assume every qubit is in a 'superposition' state, between the valid binary values. Thus, every qubit variable is assumed to have a value of $\frac{1}{2}$. The plots shown in Fig. 5.13 show the results for the Taylor approximations up to the fourth order.

In this case, it can be seen that the approximation yields much more reasonable results, at least being in the same order of magnitude as the original fractional objective. However, the approximations still do not capture the shape, nor the same minimum solution as that of the original fractional objective function. Since, the original function has its minimum at solution number 7, while the approximations consistently have their minima at solution number 9. These solutions correspond to $\mathbf{q}_7 = [0,0,1,1,0,0]$ and $\mathbf{q}_9 = [0,0,1,0,0,1]$.

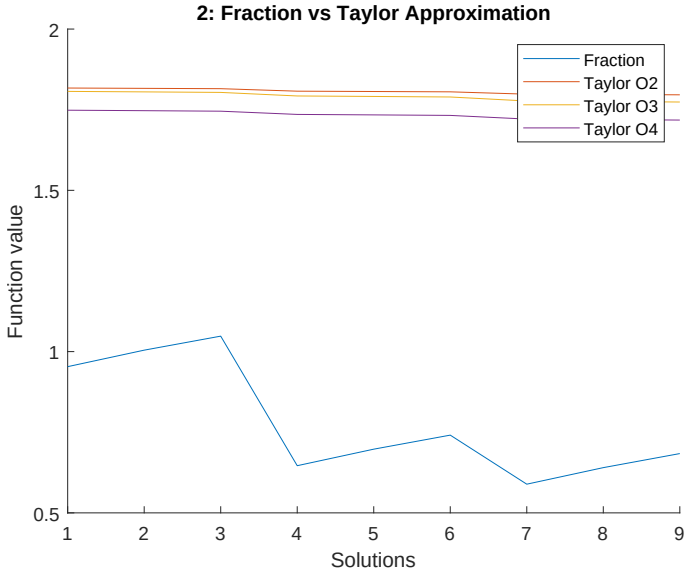


Figure 5.13: Taylor approximations using 'superposition' solution.

Since in both cases the Taylor approximation fails to capture the correct shape and minimum solution as that of the original fractional objective function, the Taylor series approximation method is deemed unsuitable for use in converting the fractional objective function into an equivalent non-fractional function. As such, further literature on possible conversion methods for this purpose must be investigated.

Literature on Conversion Methods for Fractional Functions

The conversion methods attempted thus far have not yielded equivalent non-fractional functions. However, this problem remains of critical importance to solve if a practical implementation for the QA is desired. Thus, it is necessary to look further into the literature to find possible methods that might be successful. One field in which some success may be found is the field of Linear Fractional Programming (LFP).

Although the current fractional objective function is definitely non-linear, considering the many high-order multiplications in the function, it may be possible to reformulate the fractional function as an LFP problem. For instance, the two-truss problem relies on six individual binary qubit variables. The fractional expression however, contains many terms with different products of these original six variables. An idea might be to define a new auxiliary variable, which can be used to replace each of the higher-order products. For example, suppose the following fractional function F exists, as a function of the binary variables in $\mathbf{q}$:

$$\begin{aligned} \mathbf{q} &= \left[q_1, q_2, q_3, q_4, q_5, q_6\right]^T \\ F(\mathbf{q}) &= \frac{q_1 + q_2 + q_3 + q_3 q_4 + q_4 q_5 q_6}{q_1 + q_2 + q_3 + q_4 + q_5 + q_6} \end{aligned} \tag{5.18}$$

The non-linear terms $q_3 q_4$ and $q_4 q_5 q_6$ might respectively be replaced by the linear auxiliary variables q_{a1} and q_{a2}. In order to make this replacement valid, constraints will be necessary to enforce that $q_{a1} = q_3 q_4$ and $q_{a2} = q_4 q_5 q_6$. Without considering the specifics of how such a constraint might need to be enforced, it seems plausible that this method could at least be used to convert the unconstrained non-linear fractional objective function into an LFP problem with constraints. As such, what follows is a brief look into some of the literature that was uncovered on the conversion of LFP problems to Linear Programming (LP) problems.

Method by Hasan and Acharjee From the literature on LFP problems, some clues have been found with respect to converting LFP problems into simpler Linear Programming (LP) problems. Specifically, a publication by Hasan and Acharjee indicates that the conversion of LFP to LP is a field that has been active since at least the 1960s [40]. According to Hasan and Acharjee, the existing methods for LFP to LP conversion are complicated or computationally expensive [40]. However, the new method proposed by the authors appears to be fairly straightforward and will be shown below. An LFP is defined as shown in Eq. (5.19).

$$Z = \frac{cx + \alpha}{dx + \beta} \tag{5.19}$$

The LFP can be transformed to an LP by rewriting in the following manner:

$$\begin{aligned} Z &= \frac{cx+\alpha}{dx+\beta} \\ &= \frac{cx+\alpha}{dx+\beta}\frac{\beta}{\beta} \\ &= \frac{cx\beta+\alpha\beta}{\beta(dx+\beta)} \\ &= \frac{cx\beta-dx\alpha+dx\alpha+\alpha\beta}{\beta(dx+\beta)} \\ &= \frac{(c\beta-d\alpha)x+\alpha(dx+\beta)}{\beta(dx+\beta)} \\ &= \left(c-d\frac{\alpha}{\beta}\right)\frac{x}{dx+\beta}+\frac{\alpha}{\beta} \end{aligned} \tag{5.20}$$

Then, defining:

$$p=\left(c-d\frac{\alpha}{\beta}\right) \qquad y=\frac{x}{dx+\beta} \qquad g=\frac{\alpha}{\beta} \tag{5.21}$$

the LFP then becomes the LP:

$$F(y)=py+g \tag{5.22}$$

One major caveat for this method, however, is that it only works when $\beta \neq 0$. In the case of the fractional objective function that is obtained for the truss sizing optimization problem, the function does not contain any constant terms. This means that for the truss problem, the situation arises that $\beta = 0$, and the method described by Hasan and Acharjee will not work. Regardless, some brief testing was performed using this method of converting the fractional two-truss sizing optimization problem into a non-fraction assuming various values for α and β. However, the results were not representative of the original fractional objective function, and the method was therefore abandoned, in search of other alternatives.

Method by Simi and Talukder More recently, a new approach to converting LFP to LP problems was formulated by Simi and Talukder [79]. In essence, this method is quite similar to the approach that was initially attempted, where the denominator of the fractional function is subtracted from the numerator, to arrive at a non-fractional objective. However, the approach includes an extra step that scales the magnitude of the denominator term. Simi and Talukder describe the following method [79]. Given an LFP, as shown in Eq. (5.23):

$$F(x)=\frac{c^Tx+\gamma}{d^Tx+\beta} \tag{5.23}$$

A feasible solution to the problem, x^* is chosen, and the LFP is evaluated at this point to yield a valid function value F^*.

$$F^*=F(x^*)=\frac{c^Tx^*+\gamma}{d^Tx^*+\beta} \tag{5.24}$$

The LP problem can then be defined as:

$$\phi(x)=\left(c^T-F^*d^T\right)x \tag{5.25}$$

The method was implemented and briefly tested with the two-truss sizing optimization problem. However, some changes and assumptions were necessary for this method to be applied to the fractional objective. Namely, it was assumed that each high-order term could be represented by its own new individual linear variable. Furthermore, no method could be found to implement constraints such that these replacement variables would be equal to their original definition, and therefore the problem remained unconstrained. Additionally, the method relies on an assumed feasible solution. For this purpose, the mid-sized solution of the two-truss problem was used. This means that, given the three available options for both of the trusses in the two-truss problem, the areas given in the 'Option 2' column of Table 5.2 were used to calculate the value

of F^*. Lastly, the constant terms γ and β were assumed to be zero, as these terms do not naturally appear in the original fractional objective.

Despite these differences and assumptions, this method seemed to give promising results for the small two-truss problem. Namely, for the two-truss problem, the expected outcome is one that corresponds to choosing the largest possible cross-section for the first truss, while choosing the smallest possible cross-section for the second truss. These choices result in the configuration where each truss is closest to the limit stress, given the available choices in cross-sectional area. Indeed, using the method proposed by Simi and Talukder, these same results are found to have the minimum solution to the converted objective function. Thus this method appeared promising for converting fractional objectives into equivalent non-fractional forms.

However, further testing of the method using the larger four-truss problem yielded poor, and seemingly arbitrary results. This may be due to the assumptions that were necessary to implement the method. Particularly the assumption that every high-order binary variable multiplication can be replaced by a new linear variable, without being able to impose constraints to enforce similar behavior, would be a likely source of errors.

The limited success of this method perhaps indicates that the general idea for this conversion method is good, but that the fractional objective as given for the truss sizing optimization problem is not particularly compatible. Especially when the size of the problem is increased, and so too the number of high-order terms in the fractional objective, the direct conversion method proposed by Simi does not work for this problem, using the previously described assumptions.

Iterative Method by Ajagekar et al. Although the publication by Ajagekar et al. had previously already been encountered, its importance and relevance to the truss sizing optimization problem was perhaps somewhat underestimated. In their work, the authors describe an iterative method for minimizing a fractional objective, specifically with application to quantum annealing in mind [3]. However, the reason for not implementing their method up until this point was that its iterative nature was seen as a large disadvantage compared to the possibility of direct methods existing. However, evidently, the direct methods of converting the fractional objective to an equivalent non-fractional form have thus far yielded poor results. As such, finally, the iterative method demonstrated by Ajagekar et al. was further investigated.

The method proposed by Ajagekar et al. can be adapted to the truss sizing optimization, and is described by the following procedure. Asserting that the original fractional objective F is a function of the binary variables $\mathbf{q}$, the objective function can be written as shown in Eq. (5.26).

$$F(\mathbf{q}) = \frac{N(\mathbf{q})}{D(\mathbf{q})} \tag{5.26}$$

Then, some additional parameters are needed and initialized before the iterative procedure can be started. As such, these initial values are set as shown in Eq. (5.27).

$$\begin{aligned} iter &= 0 \\ \lambda &= 0 \\ obj &= \infty \\ \delta &= 10^{-6} \end{aligned} \tag{5.27}$$

Consequently, the iterative procedure can be started, as shown in Eq. (5.28).

$$\begin{aligned} &1: \textbf{while} \left| \lambda - obj \right| > \delta \\ &2: \quad iter = iter + 1 \\ &3: \quad F_{nf}(\mathbf{q}) = N(\mathbf{q}) - \lambda D(\mathbf{q}) \\ &4: \quad \text{find } \hat{\mathbf{q}} \text{ s.t. } \min\left(F_{nf}(\hat{\mathbf{q}})\right) \\ &5: \quad obj = \lambda \\ &6: \quad \lambda = F(\hat{\mathbf{q}}) \end{aligned} \tag{5.28}$$

It can be seen that, similar to previously attempted methods, an $N - \lambda D$ approach is taken in step 3 to rewrite the objective as a non-fraction. Step 4 can then be performed using the QA, or, for testing purposes, by using brute-force analysis, yielding a solution for $\hat{q}$. The function evaluation that takes place in step 6

can then simply be performed classically using the original fractional objective function F. Due to the iterative nature of the procedure, the process is repeated several times until the difference between the current fractional objective function value and the value in the previous iteration is below a user-defined threshold.

The implementation of the iterative procedure was fairly straightforward, only requiring a few additional parameters and a simple while-loop in order to be tested. The method was tested using the familiar sample problems, being evaluated by brute-force. The results for the iterative solution procedure are shown in Figs. 5.14 to 5.16.

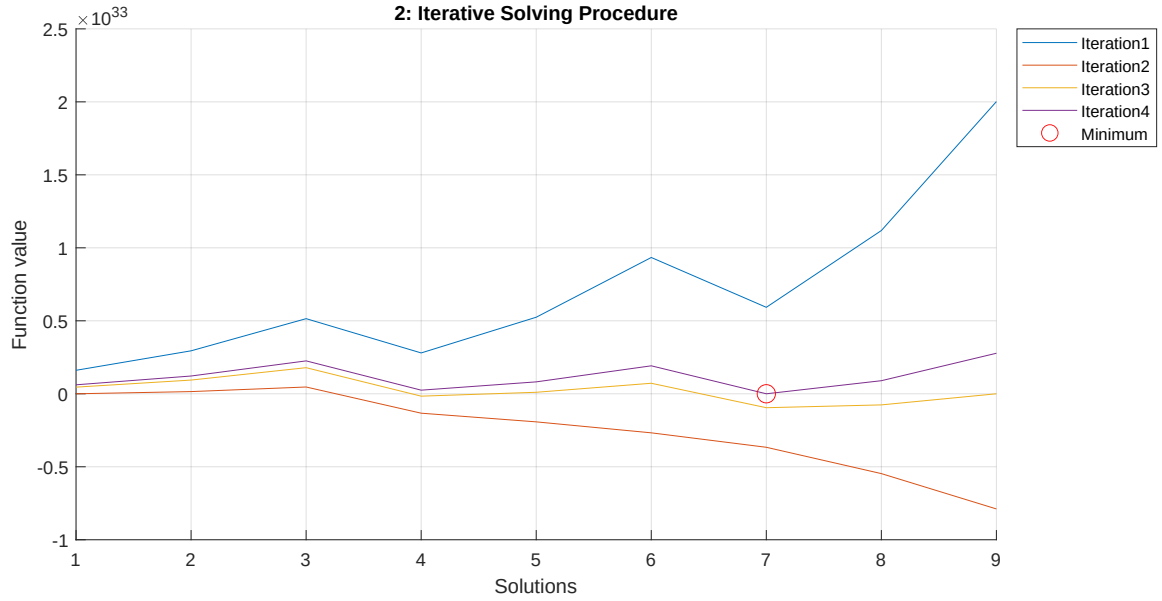


Figure 5.14: Iterative solution procedure for the two-truss problem. Global minimum is found to be solution number 7, corresponding to $[0,0,1,1,0,0]$.

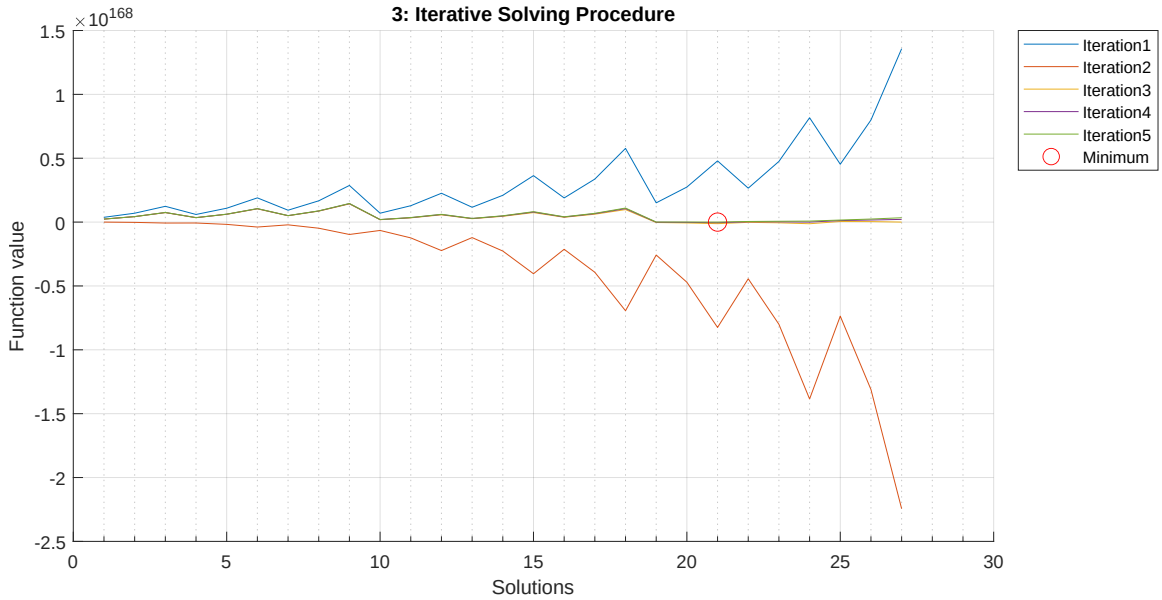


Figure 5.15: Iterative solution procedure for the three-truss problem. Global minimum is found to be solution number 21, corresponding to $[0,0,1,1,0,0,0,0,1]$.

These results can be compared with those of the original fractional objective functions, as were previously shown in Figs. 5.9 to 5.11. It can be seen that the iterative method is consistently able to converge on the same minimum solutions as described by the fractional objective functions. Therefore, this procedure can successfully transform the fractional problem into a non-fractional form. Although several iterations are necessary

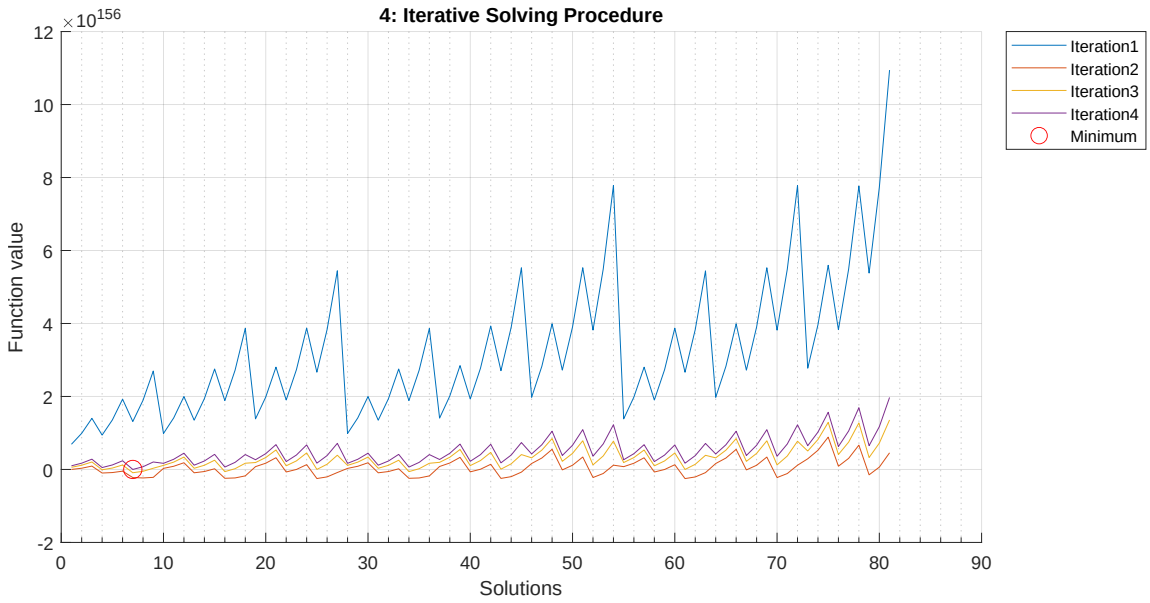


Figure 5.16: Iterative solution procedure for the four-truss problem. Global minimum is found to be solution number 7, corresponding to $[1,0,0,1,0,0,0,0,1,1,0,0]$.

to find the correct solution, the fact that a non-fractional problem is being solved in step 4 of the procedure shown in Eq. (5.28) means that this method is suitable for use with the practical QA. Since, by being non-fractional, it should be possible to finally rewrite this non-fractional expression as a QUBO problem. Thus, another step has been taken towards feasibly solving the truss sizing optimization problem using the actual QA. The practical aspects of getting this procedure to work with the QA, and the testing thereof are discussed in the upcoming section.

5.2.5. Practical Implementation of Truss Sizing Optimization

Based on the previous theoretical work, in which a suitable method was found to define an objective function for the truss sizing optimization problem, as well as finding an iterative method by which the minimum solution to this objective function can be found, the practical implementation of the problem for use with real quantum annealing hardware can be investigated. Thus far, the work done will allow for a fractional objective function to be found for any 2-dimensional system of trusses, having three possible discrete choices in cross-sectional area for every truss. Then, to find the minimum solution of this fractional objective, a rewritten non-fractional function is iteratively solved. This rewritten non-fractional function is nearly, but not quite, suitable for implementation on the QA. In this section, the final steps will be discussed for the practical implementation, and the final results thereof.

Once a problem has been set up in a quantum annealing-compatible QUBO formulation, two different solving methods can be used. The problem can naturally be submitted to the quantum computing hardware to find a solution, however, alternatively, D-Wave offers a simulated annealing (SA) method that can be employed to find solutions without requiring the use of the quantum computing hardware [19]. Simulated annealing is a computationally intensive solving procedure that can be run on local classical computing hardware. The benefit of using SA is that it does not expend the limited amount of quantum computational time that is allotted to basic user accounts on D-Wave's Leap quantum software development platform. This made SA ideal for use throughout the development process for testing code functionality, before committing to using the physical quantum computing hardware.

Problem Size Reduction by High Order Truncation

When it comes to solving problems using the physical quantum annealing hardware or the SA method, it is beneficial if problems can be simplified or reduced in size. This is beneficial because these methods are not guaranteed to yield the global minimum solution, and errors can occur. For quantum annealing this is due to the probabilistic nature of the quantum annealing process. Similarly, the SA implementation that D-Wave offers with their development tools is also not guaranteed to yield the true minimum solution. However,

when the size or complexity of a problem can be decreased, this means the total size of the solution space decreases, and the likelihood of finding high-quality solutions is increased. For this reason, one of the first steps that is undertaken when practically implementing the truss sizing optimization problem for use with the QA is to find ways by which the complexity of the problem can be reduced, while not influencing the minimizing solution.

One key realization at this point in the process was that not all terms in the objective function contribute useful information to valid solutions to the problem. For instance, consider the two-truss problem. This problem, with three different choices of cross-sectional area per truss, utilizes a total of six different qubit variables in its expression for the objective function. As such, the highest-order term in the expression is a sixth-order term, where every variable is multiplied together. However, logically, it is only ever desirable to select a single ideal cross-sectional area for each truss, given the set of possible discrete choices. This means that, for the two-truss problem, valid solutions to the problem will have two binary variables end up with a value of 1, while the remaining four variables should all have a value of 0. In turn, this means that all terms in the objective function that are higher than second order only contribute information to invalid solutions. Since, if only two of the binary variables are expected to be 1, then any term containing three or more variables will *never* yield a contribution to the final objective function value. The first major simplification to the objective function expression is therefore to truncate any term that has a higher order than the number of trusses in the complete structure.

The effect of performing this simplification is that the total number of terms in the objective function is greatly reduced. For example, before truncation, the objective function for the two-truss problem has a total of 63 terms in the expression. This number is not arbitrary, as it is also the total number of possible combinations of six variables, without repetitions, for all possible combinations from first-order up to and including sixth-order. However, truncating every term that is higher than second order, leaves a total of 21 terms, meaning that two-thirds of the terms in the objective function were only contributing information to undesirable and invalid solutions. This effect is even exacerbated for the three- and four-truss problems. In those cases, the total number of terms gets reduced from 511 to 129, and from 4095 to 793 terms respectively. This means that those expressions for the objective functions are reduced in size by around 75% and 80% respectively. For reference, these calculations were performed in a simple Excel sheet, which is available online [91].

Linear Scaling of the Objective Function

Following the simplification of the objective function, a linear scaling of the terms in the objective function can be performed. This means that a constant scaling factor is applied to every term in the objective function. The scaling is chosen such that the maximum absolute magnitude of any of the terms in the function will be equal to a specific, user-defined value. Although the usefulness of this scaling action may not be entirely obvious at this point, it offers an important benefit. Namely, by implementing this linear scaling, a user-defined parameter is created that allows for convenient scaling of the magnitude of the output of the fractional objective function. In turn, this parameter may become useful for manually tuning the importance of the objective function with respect to other properties or constraints related to the practical implementation for quantum annealing. As an example, consider the simple objective function shown in Eq. (5.29).

$$F = 3q_1 + 2q_1q_2 + 5q_2q_3 \tag{5.29}$$

It can be seen that the maximum coefficient in this function has a value of 5, thus $c_{max} = 5$. The function can be scaled such that the user-defined maximum coefficient has a value of c_{user}. As an example, setting $c_{user} = 1$, first the scaling coefficient c_{scale} is calculated as shown in Eq. (5.30). For this example this will be 1/5. Then, to scale the objective function, the full expression for the objective function is multiplied by c_{scale}, as shown in Eq. (5.31).

$$\begin{aligned} c_{scale} &= \frac{c_{user}}{c_{max}} \\ c_{scale} &= \frac{1}{5} \end{aligned} \tag{5.30}$$

$$\begin{aligned} F_{scaled} &= c_{scale}F \\ &= c_{scale}\left(3q_1 + 2q_1q_2 + 5q_2q_3\right) \\ &= \frac{3}{5}q_1 + \frac{2}{5}q_1q_2 + 1q_2q_3 \end{aligned} \tag{5.31}$$

This scaling operation is straightforward in its implementation, as the c_{max} value can be found through simple one-line commands in Python, such as by using the *amax* function present in the Numpy package [63]. It is also possible to perform the scaling separately for both the numerator and the denominator of a fractional objective function, by finding the maximum magnitude c_{max} separately for both parts of the fraction. However, within the iterative solving approach, the scaling is applied to the rewritten non-fractional objective function.

Non-Linear Scaling of the Objective Function

Throughout the brute-force testing of the objective functions for the different sample problems, as discussed in Section 5.2.3, it was seen that certain objective functions have very little difference in function value for the global minimum solution and other local minimum solutions. In practice, this would mean that the global minimum solution is quite difficult to find, as other local minimum solutions have roughly the same function value. In quantum computing terminology, the set of possible function values that can be found for an objective function are commonly referred to as the *energy landscape* for that objective function. This terminology is used in spite of the fact that the objective function is not necessarily written in terms of typical 'energy' units, such as Joules. Nevertheless, the energy landscape invokes a visual intuition for finding globally optimal solutions, as these would be represented by the lowest valley in the energy landscape. The problem where global and local minima have roughly the same function value can therefore be interpreted as the energy landscape having a number of approximately equally deep valleys. Although the QA should be good at finding one of these valleys, it may have difficulty identifying which exact valley is the lowest.

The main cause for very small differences between local minima and the global minimum, is that the specific terms in the objective function that cause these small differences have small coefficients compared to other terms in the function. The method by which these small differences between local and global minima can be amplified is therefore to scale small coefficients in the objective function to become larger and more influential, while not affecting the terms that are already significant. This calls for a *non-linear scaling* method, which scales small coefficients by a large amount, but barely influences larger coefficients. The goal of this scaling is then to increase the difference in the Hamiltonian energy of the global minimum solution and the other local minima. By increasing this difference, it should become easier for the QA to find the global minimum solution.

A Python function was created and implemented that performs this non-linear scaling. By performing this scaling, it may improve the likelihood of identifying the true global minimum using the QA. The method relies on simple user-defined parameters that will allow for manual tweaking once the problem has been fully implemented for use with the QA.

Given the user-defined scaling parameter c_{NL}, a positive coefficient c_{in+} from the objective function is input into the non-linear scaling function. The non-linearly scaled coefficient is then found by Eq. (5.32).

$$c_{out+} = \frac{c_{in+}}{c_{in+} + c_{NL}} \tag{5.32}$$

For negative input coefficients, c_{in-}, the non-linear scaling is performed by:

$$c_{out-} = -\frac{c_{in-}}{c_{in-} - c_{NL}} \tag{5.33}$$

To determine if the input coefficient must be scaled using either Eq. (5.32) or Eq. (5.33), a simple if-statement is used. By setting c_{NL} to be a certain small number, such as 0.002, the amount of scaling that is applied to small coefficients is much more aggressive than for relatively larger coefficients. As a demonstration of this function, several plots are given in Fig. 5.17, showing the influence of changing the parameter c_{NL}. As can be seen, selecting smaller values for c_{NL} leads to more aggressive scaling of small coefficients, but also causes all relatively larger coefficients to become essentially equal. The use of this non-linear scaling function will therefore be a balancing act of increasing the importance of small coefficients, while not losing distinction for the larger terms. The Python code for the non-linear scaling function, and for producing the plot from Fig. 5.17 is available online [91].

To demonstrate the effect of the non-linear scaling on the energy landscapes, a brute-force analysis can be performed. Since the non-linear scaling will be a part of the iterative solving procedure described in the previous section, the energy landscapes of the first iteration of the solving procedure are shown. This first iteration is convenient to investigate, since from Eq. (5.27) it is known that $\lambda = 0$, meaning that the objective function in step 3 in Eq. (5.28) only involves the numerator of the fractional objective function. This numerator is first linearly scaled, choosing the user-defined maximum coefficient to be $c_{user} = 1$. This simply

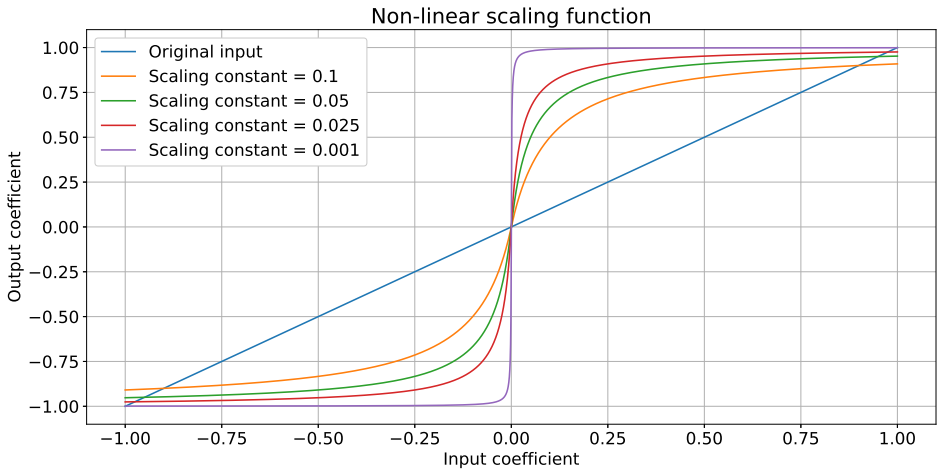


Figure 5.17: Non-linear scaling function.

brings the function values in the energy landscape to a more reasonable magnitude and allows for the non-linear scaling to work as intended. Then, the non-linear scaling can be applied, using a scaling parameter of $c_{NL} = 0.1$. The plots in Figs. 5.18 to 5.20 show both the original and non-linearly scaled energy landscapes of the first iteration in the solving procedure for the sample truss problems. It can be seen that the small fluctuations in the energy landscape are amplified, which should make the problem easier to solve for the QA.

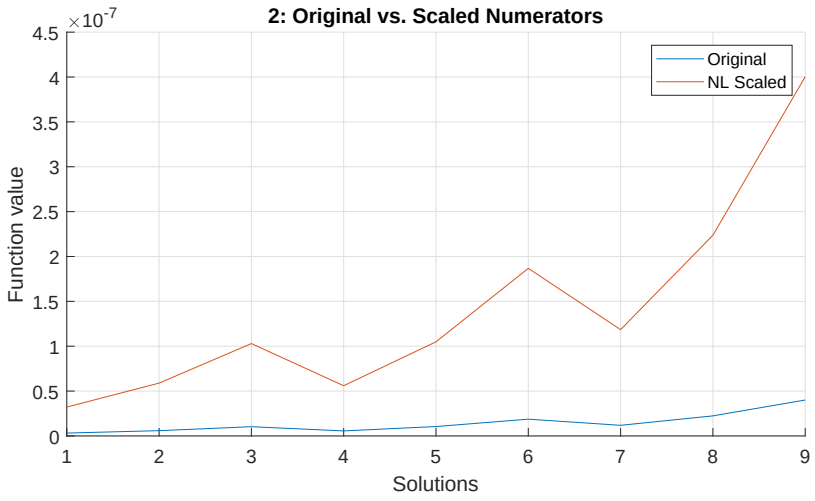


Figure 5.18: Effect of non-linear scaling in two-truss problem.

It is important to note that this non-linear scaling function was developed solely for the purpose of increasing the differences between the global and local minima in the energy landscapes, as a part of this study. It is not intended to be used as a general-purpose tool for problems with unknown energy landscapes. Since, when using overly aggressive scaling factors for c_{NL}, this may cause the global minimum solution to change.

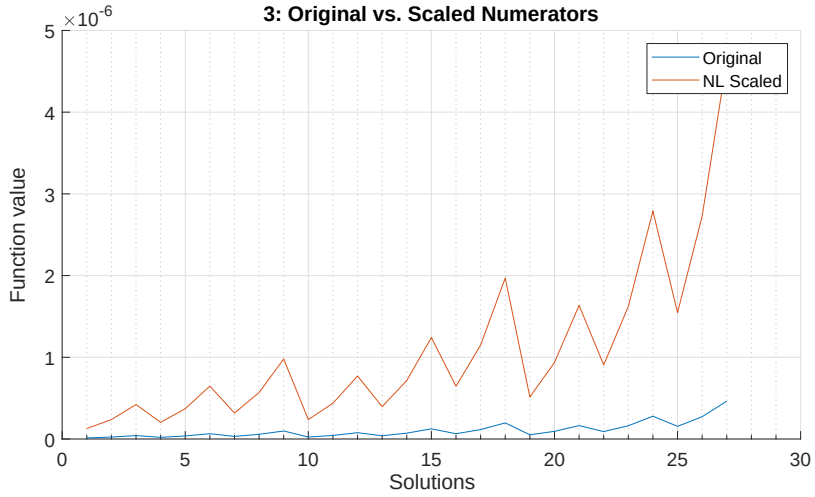


Figure 5.19: Effect of non-linear scaling in three-truss problem.

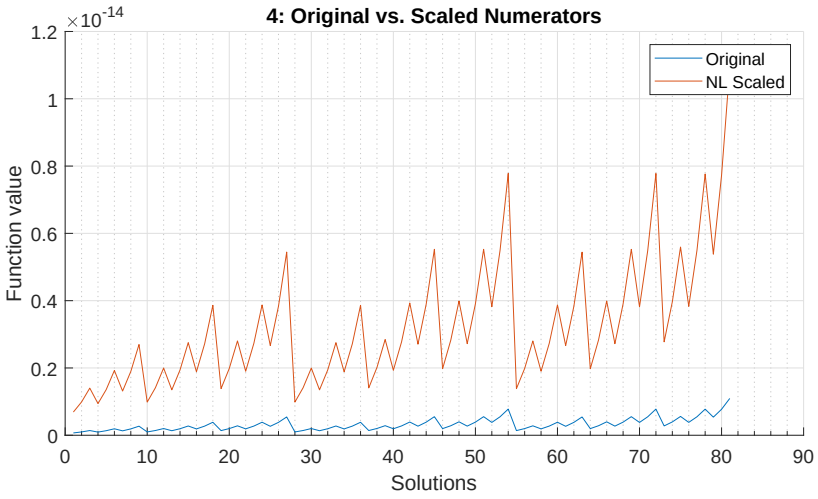


Figure 5.20: Effect of non-linear scaling in four-truss problem.

Problem Size Reduction by Truncation of Insignificant Terms

After applying the linear scaling to make the objective function yield results of a more convenient order of magnitude, and applying the non-linear scaling to help amplify small differences between local and global minima, one final simplification step can be undertaken to reduce the size and complexity of the objective function. This simplification is to truncate terms that fall below a user-defined order of magnitude.

It is known that the physical QA has a certain range wherein qubit biases can be controlled. For linear qubit biases this is a range of [-2,2] and the quadratic qubit biases can be controlled to be in the range of [-1,1]. These allowable bias ranges can be found through the Python commands:

```
DWaveSampler().properties['h_range']
DWaveSampler().properties['j_range']
```

In the case that a problem is submitted for which the biases exceed these allowable ranges, the problem will automatically be scaled by default [27]. Inside the QPU, the qubit biases are controlled through a physical process, relying on small magnetic fields within the QPU [22]. As such, these biases can only be controlled with a finite precision [26]. For example, in practice, the QA is able to control linear qubit biases with an error between about 0.002 and 0.016. For the quadratic qubit biases the control error ranges from about 0.0005 to 0.0075 [23]. Since all terms in the objective function are subject to this control error, this means that the terms that are smaller than this error cannot reliably be taken into account by the QA. Therefore, such small terms can be removed from the objective function altogether to simplify the problem further.

Care should be taken when choosing an order of magnitude for the truncation threshold. If the threshold is relatively large, for example truncating terms that are smaller than 10^{-2}, this may influence *which* solution becomes the most optimal one, as relatively much information will be truncated from the objective function. On the other hand, when choosing a smaller magnitude for truncation, e.g. 10^{-8}, and briefly neglecting the limited physical precision of the QA, it is more likely that the truncated objective function will keep the same optimum solution. However, in trying to take into account this greater precision, the objective function will be more complex. To be conservative, the truncation magnitude can be chosen to be smaller than the precision that the physical QA can take into account.

Unary Constraint

The majority of all possible solutions to the truss optimization problems are invalid. Solutions are invalid when an incorrect number of cross-sectional areas are selected. In other words, when either zero or more than one cross-section is selected for a truss member, the solution is invalid. Solutions can only be valid when exactly one cross-sectional area is chosen for every truss member. For example, the two-truss problem has six qubit variables and therefore has a total of $2^6 = 64$ possible solutions. However, only 9 different valid solutions to this problem exist. To promote the selection of valid solutions, and prevent invalid solutions from being chosen, the *unary constraint* is implemented.

In a previous section of this report, Section 4.2.2, the unary constraint has already been discussed in detail. In this section the description will therefore be rather brief, although still necessary for completeness. The unary constraint is a constraint that will promote the selection of only one cross-sectional area per truss. This is accomplished by adding a penalty function to the objective function such that the Hamiltonian energy for invalid solutions is typically increased, while for valid solutions to the problem the Hamiltonian energy is decreased. The strength of the unary constraint is denoted as λ, and is a user-defined parameter, which can be tuned to meet the needs of the user.

In the context of the truss sizing optimization problem, the unary constraint is required to be a hard constraint. This means that it is strictly forbidden for invalid solutions to be given as the optimal solution to the problem. Hence, the value of λ must be chosen such that the contribution of the unary constraint to the energy landscape is greater than the typical fluctuations between valid and invalid solutions due to the original objective function. To this end, it has been found that a good starting point for the value of λ is to set it to be twice the size of the maximum term in the objective function. However, when tuning the strength of the unary constraint it is also typical to use a trial-and-error approach, by increasing the strength of the constraint until noncompliance stops occurring.

For example, if the maximum term in the objective function has a value of 1, then an initial guess for the strength of the unary constraint would be $\lambda = 2$. In turn, due to how the unary constraint works, this means that the energy of valid solutions gets decreased by 2 for every truss, while the energy for invalid solutions is either left alone or is increased by 6 depending on the severity of the error for every truss. In practice, this means that major jumps will be formed in the energy landscape, ensuring that all valid solutions have much lower energies than any of the invalid solutions.

Quadratization

The last step before being able to solve the truss sizing optimization problem using practical quantum annealing, or SA for that matter, is to perform a *quadratization* of the objective function. Up until this point, the objective function has been manipulated, scaled, and truncated in order for it to become more compatible with the QUBO problem formulation. However, one key issue has yet to be solved: the function might still contain many terms that are greater than quadratic order. Therefore, it can still not be used with the QA, since per definition the QA can only solve quadratic problems. Performing a quadratization of a high-order objective function ensures that it is rewritten as a quadratic-order function, with equivalent solutions.

There are many different methods of quadratization discussed in literature, an extensive overview of which is given by Dattani [29]. Some of these methods utilize auxiliary variables to rewrite the high-order objective function into an equivalent quadratic-order expression, while other methods are able to do so without the need for auxiliary variables. Each method has its respective benefits and drawbacks. For example, it is convenient when no auxiliary variables are needed, yet in that case it may require much effort to rewrite the objective function in a quadratic form. On the other hand, if a method uses auxiliary variables, it may be easier to write in a quadratic form, but the additional variables increase the complexity of the objective function, making it more difficult to find the optimum [29].

In practice, for the truss sizing optimization problem, the most straightforward quadratization solution is to simply rely on the implementation that is provided by D-Wave [20]. In their implementation, all high-order terms are rewritten and replaced to be in terms of auxiliary variables, such that the final problem is at most of quadratic order. An additional user-defined parameter is used to select the strength with which the quadratization is enforced [20]. If the quadratization is not enforced correctly, this can result in a poor approximation of the original high-order objective function. The quadratization strength is problem-dependent and must be tuned such that the quadratization is always obeyed, to have an accurate representation of the original high-order objective function.

The quadratization method provided by D-Wave does have a drawback. Depending on how many auxiliary variables are needed to quadratize the problem, the complexity of the problem can skyrocket. This is one of the reasons why it is beneficial to simplify the objective function as much as possible before performing the quadratization. Applying this to the known sample problems, the two-truss problem does not need to be quadratized since it was previously already truncated to the second order. However, the three-truss problem has 9 variables and was truncated to the third order. Therefore, the quadratization is necessary to rewrite the problem into a quadratic form, growing the problem size to a total of 25 variables. Similarly, quadratizing the four-truss problem increases to problem size from the original 12 binary variables to a total of 81 variables.

With the quadratization, the truss sizing optimization is finally written as a QUBO problem. This vital step finally concludes all of the preparatory work that was necessary for the problem to be compatible with the quantum computing hardware. The next section will discuss the solution process and show results that are obtained.

5.3. Phase 3: Solving the QUBO Problem

With the conclusion of phase 2, the truss sizing optimization problem is now finally written in a QUBO formulation and can be solved using the QA. Now, in phase 3, the final steps for solving the sample problems are discussed. First, the analysis procedures will be discussed in more detail. Then, because there are a number of user-defined parameters relevant to the analyses, the tuning process is discussed for selecting values for these parameters. Once satisfactory values are chosen for the user-defined parameters, the results for the sample problems are shown and discussed.

5.3.1. Analysis Procedures

To solve the reference truss problems, three different analysis methods are applied. First, brute-force evaluation of the original fractional objective function is used to obtain a baseline solution. Then, both SA and quantum annealing are used, following the iterative procedure described in [3]. The specifications of the local classical computing hardware used are given in Table 5.5.

The comparison between the analysis methods will focus primarily on the computational time and the probability of obtaining the global optimum solution. To this end, the brute-force analysis will function simply to obtain the reference solution, since finding the global optimum is guaranteed. There are other more efficient (and more complicated) classical analysis methods available for truss optimization problems, as reviewed by Stolpe [82]. However, for the purposes of this study, simple brute-force analysis is sufficient to

Device	Lenovo Legion Y540-15IRH
CPU	Intel Core i7-9750H 2.6 GHz
Memory	16 GB DDR4 2667 MHz
GPU	Mobile NVIDIA RTX 2060 6GB

Table 5.5: Classical computing hardware.

produce the reference solutions. Since there is some variance in the amount of time needed to complete the analyses, they will all be performed multiple times. The brute-force analyses are each performed three times, since this classical approach performs rather consistently.

Although SA is an entirely different optimization algorithm compared to quantum annealing, SA will be used to verify the functioning of the iterative solving process. Furthermore, by using both SA and the QA, the probability of obtaining the global minimum solutions using both methods can be compared. Thus, to gain some insight into how well these analysis methods are able to find the global minimum solution, each analysis will be performed a total of 10 times. The choice to perform each of the SA and QA analyses 10 times was based on the fact that a limited amount of quantum computational time was available.

One parameter that will be seen to play a large role in the performance of the SA and quantum annealing analyses is the so-called *number of reads*. Once a problem has been submitted to be solved, this number indicates how many times that problem will be solved, before yielding the final lowest-energy solution that has been found. Thus, by increasing the number of reads, the computational time will increase significantly. However, because both SA and quantum annealing are probabilistic solving methods, increasing the number of reads also increases the likelihood that the global minimum solution will be found. For this reason, several different settings for the number of reads will be used, to more broadly assess the performance and usability of these methods.

5.3.2. Parameter Tuning

It has already been mentioned before, but SA is a convenient method for solving QUBO problems, that does not rely on the quantum computing hardware. The main benefit of this method is therefore that it does not use the limited amount of computational time that is allotted to basic user accounts on the D-Wave quantum computing platform. Since several important user-defined parameters influence how the problem is solved, for example the strength of the unary constraint, SA allows for appropriate initial settings for these parameters to be found.

When it comes to fine-tuning the performance of the QA, there are also several specific settings that do not apply to simulated annealing. These would be the settings that directly control how the quantum annealing hardware performs its task. Examples are the *chain strength*, *annealing time*, *annealing schedule*, and depending on if *reverse annealing* is used, the *reverse fraction*. Naturally, there are many more settings that allow for even greater control over the behavior of the QA, however, in this project these are left at their default configurations. The full list of options and the descriptions thereof is provided by D-Wave [27].

To fully define the analysis procedure, values related to the iterative solving procedure were first chosen. The maximum number of iterations within one solving attempt was set to 15. It was seen that by brute-force, the analyses converge in around 5 iterations. For the SA and quantum annealing analyses this value was tripled to give some room for potential errors to be corrected by the procedure. However, if the procedure does not converge after 15 iterations, it is stopped, to prevent excessive expending of computational time[1]. The convergence threshold δ for the iterative analysis (used in Eqs. (5.27) and (5.28)) is set at a value of 10^{-6}, which is the same value used by Ajagekar et al.

From initial trial analyses using SA, starting values for the different problem parameters were found. First of all, it was determined that performing a total of 256 reads per iteration gives consistent results for all of the sample problems. Performing fewer reads is less computationally expensive, and works well for the smaller sample problems. Thus, in all, it was decided to use a total of 16, 64, and 256 reads per iteration for each of the sample problems. Second, it was chosen to set the linear scaling maximum magnitude c_{user} to a value of 1 for all analyses. Third, the non-linear scaling parameter c_{NL} has been set to a value of 0.1 for all analyses. It was seen that this value improves the distinction between minima in the energy landscapes for the sample problems, while preserving the global minima. Fourth, the unary constraint strength was set to a value of 10,

[1] Due to a slight programming error, setting the maximum number of iterations to 15, leads to a total maximum of 16 iterations being performed for the simulated annealing and quantum annealing analyses.

with which the constraint is obeyed consistently. Fifth, terms that have a magnitude smaller than 10^{-8} are truncated, to slightly reduce the number of terms in the objective functions. This is a conservative truncation, as it is well beyond the precision of the quantum annealing hardware capabilities. However, setting a much less conservative truncation magnitude, such as 10^{-3}, could potentially lead to changes in the global minimum solution, which should be avoided. Finally, the quadratization strength is initially also set to a value of 10, to equally match the strength of the unary constraint.

For the QA, some additional parameters are needed to solve the sample problems. Based on preliminary testing, the anneal time was left at its default value of 20 μs, as no improvements to performance were observed by using larger values, while consuming more QPU access time. Furthermore, reverse annealing also did not appear to have a beneficial influence on the performance of the QA. Due to this, and for the simplicity of the analysis procedures, it was chosen to only use the default forward annealing approach for the truss sizing optimization problems. Initially, a chain strength of 10 was selected, but it was seen that chain breaks would still occur for the three- and four-truss problems. For those problems a chain strength of 30 performed better, preventing chain breaks from occurring. For specifically the four-truss problem, it was also observed that the unary constraint was not always obeyed when using the QA. Hence, the unary constraint, and correspondingly the quadratization strength was increased to a value of 20. The number of reads used for the quantum annealing analyses is tailored to the expected performance for the specific sample problems, to prevent excessive spending of QPU access time. Thus, for the two-truss problem the analysis is performed using 16 and 64 reads per iteration. For the three-truss problem this is increased, performing the analysis using 64 and 256 reads per iteration. Lastly, for the four-truss problem, only a setting of 256 reads is used, since poor results are expected when using lower settings.

In Table 5.6 a summary is given of all parameters used for each of the analyses. Parameters that are irrelevant or not applicable for an analysis are indicated by *NA*.

Parameters	Brute-force			Simulated Annealing			Quantum Annealing		
Truss system	2-truss	3-truss	4-truss	2-truss	3-truss	4-truss	2-truss	3-truss	4-truss
Total number of times analyzed	3	3	3	10	10	10	10	10	10
Maximum number of iterations	NA	NA	NA	15	15	15	15	15	15
Iteration convergence threshold	NA	NA	NA	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Number of reads per iteration	NA	NA	NA	{16, 64, 256}	{16, 64, 256}	{16, 64, 256}	{16, 64}	{64, 256}	{256}
Highest order terms allowed	NA	NA	NA	2	3	4	2	3	4
Linear scaling maximum magnitude	NA	NA	NA	1	1	1	1	1	1
Non-linear scaling strength	NA	NA	NA	0.1	0.1	0.1	0.1	0.1	0.1
Unary constraint strength	NA	NA	NA	10	10	10	10	10	20
Quadratization strength	NA	NA	NA	10	10	10	10	10	20
Precision truncation magnitude	NA	NA	NA	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
Chain strength	NA	NA	NA	NA	NA	NA	10	30	30
Annealing time $[\mu s]$	NA	NA	NA	NA	NA	NA	20	20	20

Table 5.6: Parameters used for all analyses.

5.3.3. Results: Two-Truss Problem

The two-truss problem is the simplest of the three sample problems. To set up the fractional objective function for the problem, the procedure is followed that is described throughout the previous sections of this chapter. For this problem, it takes approximately 13.5 seconds to set up the fractional objective function. Once this objective function has been found, it is written to a text file. This file can then imported and interpreted before performing every analysis. When running the same problem many times, the same text file can simply be imported each time, and no longer needs to be set up from scratch. In total, six different analyses were run for this problem:

1. BF2_F_V: Brute-force analysis of the fractional objective function, for valid solutions.
2. SA2_16: Simulated annealing, with num_reads = 16.
3. SA2_64: Simulated annealing, with num_reads = 64.

4. SA2_256: Simulated annealing, with num_reads = 256.
5. QA2_16: Quantum annealing, with num_reads = 16.
6. QA2_64: Quantum annealing, with num_reads = 64.

For each of the SA and QA analyses, two different time-metrics are provided. First, the total real time (RT) is given, which is the time it took to analyze the problem and includes some sources of overhead. Second, the pure solve time (ST) is measured, which for SA is the time spent purely performing the simulated annealing task, and for quantum annealing is the amount of QPU access time needed to solve the problem. The brute-force analysis was performed three times, while the SA and QA analyses were each performed ten times. Based on these analyses, mean run-times and standard deviations were calculated, and are shown in Fig. 5.21.

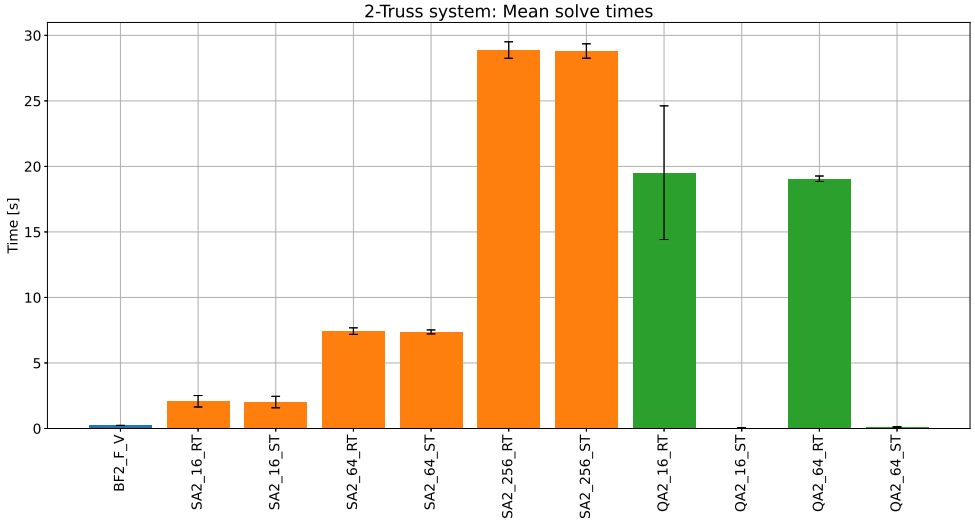


Figure 5.21: Mean analysis times for two-truss problem.

Using both SA and the QA, the analyses were performed ten times, to gain some insight into the probability of obtaining the globally optimal solution. In Figs. 5.22 and 5.23 the probability of obtaining valid solutions is shown as a bar chart, with the brute-force line plot of the original fractional objective function superimposed to show the energy landscape. The global minimum solution of the fractional objective function is located in the plot at solution number 7.

Discussion From the probability charts for the SA and QA analyses, Figs. 5.22 and 5.23, it can be seen that for both methods each of the ten runs resulted in the same global optimum that is found by brute-force. The solution that is found in all cases is $[0,0,1,1,0,0]$, which indicates that the optimal choices for the truss cross-sections are the largest possible choice for the first truss, and the smallest possible choice for the second truss. Furthermore, it can be seen that this result is returned even when only 16 reads are performed for every step in the iterative solving process for both SA and QA methods. The apparently high probability of finding the correct result, as well as the low number of reads per iteration that is needed, indicate that the problem is easy to solve.

When it comes to the amount of computational time needed to perform the analyses, as indicated in Fig. 5.21, it can be seen that the brute-force analysis of the original fractional objective function, limited to the set of valid solutions, runs extremely quickly. In this case it took on average only about 0.23 seconds to solve the problem. For the QA it can be seen that the total real analysis time is more consistent using 64 reads, compared to using only 16 reads, as the standard deviation is much smaller. The amount of QPU access time needed to solve the problem is extremely low, appearing to be virtually non-existent in the chart. On average, the QPU access time to solve the problem was 59176 μs and 118207 μs using 16 and 64 reads respectively. This is faster than the brute-force analysis, however, the real total amount of time to solve the problem is dominated by various sources of overhead, making the procedure much slower in general.

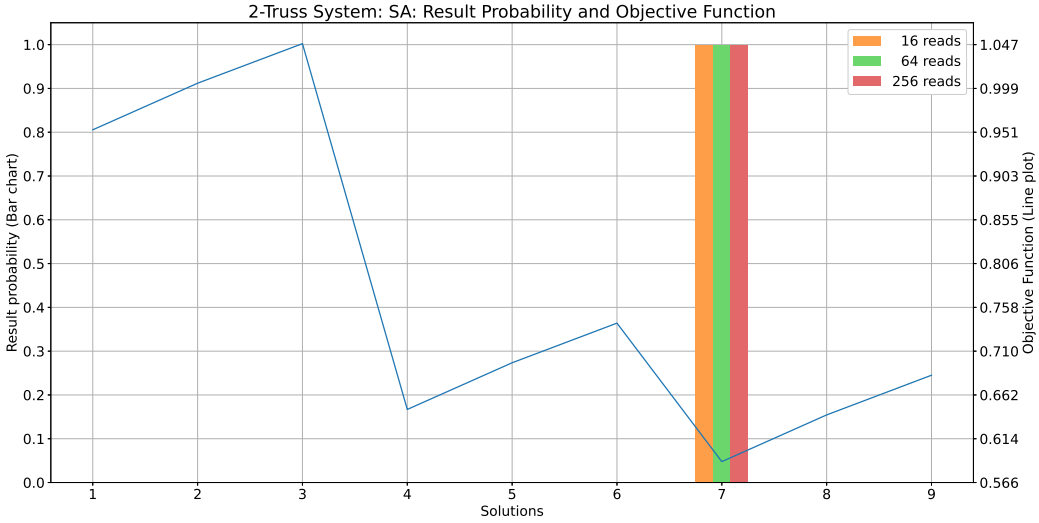


Figure 5.22: Solution probability histogram of two-truss problem using SA, compared to original objective function.

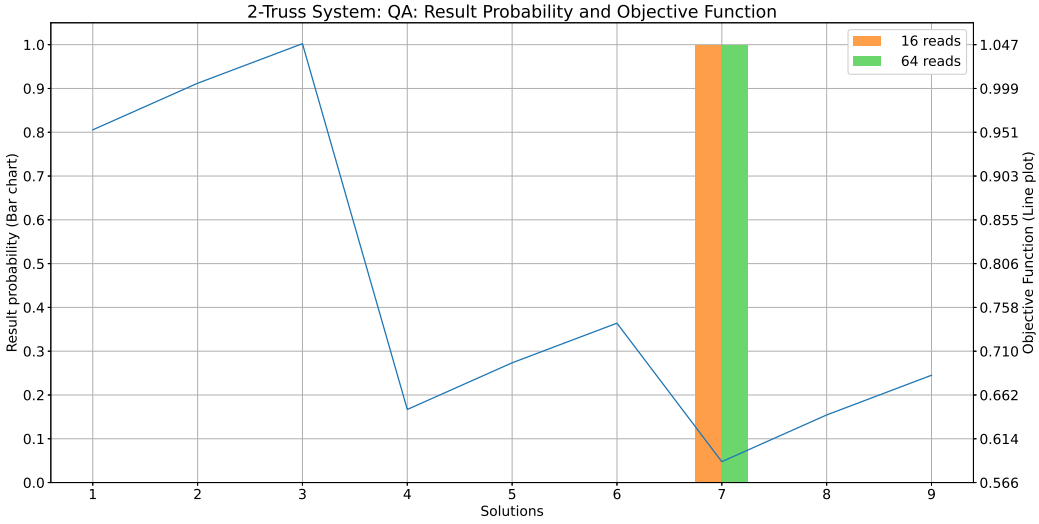


Figure 5.23: Solution probability histogram of two-truss problem using the QA, compared to original objective function.

5.3.4. Results: Three-Truss Problem

The three-truss problem is a more complicated problem to set up compared to the two-truss problem, due to the increased number of variables. Setting up the fractional objective function for this problem, and writing that expression to a text file took approximately 81.4 seconds. However, once the expression was written to a file, it could simply be imported and reused for every analysis. Similar to the two-truss problem, the three-truss problem was analyzed using six different procedures in total. The procedures are:

1. BF3_F_V: Brute-force analysis of the fractional objective function, for valid solutions.
2. SA3_16: Simulated annealing, with num_reads = 16.
3. SA3_64: Simulated annealing, with num_reads = 64.
4. SA3_256: Simulated annealing, with num_reads = 256.
5. QA3_64: Quantum annealing, with num_reads = 64.
6. QA3_256: Quantum annealing, with num_reads = 256.

Again, the brute-force analysis was performed three times, while the SA and QA analyses were run ten times each. In Fig. 5.24 the mean analysis times can be seen, including the corresponding standard deviations.

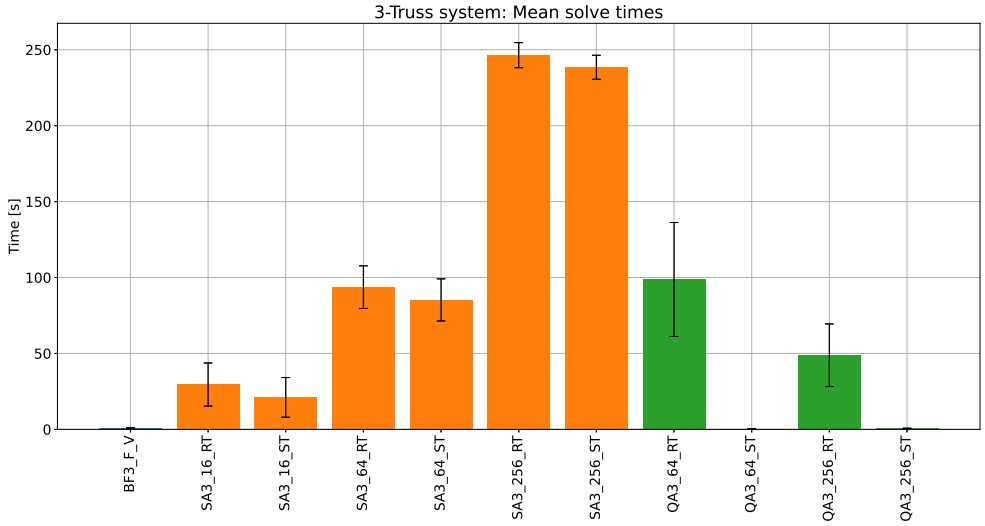


Figure 5.24: Mean analysis times for three-truss problem.

In Fig. 5.25 and Fig. 5.26 the solution probability histogram is shown for both the SA and QA analyses, including the original fractional objective function. The global minimum solution of the original objective function is located at solution number 21 for the three-truss problem. Since this problem is more complicated compared to the two-truss problem, sometimes invalid final solutions are found. These are indicated in the solution probability histogram, in the final column on the right-hand side, above which *NV* (non-valid) is written.

Discussion From the probability histograms it can be seen that the problem is more difficult to solve compared to the two-truss problem, as the histograms show that a spread of different solutions were found. When using SA, the global optimum solution was returned for every attempt when using 256 reads per iteration. Using 64 reads per iteration however is seen to provide very good solutions also, as it tends to give the global optimal solution 21 as well as the nearly-optimal solutions 19 and 20. When using quantum annealing, the majority of solutions are either optimal or extremely close to optimal when using 256 reads per iteration.

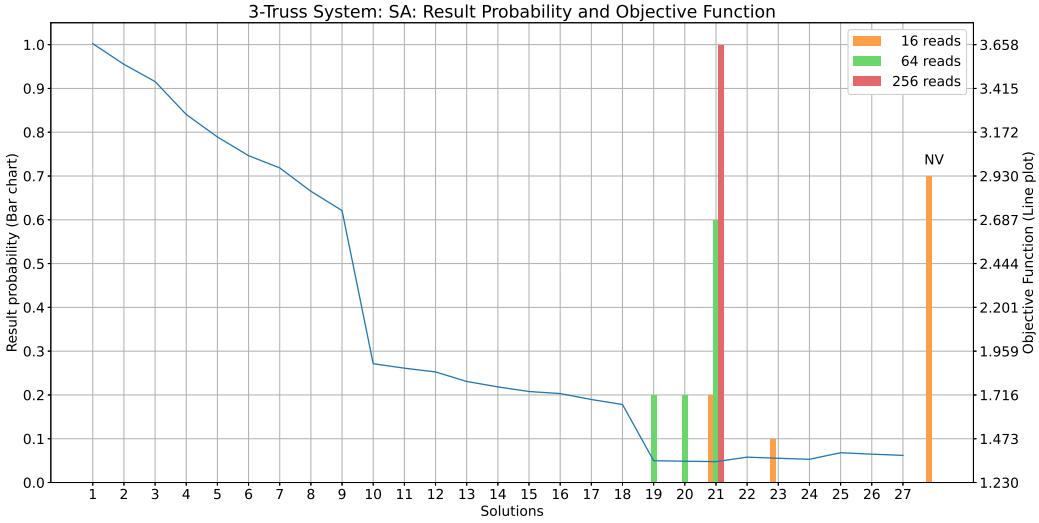


Figure 5.25: Solution probability histogram of three-truss problem using SA, compared to original objective function.

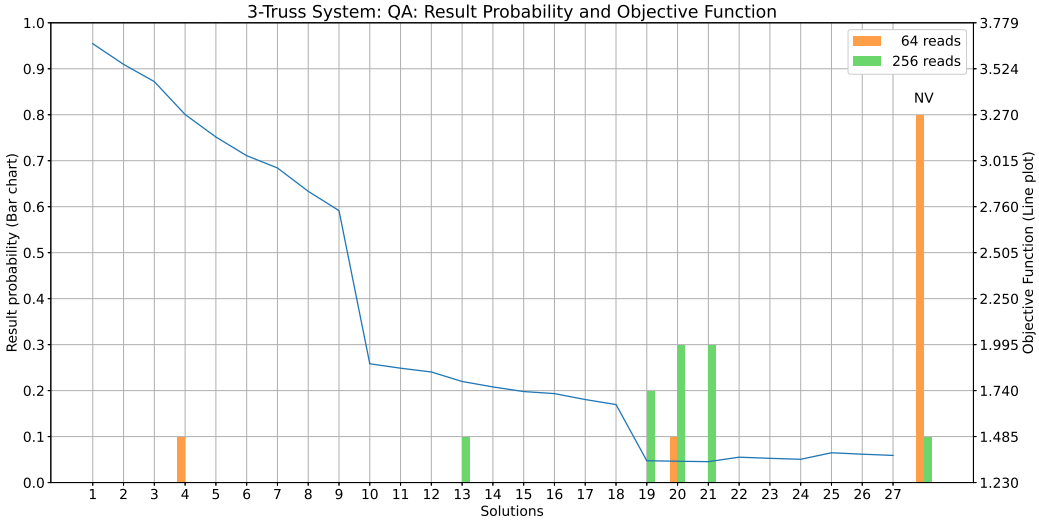


Figure 5.26: Solution probability histogram of three-truss problem using the QA, compared to original objective function.

When using 64 reads quantum annealing performs rather poorly, with 8 out of 10 analyses yielding a non-valid result.

The computational time required to solve the three-truss problem by brute-force is about 0.9 seconds. By contrast, the SA and QA analyses take between about 25 to about 250 seconds of real analysis time to solve, depending on the number of reads. However, an interesting phenomenon can be seen for the quantum annealing analysis times. Namely, when using 256 reads per iteration, the mean real analysis time is almost half of that when using only 64 reads. This can be explained by the fact that when using a higher number of reads, it becomes more likely that the correct solution is found for every iteration of the procedure. This means that on average fewer iterations are needed before the solution algorithm converges. However, even though the amount of real time needed to solve the problem is much lower when using 256 reads of quantum annealing, the actual QPU access time to solve the problem increases from 337477 μs when using 64 reads to 558414 μs when using 256 reads. Discounting the sources of overhead, this means that using quantum annealing with 256 reads per iteration is a faster method of 'solving' the problem than when using brute-force analysis. However, this forsakes the guarantee of finding the global optimum solution. In reality the method takes much longer due to the overhead from various programming steps, calculating an embedding for the problem, network lag from problem submission, and solution retrieval from the D-Wave server.

5.3.5. Results: Four-Truss Problem

The last problem, with the highest number of variables involved, is the four-truss problem. The initial setup of the fractional objective function in this case took up a total of 3430.5 seconds, which is a staggering increase from the 81.4 seconds that were necessary to set up the three-truss problem. In this case, it is a great benefit that the fractional objective function is written to a text file, which is in turn imported and interpreted before running every analysis. Were this not the case, it would mean that every analysis would be prefaced by a nearly hour-long setup process, greatly slowing down the testing and analysis process. Overall, the following analyses were performed:

1. BF4_F_V: Brute-force analysis of the fractional objective function, for valid solutions.
2. SA4_16: Simulated annealing, with num_reads = 16.
3. SA4_64: Simulated annealing, with num_reads = 64.
4. SA4_256: Simulated annealing, with num_reads = 256.
5. QA4_256: Quantum annealing, with num_reads = 256.

In this case quantum annealing was only used with a setting of 256 reads per iteration, as it was deemed improbable that using fewer reads would lead to good results, and would only serve to fruitlessly expend the limited amount of QPU access time available. Brief testing with 1024 reads did not appear to give more reliable results, while expending far more valuable QPU access time. Hence, when using the QA, the decision was made to only investigate this problem using 256 reads per iteration.

As with the previous sample problems, the brute-force analysis was performed three times, while the SA and QA analyses were all performed ten times each. The mean analysis times for all analyses can be seen in Fig. 5.27. The solution probability histograms for the SA and QA analyses are shown in Figs. 5.28 and 5.29, including a plot of the original fractional objective function, for which the global optimum solution exists at solution number 7.

Discussion It can be seen that the problem is reliably solved using 256 reads per iteration when using SA. With a setting of 64 reads, performance appears to be decent, finding the global optimum most of the time, and sporadically yielding other local minima. However, when it comes to quantum annealing, even when using 256 reads, seemingly random yet valid final results are produced. None of the quantum annealing analyses found the global optimum solution, and only some of the solutions that were found can be considered to be local minima. It could still be the case that a higher number of reads would lead to more consistent final solutions, however, this could not be extensively tested due to the large amount of QPU access time required.

For quantum annealing, an issue that could be the cause of the seemingly random final solutions is that most attempts at solving the problem (7 of the 10 total) ended with an iteration run-out, rather than ending with a converged final solution. This issue could potentially also be solved with a larger number of reads per iteration. A single test-run was performed using 1024 reads per iteration. However, this test still ended

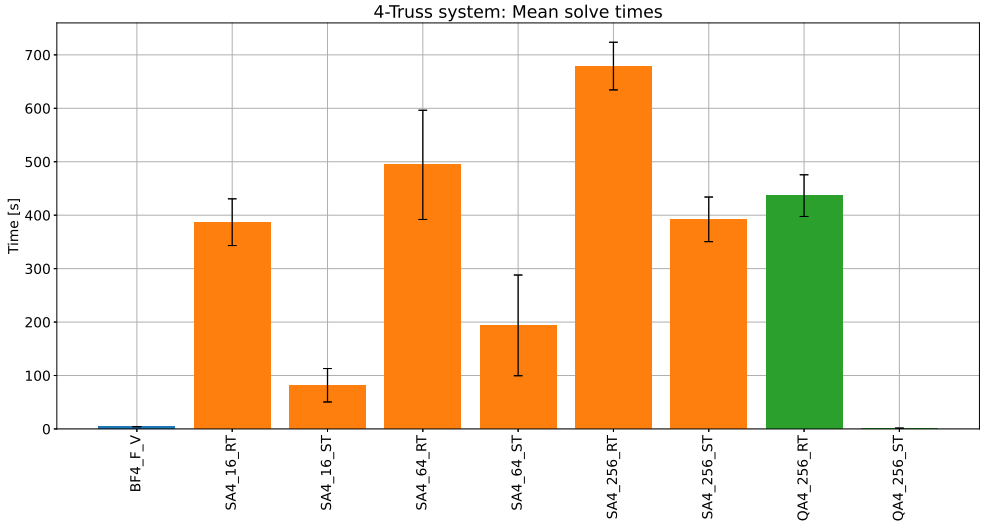


Figure 5.27: Mean analysis times for four-truss problem.

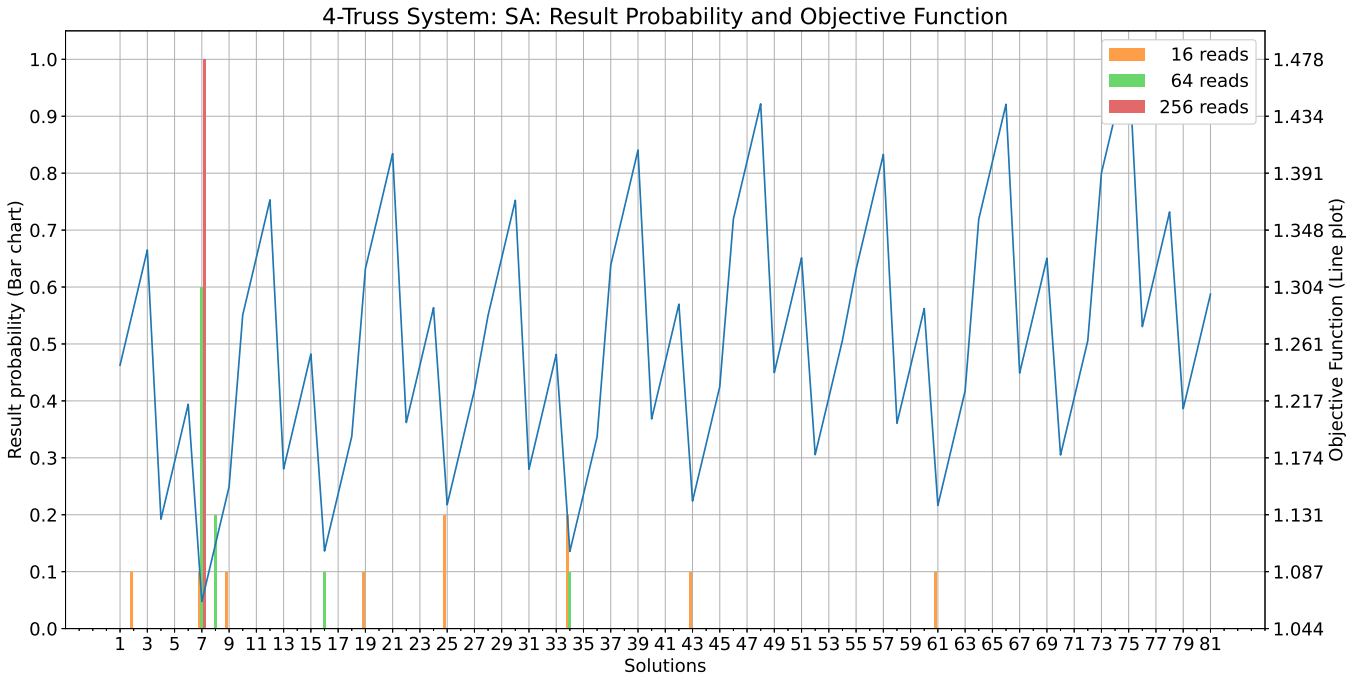


Figure 5.28: Solution probability histogram of four-truss problem using SA, compared to original objective function.

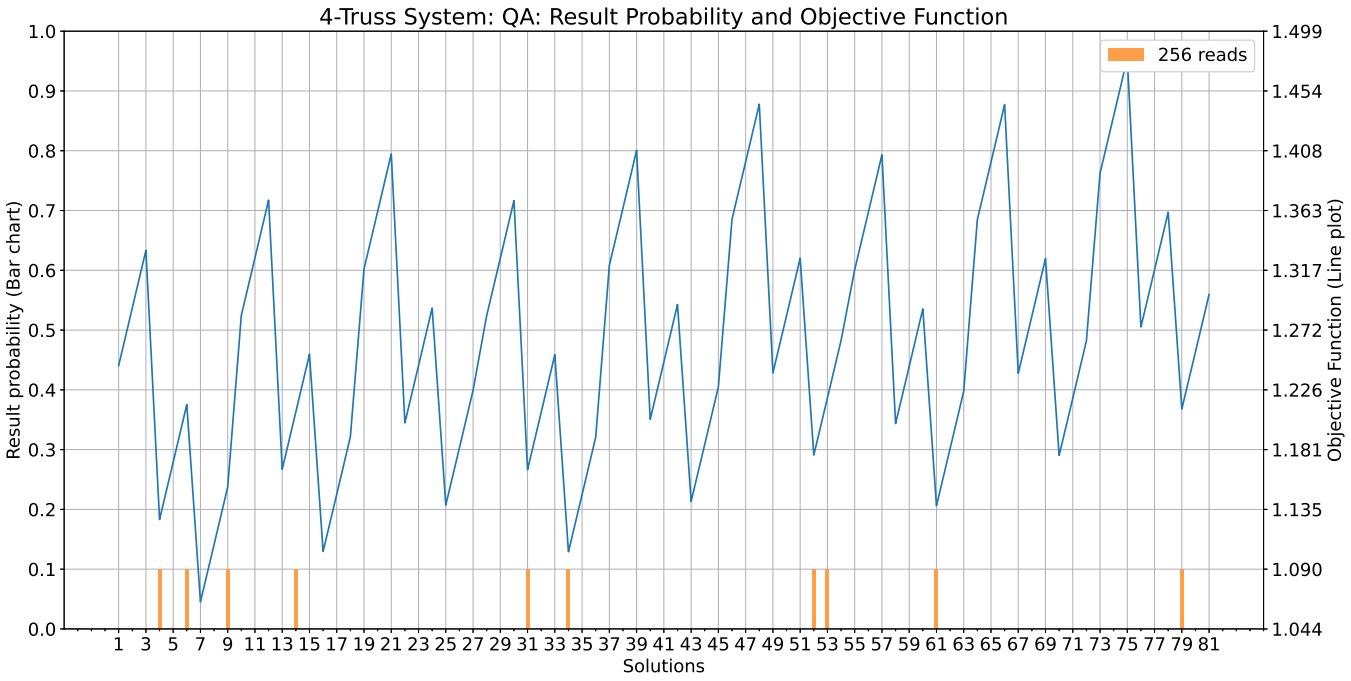


Figure 5.29: Solution probability histogram of four-truss problem using the QA, compared to original objective function.

with an iteration run-out and a non-optimal solution, while requiring a tremendous total of 5312595 μs of QPU access time. Although a single analysis is not enough to calculate any meaningful statistics or form any definitive conclusions, it indicates that the problem may simply be too difficult for the QA to find a consistent solution.

When brute-force evaluating the fractional objective function for the four-truss problem, the mean time to solve the problem is only 4.37 seconds. In comparison, the SA and QA real analysis times range from approximately 380 seconds to about 675 seconds depending on the number of reads. Of further note are the large differences between the real analysis time and the actual solve time for the SA analyses. For each of these analyses, there is roughly a 300 second difference between the real analysis time, and the actual solve time. This difference is mostly due to programming overhead related to the interpretation of the original fractional objective function from the source text file. This source of overhead also plays a role in the quantum annealing analysis, explaining the majority of the difference between the real analysis time, and the QPU access time.

The quantum annealing QPU access time is on average 1280156 μs, meaning that this is the fastest method to 'solve' the problem. However, practically, the accumulated overhead throughout the process leads to a real analysis time of about 437 seconds on average. Furthermore, it was seen that the QA was not able to consistently yield the global minimum result. It could still be the case that using a much higher number of reads for the quantum annealing analysis would yield more consistent results. However, this could unfortunately not be extensively tested due to QPU access limitations.

5.4. Final Discussion

Having investigated the three sample problems, a number of final observations can be made based on the data that was produced. In this section these will be given.

Problem Setup Before any of the analyses can be performed, the fractional objective function that describes the truss sizing optimization problem must be set up. This setup procedure is performed using the MATLAB software and is finished when the symbolic fractional objective function is written to a text file. Once the fractional objective function has been found, the analysis procedure can continue within Python, where

the function is imported and further processed in order to finally be written in a QUBO-compatible format. The time to set up the fractional objective function was measured once for each of the sample problems. Although already reported in the previous section, the setup times are reiterated in Table 5.7, also showing the factor with which the setup time increases for each subsequent problem.

The initial setup time is reasonable for the two-truss problem. However, it seemingly becomes exponentially worse for larger truss systems. Using the Microsoft Excel software package, an exponential trendline for the problem setup time can be fitted to the data. This is shown in Fig. 5.30. It can be seen that the exponential curve can be fitted with an R-squared value of $R^2 = 0.96$, which indicates a good fit. From this trendline it can be approximated that setting up the fractional objective function for a five-truss problem would take $T_5 = 0.0384 \cdot \exp(2.7687 \cdot 5) = 39490$ seconds, or about 11 hours. Evidently, it becomes extremely impractical to generate the fractional objective function for larger truss systems. Thus, this severely limits the practical usability of the method that has been developed for the setup of the fractional objective functions.

Truss System	Variables	Setup Time [s]	Growth Factor [-]
2	6	13.504	-
3	9	81.390	6.0270
4	12	3430.542	42.1494

Table 5.7: Overview of objective function setup time.

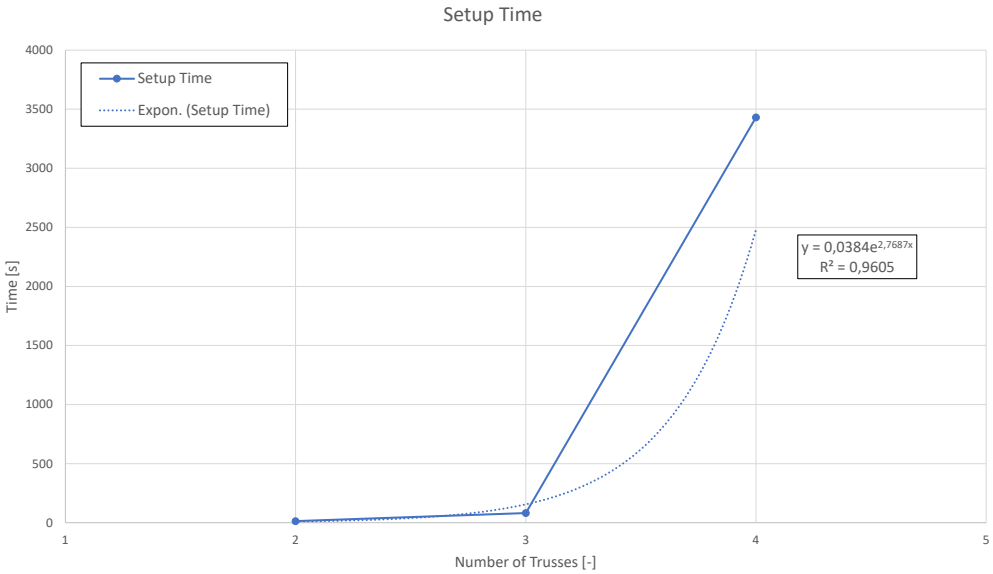


Figure 5.30: Problem setup time and exponential trendline.

Chaining Overall, the fractional objective functions for the truss sizing optimization problems represent fully, or nearly-fully connected problems. This means that (nearly) all possible unique multiplications of the binary problem variables are present in the problem. For the QA this means that every logical variable must be embedded in such a way that it can form a connection with every other logical variable. Through the process of embedding, logical variables are turned into physical variables, where a single logical variable is represented by a chain of physical qubits. When problems necessitate high variable connectivity, this automatically means that long qubit chains will be used to describe these variables. Since, the physical layout of the qubits in the QA has at most, on the Chimera architecture, a total of 6 connections to nearby qubits. By using chains of physical qubits to represent the logical variables, the maximum connectivity between the logical variables can be increased.

However, due to the fact that such high connectivity is required for the truss optimization problems, this is an indication that the problem is not particularly well suited for solving on the QA. Although the connectivity limitation can be overcome to a limited degree by using chains, a high chain strength must be used to prevent these chains from breaking. In turn, the high chain strength is detrimental to the performance of the QA, as

it reduces the probability of finding the optimal solution to the problem [33]. Because some very long chains are used for the larger truss optimization problems, sometimes up to 14 qubits in length, this indicates poor compatibility between the current generation of quantum annealers and the current method of formulating the truss sizing optimization QUBO. The fact that such long chains are needed might be the root cause of the low consistency and reliability seen for the three- and four-truss problems. The next generation of quantum annealers to be produced by D-Wave will rely on the Pegasus hardware architecture, which will enable up to 15 connections between physical qubits [30]. This would significantly reduce the length of the chains needed to embed the truss optimization problems.

Simulated Annealing Simulated annealing was used to solve the sample problems mostly because the method allowed for code to be tested and fixed without spending the limited amount of computational time available on the D-Wave quantum annealing platform. Observing the behavior compared to quantum annealing, SA with 256 reads per iteration produced the global optimum result in all cases, for all sample problems. As such, the method, although slow for the larger problems, is seen to be much more reliable than quantum annealing. By performing the SA analyses, it is clear that the iterative solving procedure works as intended. Since the same procedure is used for the quantum annealing analyses, the iterative solving method itself is not expected to be the cause of any of the difficulties seen in the quantum annealing analyses. Based on the SA analyses, it can be concluded that the method shown in this chapter for casting the truss sizing optimization problem into a QUBO form is valid and feasible for small problems.

Quantum Annealing Neglecting the various sources overhead, quantum annealing provides a very fast method of finding solutions to the truss sizing optimization problems. Counting just the pure QPU access time, the method tends to find results for the largest sample problem in approximately 1.3 seconds, and is even faster for the smaller sample problems. This means that the pure solve time for quantum annealing is faster than the analysis time using the classical brute-force method. Of course, in a more traditional setting, more efficient classical optimization methods would be used to optimize truss structures [82], but for this study brute-force analysis provided the simplest way to obtain the reference global optimum solutions. However, when taking the overhead into account, quantum annealing is much slower in practice. This overhead could potentially be alleviated by more efficient programming. Furthermore, having on-site unlimited access to a quantum annealing system would also help to significantly reduce the amount of overhead time during the analyses. Therefore, based on the computational time needed to solve problems, quantum annealing has the potential to become a competitive problem-solving method.

When weighing the value of the quantum annealing technology for the truss sizing optimization problem, the quality of the results that it produces must also be taken into account. It was seen that quantum annealing provides consistent optimal results for the smallest of the sample problems. However, for problems beyond the very smallest, the method has shown difficulty in finding optimal solutions. It is possible that using a greater number of reads per iteration can amend these difficulties, however, this comes at the cost of much increased computational time requirements and could not practically be tested. One of the main causes for this poor reliability is expected to lie in the fact that the three- and four-truss problems have connectivity requirements beyond what the QA natively supports. This leads to complicated embeddings with long chains of qubits representing the individual problem variables, which has a negative impact on the probability of finding the global optimum solution. The next-generation quantum annealing hardware, using the Pegasus architecture, could significantly reduce these problems.

Although it has been shown that it is *feasible* to solve small truss sizing optimization problems using the QA, it is evident that there are currently significant limitations to the practical usability of the methods discussed in this chapter. As such, it can be concluded that it is not currently beneficial to apply quantum annealing to solve these practical truss sizing optimization problems. With further development of quantum annealing technology, and with improvements to the methods that have been shown in this chapter, there is nevertheless the potential for this conclusion to change.

5.5. Chapter Summary

Throughout this (admittedly lengthy) chapter, a symbolic finite-element method was developed with which arbitrary 2-dimensional truss sizing optimization problems could be set up and solved using the QA. The starting point was a relatively simple question, being: *What is the most optimal selection of cross-sectional areas for a system of trusses, given discrete allowable choices?* However, this proved to be quite a complicated question to answer.

Following the symbolic method, the first challenge to be overcome was that of solving symbolically defined finite-element problems. Symbolic expressions for the nodal displacements in the truss system needed to be found, after which these could be further processed to find expressions for the truss stresses and reserve factors. It was eventually found that by using the MATLAB software to perform these steps, instead of using Python, the symbolic finite-element problem could be solved relatively easily.

From this point onwards, the next challenge was to find a functional formulation for an objective function, which would be able to serve as the target for an optimization problem. Through some trial-and-error, an objective function formulation was eventually found which appeared to agree with the expectations an engineer might have for this type of optimization problem. Namely, the minimum solution to these objective functions generally steers the truss cross-sectional areas towards a configuration where the truss stresses approach the material limit stress. This also implies that, through iteration, the minimum weight truss-system would be found for which every truss complies with the material stress requirements.

The unfortunate consequence of the method used to formulate the objective functions is that the method yields functions which are written in a fractional form. These are inherently incompatible with the QUBO problem formulation, since QUBO problems are non-fractional in nature. Thus, this led to the next challenge, which was to find a way to make the objective functions compatible with a QUBO problem formulation. Although some methods to rewrite the fractional objective function directly into a non-fractional form were attempted, none were successful in yielding an equivalent function with the same minimum solutions as the original. The only method that proved to be successful was to use an iterative approach, which is eventually able to converge on the same global minimum solution as is present in the original fractional objective function. Thus, by using this iterative approach, another step is taken towards a formulation for the truss sizing optimization problem which could be solved using the QA.

The final step to make the truss sizing optimization problem compatible with the QA involved simplifications, constraints, and processing of the objective function. While most of the simplifications and processing serve to make the optimization problem easier to solve for the QA, the most important step is to perform a quadratization of the objective function. This ensures that any high-order terms in the function are rewritten in such a way that the objective function contains only linear- and quadratic-order terms. This means that the objective function is finally written as a QUBO problem, and can be submitted to the QA.

The performance of the QA was evaluated using three simple reference truss problems. These problems were solved using brute-force analysis, simulated annealing, and quantum annealing. The brute-force analyses served as a reference for the global optimum solution, as finding the global optimum is guaranteed using this method. In turn, SA was used to verify the functionality of code, and help to fine-tune various parameters that influence how the problem is solved. Naturally, quantum annealing was used to investigate if it is feasible to solve the truss sizing optimization problem using quantum computing techniques.

From the results of the many analyses performed in this chapter it was seen that, in general, the QA has difficulty finding the global optimum solutions for all but the smallest of the sample problems. One of the major contributing factors in this difficulty is the fact that the truss sizing optimization problem requires high qubit connectivity. In the current generation of quantum annealers, the maximum connectivity between qubits allows for only 6 connections between qubits. In turn, this means that long chains of physical qubits are needed to represent each of the individual logical problem variables, which has a negative impact on the probability of finding the global optimum solution [33]. In the next generation of D-Wave quantum annealers, the connectivity between physical qubits will be increased to a maximum of 15. This would help to reduce the length of the qubit chains needed to embed the truss sizing optimization problem, and would likely lead to performance improvements.

Regardless of whether global optimum solutions are found, the QA has been shown to be extremely fast at finding solutions to the truss optimization problems, if sources of overhead are neglected. With more efficient programming and more direct access to the QA, these sources of overhead could be reduced significantly. This means that, overall, the technology shows promise for becoming a competitive problem-solving method. However, as it currently stands, the amount of time overhead present in the truss sizing optimization procedure means that it is much slower to use the QA in practice than it is to simply perform a brute-force

analysis. This fact, along with the poor probability of obtaining the global optimum solution for the larger truss systems, means that it is currently not beneficial to apply quantum annealing to these truss sizing optimization problems. Nevertheless, it has been shown that it is at least feasible, through significant effort, to translate such an engineering optimization problem into a quantum-compatible form. The lessons learned throughout this process may hopefully help future researchers to formulate other interesting and practically-oriented optimization problems for use with the quantum annealer.

6
Conclusion

In this chapter, the conclusion to the entire thesis project is offered. The general findings and conclusions from all previous chapters of the report are first summarized. Then, answers will be provided to the research questions that were initially posed at the start of the thesis.

6.1. General Findings and Conclusions

In this thesis, the feasibility of using quantum computing to aid in solving practical engineering optimization problems was investigated. As an introduction, the Traveling Salesman Problem was investigated. To set up the TSP an intuitive approach was used to directly create the QUBO matrix that defines the problem. The constraints that are necessary to ensure valid solutions are found were similarly added through an intuitive understanding of the structure of the QUBO matrix. This intuitive approach was taken despite the existence of mathematically defined formulations of the TSP QUBO. This choice was made to help readers new to the topic of quantum annealing and QUBO problems get acquainted with the methods, and hopefully develop an intuitive understanding of QUBO matrices.

The main goal of the thesis was to investigate a typical engineering optimization problem. For this purpose, it was chosen to investigate 2-dimensional truss structures due to their simplicity. A common objective for truss structure optimization is to minimize the system weight, while complying with various strength, displacement, or stability constraints. Investigating how such an optimization could be performed with the quantum annealer therefore became the main area of research. Since the quantum annealer relies on binary variables, an optimization using discrete sets of allowable truss cross-sectional areas was the most natural approach. The main challenge was to consequently formulate a method to cast the discrete truss sizing optimization into a QUBO problem framework.

The first method to cast the truss optimization problem into a QUBO matrix was developed based on intuitive methods. However, it was found to be quite difficult to formulate stress constraints that are compatible with the QUBO formulation. The method that was eventually settled on works by inserting a preference term in the QUBO matrix that steers the QA towards solutions for which the truss RFs are close to 1. After due consideration, this was deemed an unsuitable method, as it yielded trivial optimization problems. Since the quantum annealer always followed the preference defined by the stress constraint, yet in doing so, it only ever gave extremely predictable solutions for the optimization problem. As such, the method leads to a simple decision-making process, where the selection of a new set of cross-sectional areas predictably depends on the current truss RFs. A purely classical implementation of this decision-making scheme, compared to the implementation that uses the quantum annealer, was several orders of magnitude faster while yielding identical final results. It was therefore concluded that this QUBO formulation for the truss optimization problem is trivial. Although extensions and improvements to the method might lead to less trivial optimization problems, these options were forsaken in order to concentrate efforts on a different and potentially more promising methodology.

The second method that was developed for setting up the truss sizing optimization problem, for use with the quantum annealer, takes an approach that is based more on typical finite-element analysis procedures. The essence of the method is that the truss cross-sectional area is defined using a symbolic expression, which is based on a set of allowed discrete choices for the cross-sectional area, and a binary qubit variable corresponding to each choice. Using this symbolic expression for the cross-sectional area, a linear finite-element truss problem is solved symbolically to obtain symbolic expressions for the truss stresses. Several attempts were made to use these symbolic expressions to set up an objective function that could be used as the tar-

get for the optimization of the truss system. Eventually, a suitable general method to define an objective function was formulated. The optimum solution to these objective functions is generally the choice of truss cross-sectional areas that yields a truss system in which every truss is as close as possible to the material limit stress. However, these objective functions are fractional in nature, which means they are not natively compatible with the quantum annealer. An iterative optimization approach was implemented that writes the objective function in a more compatible non-fractional form, and can eventually find the same optimum solution described by the original fractional objective function. Following some further steps to simplify the objective function, as well as an important quadratization step, the discrete truss sizing optimization problem is finally written in a QUBO form. The quantum annealer can then be used to search for optimum solutions to this problem.

The problems that were investigated using this symbolic finite-element QUBO method were three simple truss structures, consisting of two, three, and four total truss elements. The symbolic method used to set up the objective functions for these truss sizing optimization problems scales exponentially. It is likely to be infeasible to use this method for defining objective functions for truss systems containing more than four trusses. Nevertheless, it was found that quantum annealing works reliably for the smallest of these problems. However, the larger two problems showed that the QA had difficulties with finding the global optimum solution. It was also found that, although the QA only uses very little actual QPU access time to solve the problems, there is a large amount of overhead present in the method, meaning that it takes a long time to solve the truss sizing optimization problems in practice.

Overall, the poor scaling of the method to set up the objective function, the low probability of returning the global optimum solution, and the large amount of practical time necessary to find solutions, currently make quantum annealing a poor choice for the truss sizing optimization problem. The next generation of quantum annealing technology is expected to improve performance for this type of optimization problem, due to the increase in qubit connectivity the new hardware will provide. However, further improvement to the methods of casting such practical engineering optimization problems into a quantum-compatible form are certainly also needed if quantum annealers are to be used for practical engineering applications in the future. In Chapter 7 some closing remarks on potential improvements and future work on this topic are offered.

6.2. Answering the Research Questions

The research and development process followed throughout this thesis was guided by a number of research questions. The main research question was:

> ***What method can be used to solve practical structural optimization problems using a quantum computer, and how well does it perform compared to classical methods?***

To answer this main research question, three supporting research questions were asked. These research questions will be reiterated and answered in this section.

6.2.1. Research Question 1

1. *What typical optimization problem can be formulated and solved classically, such that it can act as a reference for the performance of quantum solution algorithms?*
   (a) *What is a typical problem in aerospace engineering that is suitable for solving on both classical and quantum computers?*
   (b) *What method is used to classically solve this problem?*
   (c) *What performance metric can be used to compare the performance of classical and quantum solution algorithms?*

This research question has served to lay the foundation for the thesis project. From the initial literature study, it became apparent that the quantum annealer was the most likely type of quantum computer to be used for practical optimization problems. Within aerospace engineering, structural finite-element problems are ubiquitous. For such problems, 2-dimensional truss structures are one of the most basic types of structures. Therefore, the focus of this thesis was laid on this type of simple truss structure.

The quantum annealer uses binary (0,1) problem variables to solve optimization problems. Thus, for this project, the most natural approach was to use the truss structures to define discrete truss sizing optimization problems. Since the quantum annealer can only solve minimization problems, objective functions must be

formulated which can be used as the target for minimization. Classically, there are several options to find the minimum solution to such functions. The simplest methods to implement in this project were brute-force analysis and simulated annealing. Other, more efficient classical methods are also available, but these would have been more complicated to use. Since the focus of this project was on the feasibility of using quantum computing for these problems, the simplest classical solution methods were chosen.

To evaluate the performance of the quantum annealer with respect to the classical solution methods, several metrics were considered. Classically, brute-force analysis is guaranteed to give the global optimum solution, which is needed for reference purposes. For simulated annealing, as well as quantum annealing, it is important to gauge the probability of finding the global optimum solution, since this is not guaranteed with these methods. Furthermore, the time-to-solution was measured to gauge the performance of the analysis methods. To give insight into potential improvements, a distinction was made between the total practical analysis time, including various sources of overhead, and the pure solve time excluding overhead.

6.2.2. Research Question 2

2. *What method can be used to cast an optimization problem into a formulation that the quantum computer can understand?*

    (a) *What programming interface exists that allows for problems to be submitted to the quantum computer?*

    (b) *How can the optimization problem be cast into a QUBO or Ising model formulation?*

    (c) *What parameters or settings in the quantum computer will influence the time-to-solution, and which settings give the best computational performance?*

The first sub-question for this research question again serves a rather foundational purpose. D-Wave Systems Inc. provides an extensive library of open-source Python tools that can be used to interface with their quantum annealer. Hence, Python was used throughout most of this project. Since the quantum annealer can only interpret QUBO or Ising model problem formulations, the truss system optimization problem must be cast into one of these formulations. Due to its binary nature, the QUBO formulation was preferred for the truss system optimization problems.

Finding a satisfactory answer to the second sub-question was found to be much more difficult than initially anticipated. Two methods were developed to cast truss sizing optimization problems into a QUBO format. The first method, described in Chapter 4, directly casts the truss optimization into a QUBO matrix format. However, this method was eventually found to give trivial optimization problems. The second method, described in Chapter 5, relied on symbolically solving linear FEM truss problems, using binary qubit variables to represent different choices for the truss cross-sectional areas. Using symbolic expressions for the truss stresses, optimization objective functions could then be set up. The minimum solution to these objective functions represents the configuration of trusses that brings the truss stresses closest to the material limit stress. Through various simplifications, quadratization, and by implementing an iterative solving procedure, these objective functions were eventually made compatible with the QUBO problem format. Although many challenges were encountered throughout the process described in Chapter 5, it was found to be difficult, yet feasible, to cast small truss sizing optimization problems into a quantum-compatible format.

Users can change many parameters related to the performance of the quantum annealer. The most notable of these parameters is the total number of reads that are performed during an analysis. When the number of reads is increased, the time-to-solution also increases, but it was seen that this can also improve the likelihood of obtaining the global minimum solution for the truss sizing optimization problems. The best computational performance is obtained when the minimum number of reads is used that still reliably yields the global optimum solution. All other parameters that control the behavior of the quantum annealer were left at their default states for this project.

Overall, the majority of the work needed to answer this research question was related to finding a suitable method to cast the truss optimization problem into a QUBO form. Although the approach described in Chapter 5 appeared straightforward during its conception, the many challenges encountered throughout the process indicate that the translation step from an engineering idea to a functional implementation for the quantum annealer should not be underestimated. The methods by which these challenges have been overcome throughout this process may nevertheless serve an educational purpose to future researchers.

6.2.3. Research Question 3

3. *How does quantum computing compare to classical computing for solving practical optimization problems?*

   (a) *For the reference problem, what is the computational performance of both classical and quantum solution procedures?*

   (b) *What reliability issues exist with quantum computation, and how may these issues be addressed?*

   (c) *If the size of the problem is increased, how does this affect computational performance?*

   (d) *If the quantum computing hardware would be improved in the future, how would this affect computational performance?*

   (e) *At this point in time, is it beneficial to apply quantum computing to practical aerospace engineering problems?*

The time-to-solution was measured for classical and quantum methods, using the truss optimization problem setup described in Chapter 5. In practice, classical brute-force analysis of the fractional objective function, using the set of valid solutions, is very fast compared to classical simulated annealing and quantum annealing. Although simulated annealing is very slow to use for larger problems, these analyses demonstrate that the iterative solution method works as expected and is able to converge on the global optimum solution when using a high number of reads. With quantum annealing, considering only the QPU access time, this method is extremely fast at finding solutions to each of the sample problems. However, the quantum annealer has difficulty in obtaining the global optimum solutions for the three- and four-truss sizing optimization problems.

The low probability of finding the global optimum solutions for the three- and four-truss problems using the quantum annealer could potentially be amended. First, having higher connectivity QPU hardware, as will be present in the next generation of quantum annealers, the length of the qubit chains needed to embed these problems could be significantly reduced. Second, at the expense of additional QPU access time, a greater number of reads could be used for every iteration in the solving procedure. Third, the maximum number of allowed iterations in the solving procedure could be increased, giving the quantum annealer more chances to converge on the global optimum solution. Fourth, an in-depth study of the solver parameters could be performed, to find values that maximize the probability of returning the global optimum solution. Finally, there may still be ways to further simplify the objective functions, thereby reducing the complexity of the problem.

With the current methods, the initial setup of the fractional objective function scales extremely poorly. If a five-truss problem were to be defined, it would likely take approximately 11 hours to find the objective function for such a truss system. If such a system were to be optimized by the quantum annealer, using the current iterative method, the probability of finding the global optimum solution is expected to be even lower than with the four-truss system, due to the increased complexity. Overall, truss systems with more than four trusses are not considered solvable using the current methods.

The next-generation quantum annealers using the Pegasus architecture would present a significant improvement over the current Chimera architecture. Both the number of qubits, as well as the number of connections between qubits, will be increased greatly. This will allow for more complicated problems to be solved, and will likely improve the probability of finding optimal solutions to the truss sizing optimization problems. However, using this new architecture is not expected to influence the time-to-solution compared to the current Chimera architecture. Since the user-defined annealing time will likely still default to a value of 20 μs.

The difficulties seen throughout this project were numerous. It started with symbolically solving finite-element problems, and setting up objective functions, which is a slow process for larger problems. In turn, a special iterative solving procedure was needed to make the objective function more compatible with the quantum annealer. Eventually, it was seen that the quantum annealer has a low probability of finding global optimum solutions to the larger truss sizing problems. Compared to classical methods, these findings show that, although possible, it is currently not beneficial to apply quantum computing to this type of engineering optimization problem. With further development of both quantum annealing technology, as well as the methods to cast engineering problems into a quantum-compatible form, this conclusion may eventually change.

6.2.4. Main Research Question and Research Objective

To offer some final concluding remarks about the research performed in this thesis, the main research question and the research objective are reiterated. The main research question was:

> ***What method can be used to solve practical structural optimization problems using a quantum computer, and how well does it perform compared to classical methods?***

Considering the answers that were provided to each of the research sub-questions in Sections 6.2.1 to 6.2.3, this main research question is considered to have been sufficiently answered. The research goal of this thesis was formalized as the following:

> ***Within the time-span of the thesis, the objective is to investigate the practical application of quantum computing to aerospace structural optimization problems, by developing a method to cast the problem into code that can be interpreted by the quantum computer, and evaluate the performance and reliability compared to classical methods.***

Reflecting on this research goal, the research shown in this thesis indicates that it is difficult, yet feasible, to utilize the D-Wave quantum annealer to solve simple structural optimization problems. The research objective has therefore been achieved. However, the research also shows that there are still a number of challenges left to be overcome before quantum annealing can become a competitive method for this type of optimization problem. These challenges lie both in the methods for casting engineering optimization problems into a quantum-compatible form, as well as the development of the quantum annealing technology itself.

In the next generation of quantum annealers, the connectivity between qubits will be significantly increased. This means that the length of the qubit chains in the embeddings for the truss sizing optimization problems can become much shorter. In turn, this is expected to improve the probability of finding the global optimum solutions for these problems. Furthermore, if sources of overhead can be reduced, by more efficient coding and more direct access to the quantum annealing hardware, quantum annealing has the potential to become an extremely fast method for solving larger structural optimization problems.

In this thesis, some of the first exploratory steps have been taken in an effort to solve structural optimization problems using a quantum computer. Though some of these steps have proven to be quite challenging, the lessons that have been learned may help others find ways to formulate their own problems in a quantum-compatible form. In the future, quantum annealing will hopefully become a competitive tool for engineers, allowing them to solve what has proven to be an interesting and challenging class of optimization problems.

7

Recommendations and Future Work

In this chapter, some of the recommendations and ideas for future work in the field of practical quantum annealing are given.

7.1. Improvements to the Current Methodology

Through the research process carried out in this thesis, it was discovered that it was much more difficult to find a truss optimization QUBO formulation than was initially anticipated. The majority of the time spent during the development of the symbolic finite-element method presented in Chapter 5, was spent on solving the various problems and challenges that were encountered along the way. Even though a compatible formulation was achieved, the difficulty of the translation from a truss optimization concept to a functional QUBO problem indicates that perhaps it was inevitable that the method would produce unreliable results with the quantum annealer. Given that 2-dimensional truss structures are one of the simplest types of structures, it is likely infeasible for this symbolic method to be extended for use with more complicated structures. Since simple finite-element truss systems already gave rise to many difficult challenges, structures containing beams, shells, or solid elements would likely be even more challenging to reformulate using a methodology similar to that presented in this thesis. However, there may be options to improve upon the methods shown in this thesis. Particularly, there are potential improvements in the setup process for generating the fractional objective function which defines the truss sizing optimization problem.

One of the slowest steps in the setup process is the fact that large symbolic expressions must be squared and expanded multiple times. The fractional objective function is defined as shown in Eq. (7.1).

$$T = \sum_{n=1}^{N} T_n = \sum_{n=1}^{N} \left(\sigma_{limit}^2 - \sigma_n^2\right)^2 \tag{7.1}$$

The difficulty with setting up the objective function is that the complicated symbolic expression σ_n must be squared and expanded. Furthermore, once the difference $\left(\sigma_{limit}^2 - \sigma_n^2\right)$ has been calculated, this difference is yet again squared and expanded such that it can be used as a target for minimization. This process then repeats for all trusses in the system, getting ever more complicated and time-consuming for larger truss systems, due to the increased number of variables. However, this setup process could be drastically improved if it would be known beforehand which trusses would be in compression, and which would be in tension. The initial squaring of the truss stresses is performed to ensure that σ_n^2 always evaluates to positive values. If it was known beforehand which trusses would be in tension and which would be in compression, this squaring operation could be avoided by simply setting up the objective function in different ways in each case. Namely, for tensile trusses the objective would become $(\sigma_{limit} - \sigma_n)^2$, while for compressive trusses it would be $(\sigma_{limit} + \sigma_n)^2$. The change in sign in the expression causes the negative compressive trusses to still be subtracted from the positively defined limit stress, ensuring that the difference is calculated correctly.

Another possible improvement to the setup process for the fractional objective function, which should lead to a significant speed improvement, is to apply simplifications throughout the process. Currently, one of the simplifications that is performed after the setup of the fractional objective function is complete, is to truncate all terms that are of an order greater than the number of trusses in the system. For example, for a three-truss system, all terms greater than third order are truncated. These terms only contribute information about invalid solutions to the optimization problem. Since invalid solutions should be avoided altogether, it is beneficial to simply remove these terms from the objective function. The idea for the speed improvement

is therefore to remove such high-order terms throughout the setup process of the objective function, rather than performing the entire setup process first, and performing the truncation at the end. By continuously removing high-order terms, which appear after each squaring operation, no unnecessary effort is wasted in manipulating these terms further. Since these computationally expensive squaring operations are performed multiple times, this simplification is likely to lead to a significant speedup of the objective function setup method.

The setup process for the original fractional objective functions could be further improved by considering parallel execution of certain sections of the MATLAB code. After the symbolic expressions for the nodal displacement have been obtained, the process continues with finding expressions for the truss stresses and individual truss objective functions. This happens sequentially, for every truss in the system. Once the individual truss objective functions have all been found, these are then summed to obtain the final fractional objective function for the whole truss system. However, the portion of the code that sequentially finds the expressions for the truss stresses and objective functions may be suitable for parallel execution. Since, these expressions can be set up independently from each other and only depend on the symbolic expressions for the nodal displacements. A speed improvement may be possible if the individual truss stress and objective function expressions are found simultaneously for multiple trusses in the truss system.

Finally, one of the major constraints that prevented a truly in-depth investigation into the behavior of the quantum annealer for this thesis was the limited amount of QPU access time available. This meant that many of the analyses were only performed ten times each, to gauge the performance of the quantum annealing method. Although performing ten analyses gives some insight into the behavior of the quantum computer, it is not sufficient to determine definitive statistical measures. If many more analyses had been performed, more statistically significant claims could be made with respect to, for example, the probability of obtaining the global minimum solution. As it stands, having performed only ten quantum annealing attempts to solve each of the different truss optimization problems, this is only enough to give a preliminary conclusion. If further research on this topic were to be performed, it would be extremely beneficial to have much more access to the quantum annealing hardware.

7.2. Suggestions for Future Work

The work performed in this thesis was mostly related to finding a way to translate a truss sizing optimization problem into a format that is compatible with the quantum annealer. However, even though such a translation was eventually successfully found, the quantum annealer had difficulty finding optimal solutions to the larger truss optimization problems. The issues that were encountered during the development of the symbolic finite-element QUBO method were perhaps an indication that this type of truss optimization problem is not a suitable match for the quantum annealer, causing the problem to be difficult to solve. Alternatively, perhaps the method used to translate the truss optimization problem into a QUBO format was unsuitable. Nevertheless, the technology is still considered promising, and for future research into practical applications for the aerospace industry, different types of optimization problems should be chosen.

Some unique applications of quantum annealing have been found in [3, 45, 62, 81, 85]. In these studies, tailor-made QUBO formulations are used to solve clear, practical optimization problems. These works could inspire further research into the practical applications of quantum annealing. Of particular interest might be the work shown by Van Vreumingen et al. [85], which deals with a shape optimization of a sphere, defined through finite-elements. Shape optimizations are naturally also quite applicable to the aerospace industry, considering topics like aerodynamic design, or topology-optimized structural design. It is suggested that, for future research into quantum annealing applications for aerospace engineering, inspiration is drawn from the work by Van Vreumingen et al., and a shape optimization problem is investigated.

Perhaps a similar approach to that of Van Vreumingen et al. can be used to optimize the positioning of the nodes that define a truss system. This may lead to truss system designs with optimized load-paths, which could be a first step towards the more complex shape optimizations of aircraft structures. Further inspiration for such an optimization could also be taken from the formulations for the Traveling Salesman Problem. Trusses and nodes can also be interpreted as cities and roads, meaning that it is perhaps feasible to use a TSP-like formulation to find the most efficient load paths between the loaded nodes and the structural boundaries. Further extensions might also include methods that can change the topology of the structure, by removal of inefficient elements, or exploring new possibilities by allowing new additional elements to be generated. Such complicated optimizations might be possible using the novel approaches shown by King et al. [49] and Pastorello and Blanzieri [68], which combine quantum annealing with genetic algorithms, or other learning

methods.

On a different note, many of the challenges in this project were related to casting the engineering optimization problem into a suitable QUBO form. It would have been helpful if there had been tools or computer programs available that could aid in this translation process. The development of such a general program is likely to be more appropriate work for someone that specializes in computer science and software development. Nevertheless, it could end up being an incredibly useful tool for engineers to help translate their practical optimization problems into a valid QUBO formulation. For instance, such a program could ask what type of problem the engineer wanted to solve, would assist in defining the problem variables and constraints, and then generate the QUBO matrix. This would make it much easier for engineers to integrate quantum computing into the toolbox of technologies they use on a daily basis to do their jobs.

Bibliography

[1] S. Aaronson. Read the fine print. *Nature Physics*, 11(4):291, 2015.

[2] Airbus. Airbus Quantum Computing Challenge: Bringing flight physics into the Quantum Era. Online, Jan 2019. URL https://www.airbus.com/innovation/tech-challenges-and-competitions/airbus-quantum-computing-challenge.html. Accessed: 2019-11-06.

[3] A. Ajagekar, T. Humble, and F. You. Quantum Computing based Hybrid Solution Strategies for Large-scale Discrete-Continuous Optimization Problems. *arXiv e-prints*, art. arXiv:1910.13045, Oct 2019.

[4] T. Albash and D. A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1), Jan. 2018. URL https://doi.org/10.1103/revmodphys.90.015002.

[5] A. Ambainis. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. *arXiv e-prints*, art. arXiv:1010.4458, Oct. 2010.

[6] M. Amico, Z. H. Saleem, and M. Kumph. Experimental study of Shor's factoring algorithm using the IBM Q Experience. *Phys. Rev. A*, 100:012305, Jul 2019. URL https://link.aps.org/doi/10.1103/PhysRevA.100.012305.

[7] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. 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(7779):505–510, Oct. 2019. URL https://doi.org/10.1038/s41586-019-1666-5.

[8] S. Barz, I. Kassal, M. Ringbauer, Y. O. Lipp, B. Dakić, A. Aspuru-Guzik, and P. Walther. A two-qubit photonic quantum processor and its application to solving systems of linear equations. *Scientific Reports*, 4: 6115, Aug 2014. doi: 10.1038/srep06115.

[9] C. Bernhardt. *Quantum Computing for Everyone (The MIT Press)*. The MIT Press, Mar. 2019. ISBN 0262039257. URL https://www.xarg.org/ref/a/0262039257/.

[10] R. Biswas, Z. Jiang, K. Kechezhi, S. Knysh, S. Mandrà, B. O'Gorman, A. Perdomo-Ortiz, A. Petukhov, J. Realpe-Gómez, E. Rieffel, D. Venturelli, F. Vasko, and Z. Wang. A NASA perspective on quantum computing: Opportunities and challenges. *Parallel Computing*, 64:81–98, May 2017. URL https://doi.org/10.1016/j.parco.2016.11.002.

[11] S. Boixo, T. Albash, F. M. Spedalieri, N. Chancellor, and D. A. Lidar. Experimental signature of programmable quantum annealing. *Nature Communications*, 4(1), June 2013. ISSN 2041-1723. URL http://dx.doi.org/10.1038/ncomms3067.

[12] A. Borle and S. J. Lomonaco. Analyzing the quantum annealing approach for solving linear least squares problems. In *International Workshop on Algorithms and Computation*, pages 289–301. Springer, 2019.

[13] X.-D. Cai, C. Weedbrook, Z.-E. Su, M.-C. Chen, M. Gu, M.-J. Zhu, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan. Experimental Quantum Computing to Solve Systems of Linear Equations. *Phys. Rev. Lett.*, 110:230501, Jun 2013. URL https://link.aps.org/doi/10.1103/PhysRevLett.110.230501.

[14] Y. Cao, A. Daskin, S. Frankel, and S. Kais. Quantum circuit design for solving linear systems of equations. *Molecular Physics*, 110(15-16):1675–1680, 2012. URL https://doi.org/10.1080/00268976.2012.668289.

[15] A. M. Childs, R. Kothari, and R. D. Somma. Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision. *SIAM Journal on Computing*, 46(6):1920–1950, Jan 2017. ISSN 1095-7111. URL http://dx.doi.org/10.1137/16M1087072.

[16] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. Preconditioned Quantum Linear System Algorithm. *Phys. Rev. Lett.*, 110:250504, Jun 2013. URL https://link.aps.org/doi/10.1103/PhysRevLett.110.250504.

[17] C. Coffrin, H. Nagarajan, and R. Bent. Evaluating Ising Processing Units with Integer Programming. In L.-M. Rousseau and K. Stergiou, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 163–181, Cham, 2019. Springer International Publishing. ISBN 978-3-030-19212-9.

[18] P. J. Coles, S. Eidenbenz, S. Pakin, A. Adedoyin, J. Ambrosiano, P. Anisimov, W. Casper, G. Chennupati, C. Coffrin, H. Djidjev, D. Gunter, S. Karra, N. Lemons, S. Lin, A. Lokhov, A. Malyzhenkov, D. Mascarenas, S. Mniszewski, B. Nadiga, D. O'Malley, D. Oyen, L. Prasad, R. Roberts, P. Romero, N. Santhi, N. Sinitsyn, P. Swart, M. Vuffray, J. Wendelberger, B. Yoon, R. Zamora, and W. Zhu. Quantum Algorithm Implementations for Beginners. *arXiv e-prints*, art. arXiv:1804.03719, Apr. 2018.

[19] D-Wave Systems Inc. D-Wave Systems Inc. Repositories. Online, Dec 2017. URL https://github.com/dwavesystems. Accessed: 2019-11-08.

[20] D-Wave Systems Inc. dimod.higherorder.utils.make_quadratic. Online, 2017. URL https://docs.ocean.dwavesys.com/projects/dimod/en/latest/reference/generated/dimod.higherorder.utils.make_quadratic.html. Accessed: 2020-07-08.

[21] D-Wave Systems Inc. *Getting Started with the D-Wave System: User Manual*, Oct. 2018.

[22] D-Wave Systems Inc. Introduction to the D-Wave Quantum Hardware. Online, 2019. URL https://www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware. Accessed: 2020-08-20.

[23] D-Wave Systems Inc. *QPU Properties: D-Wave 2000Q Online System (DW_2000Q_2_1)*, Aug. 2019. URL https://support.dwavesys.com/hc/article_attachments/360043207874/09-1177A-F_QPU_Properties_Online_DW_2000Q_2_1.pdf.

[24] D-Wave Systems Inc. D-Wave Leap Dashboard. Online, 2019. URL https://cloud.dwavesys.com/leap/. Acessed: 2019-11-01.

[25] D-Wave Systems Inc. dwavesystems /dwave-inspector. Online, May 2020. URL https://github.com/dwavesystems/dwave-inspector. Accessed: 2020-07-03.

[26] D-Wave Systems Inc. *D-Wave Problem-Solving Handbook*, Feb. 2020. URL https://docs.dwavesys.com/docs/latest/doc_handbook.html.

[27] D-Wave Systems Inc. *D-Wave Solver Properties and Parameters Reference*, Feb. 2020. URL https://docs.dwavesys.com/docs/latest/doc_solver_ref.html.

[28] D-Wave Systems Inc. *Solver Computation Time User Manual*, Feb. 2020. URL https://docs.dwavesys.com/docs/latest/doc_timing.html.

[29] N. Dattani. Quadratization in discrete optimization and quantum mechanics. *arXiv e-prints*, art. arXiv:1901.04405, Jan. 2019.

[30] N. Dattani and N. Chancellor. Embedding quadratization gadgets on Chimera and Pegasus graphs. *arXiv e-prints*, art. arXiv:1901.07676, Jan 2019.

[31] D. Dervovic, M. Herbster, P. Mountney, S. Severini, N. Usher, and L. Wossnig. Quantum linear systems algorithms: a primer. *arXiv e-prints*, art. arXiv:1802.08227, Feb. 2018.

[32] H. N. Djidjev, G. Chapuis, G. Hahn, and G. Rizk. Efficient Combinatorial Optimization Using Quantum Annealing. *arXiv e-prints*, art. arXiv:1801.08653, Nov 2016.

[33] Y.-L. Fang and P. A. Warburton. Minimizing minor embedding energy: an application in quantum annealing. *arXiv e-prints*, art. arXiv:1905.03291, May 2019.

[34] S. Feld, C. Roch, T. Gabor, C. Seidel, F. Neukart, I. Galter, W. Mauerer, and C. Linnhoff-Popien. A Hybrid Solution Method for the Capacitated Vehicle Routing Problem Using a Quantum Annealer. *Frontiers in ICT*, 6, June 2019. URL https://www.frontiersin.org/article/10.3389/fict.2019.00013.

[35] B. Gardas, J. Dziarmaga, W. H. Zurek, and M. Zwolak. Defects in Quantum Computers. *Scientific Reports*, 8(1), Mar. 2018. URL https://doi.org/10.1038/s41598-018-22763-2.

[36] F. Glover, G. Kochenberger, and Y. Du. A Tutorial on Formulating and Using QUBO Models. *arXiv e-prints*, art. arXiv:1811.11538, Nov. 2018.

[37] F. Glover, G. Kochenberger, M. Ma, and Y. Du. Quantum Bridge Analytics II: Combinatorial Chaining for Asset Exchange. *arXiv e-prints*, art. arXiv:1911.03036, Nov 2019.

[38] R. Hamerly, T. Inagaki, P. L. McMahon, D. Venturelli, A. Marandi, T. Onodera, E. Ng, C. Langrock, K. Inaba, T. Honjo, and et al. Experimental investigation of performance differences between coherent Ising machines and a quantum annealer. *Science Advances*, 5(5):eaau0823, May 2019. ISSN 2375-2548. URL http://dx.doi.org/10.1126/sciadv.aau0823.

[39] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009. URL https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[40] M. Hasan and S. Acharjee. Solving LFP by Converting it into a Single LP. *International Journal of Operations Research*, 8:1–14, Jan. 2011. URL https://www.researchgate.net/publication/319135012_Solving_LFP_by_Converting_it_into_a_Single_LP.

[41] I. Hen and M. S. Sarandy. Driver Hamiltonians for constrained optimization in quantum annealing. *Phys. Rev. A*, 93:062312, Jun 2016. URL https://link.aps.org/doi/10.1103/PhysRevA.93.062312.

[42] I. Hen and F. M. Spedalieri. Quantum Annealing for Constrained Optimization. *Phys. Rev. Applied*, 5: 034007, Mar 2016. doi: 10.1103/PhysRevApplied.5.034007. URL https://link.aps.org/doi/10.1103/PhysRevApplied.5.034007.

[43] IBM. IBM Q Experience. Online, 2019. URL https://quantum-computing.ibm.com/. Acessed: 2019-11-01.

[44] IBM. IBM Unveils World's First Integrated Quantum Computing System for Commercial Use. Online, Jan. 2019. URL https://newsroom.ibm.com/2019-01-08-IBM-Unveils-Worlds-First-Integrated-Quantum-Computing-System-for-Commercial-Use. Acessed: 2019-11-01.

[45] K. Ikeda, Y. Nakamura, and T. S. Humble. Application of Quantum Annealing to Nurse Scheduling Problem. *Scientific Reports*, 9(1):12837, Sep 2019. ISSN 2045-2322. URL https://doi.org/10.1038/s41598-019-49172-3.

[46] S. Jiang, K. A. Britt, A. J. McCaskey, T. S. Humble, and S. Kais. Quantum Annealing for Prime Factorization. *Scientific Reports*, 8(1), Dec. 2018. URL https://doi.org/10.1038/s41598-018-36058-z.

[47] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, May 2011. URL https://doi.org/10.1038/nature10012.

[48] H. G. Katzgraber, F. Hamze, Z. Zhu, A. J. Ochoa, and H. Munoz-Bauza. Seeking Quantum Speedup Through Spin Glasses: The Good, the Bad, and the Ugly. *Phys. Rev. X*, 5:031026, Sep 2015. URL https://link.aps.org/doi/10.1103/PhysRevX.5.031026.

[49] J. King, M. Mohseni, W. Bernoudy, A. Fréchette, H. Sadeghi, S. V. Isakov, H. Neven, and M. H. Amin. Quantum-Assisted Genetic Algorithm. *arXiv e-prints*, art. arXiv:1907.00707, Jun 2019.

[50] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81, apr 2014. URL https://doi.org/10.1007/s10878-014-9734-0.

[51] M. Kochenderfer and T. Wheeler. *Algorithms for Optimization*. The MIT Press. MIT Press, 2019. ISBN 9780262039420. URL https://books.google.nl/books?id=uBSMDwAAQBAJ.

[52] C. R. Laumann, R. Moessner, A. Scardicchio, and S. L. Sondhi. Quantum annealing: The fastest route to quantum computation? *The European Physical Journal Special Topics*, 224(1):75–88, Feb. 2015. URL https://doi.org/10.1140/epjst/e2015-02344-2.

[53] W. Lechner, P. Hauke, and P. Zoller. A quantum annealing architecture with all-to-all connectivity from local interactions. *Science Advances*, 1(9):e1500838, Oct. 2015. URL https://doi.org/10.1126/sciadv.1500838.

[54] Y. Lee, J. Joo, and S. Lee. Hybrid quantum linear equation algorithm and its experimental test on IBM Quantum Experience. *Scientific reports*, 9(1):4778, 2019.

[55] A. Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014. ISSN 2296-424X. URL http://dx.doi.org/10.3389/fphy.2014.00005.

[56] A. Lucas. Hard combinatorial problems and minor embeddings on lattice graphs. *Quantum Information Processing*, 18(7):203, Jul 2019. URL https://doi.org/10.1007/s11128-019-2323-5.

[57] C. C. McGeoch. *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*, volume 5. Morgan & Claypool Publishers LLC, July 2014. URL https://doi.org/10.2200/s00585ed1v01y201407qmc008.

[58] C. C. McGeoch, R. Harris, S. P. Reinhardt, and P. I. Bunyk. Practical Annealing-Based Quantum Computing. *Computer*, 52(6):38–46, June 2019. URL https://doi.org/10.1109/mc.2019.2908836.

[59] A. Mizel, D. A. Lidar, and M. Mitchell. Simple Proof of Equivalence between Adiabatic Quantum Computation and the Circuit Model. *Physical Review Letters*, 99(7), Aug 2007. ISSN 1079-7114. URL http://dx.doi.org/10.1103/PhysRevLett.99.070502.

[60] M. Möller and C. Vuik. A conceptual framework for quantum accelerated automated design optimization. *Microprocessors and Microsystems*, 66:67 – 71, 2019. ISSN 0141-9331. URL http://www.sciencedirect.com/science/article/pii/S0141933118303223.

[61] A. Montanaro and S. Pallister. Quantum algorithms and the finite element method. *Phys. Rev. A*, 93:032324, Mar 2016. URL https://link.aps.org/doi/10.1103/PhysRevA.93.032324.

[62] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney. Traffic flow optimization using a quantum annealer. *arXiv e-prints*, art. arXiv:1708.01625, Aug 2017.

[63] NumPy.org. numpy.amax - NumPy v1.19 Manual. Online, Jun 2020. URL https://numpy.org/doc/stable/reference/generated/numpy.amax. Accessed: 2020-08-19.

[64] M. Ohkuwa, H. Nishimori, and D. A. Lidar. Reverse annealing for the fully connected p-spin model. *Phys. Rev. A*, 98:022314, Aug 2018. URL https://link.aps.org/doi/10.1103/PhysRevA.98.022314.

[65] M. Ohzeki, C. Takahashi, S. Okada, M. Terabe, S. Taguchi, and K. Tanaka. Quantum annealing: next-generation computation and how to implement it when information is missing. *Nonlinear Theory and Its Applications, IEICE*, 9(4):392–405, 2018. URL https://doi.org/10.1587/nolta.9.392.

[66] S. Okada, M. Ohzeki, M. Terabe, and S. Taguchi. Improving solutions by embedding larger subproblems in a D-Wave quantum annealer. *Scientific Reports*, 9(1), Feb. 2019. URL https://doi.org/10.1038/s41598-018-38388-4.

[67] J. Pan, Y. Cao, X. Yao, Z. Li, C. Ju, H. Chen, X. Peng, S. Kais, and J. Du. Experimental realization of quantum algorithm for solving linear systems of equations. *Physical Review A*, 89(2), Feb 2014. ISSN 1094-1622. URL http://dx.doi.org/10.1103/PhysRevA.89.022313.

[68] D. Pastorello and E. Blanzieri. Quantum annealing learning search for solving QUBO problems. *Quantum Information Processing*, 18(10):303, Aug 2019. ISSN 1573-1332. URL https://doi.org/10.1007/s11128-019-2418-z.

[69] E. Pednault, J. Gunnels, D. Maslov, and J. Gambetta. On "Quantum Supremacy". Online, Oct. 2019. URL https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/. Acessed: 2020-07-20.

[70] A. Perdomo-Ortiz, A. Feldman, A. Ozaeta, S. V. Isakov, Z. Zhu, B. O'Gorman, H. G. Katzgraber, A. Diedrich, H. Neven, J. de Kleer, B. Lackey, and R. Biswas. Readiness of Quantum Optimization Machines for Industrial Applications. *Phys. Rev. Applied*, 12:014004, Jul 2019. URL https://link.aps.org/doi/10.1103/PhysRevApplied.12.014004.

[71] QuTech. About QI. Retrieved from Quantum Inspire, Oct. 2019. URL https://www.quantum-inspire.com/about-qi. Acessed: 2019-11-01.

[72] S. Resch and U. R. Karpuzcu. Quantum Computing: An Overview Across the System Stack. *arXiv e-prints*, art. arXiv:1905.07240, May 2019.

[73] Rigetti Computing. Rigetti right now. Online, May 2020. URL https://www.rigetti.com/. Accessed: 2020-07-22.

[74] M. L. Rogers and J. Singleton, Robert L. Floating-Point Calculations on a Quantum Annealer: Division and Matrix Inversion. *arXiv e-prints*, art. arXiv:1901.06526, Jan 2019.

[75] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, June 2014. URL https://doi.org/10.1126/science.1252319.

[76] G. Rosenberg, M. Vazifeh, B. Woods, and E. Haber. Building an iterative heuristic solver for a quantum annealer. *Computational Optimization and Applications*, 65(3):845–869, Apr. 2016. URL https://doi.org/10.1007/s10589-016-9844-y.

[77] A. Rothwell. *Optimization Methods in Structural Design*. Springer International Publishing, 2017. doi: 10.1007/978-3-319-55197-5. URL https://doi.org/10.1007/978-3-319-55197-5.

[78] S. W. Shin, G. Smith, J. A. Smolin, and U. Vazirani. How "Quantum" is the D-Wave Machine? *arXiv e-prints*, art. arXiv:1401.7087, Jan. 2014.

[79] F. A. Simi and M. S. Talukder. A New Approach for Solving Linear Fractional Programming Problems with Duality Concept. *Open Journal of Optimization*, 06(01):1–10, 2017. URL https://doi.org/10.4236/ojop.2017.61001.

[80] R. D. Somma, D. Nagaj, and M. Kieferová. Quantum Speedup by Quantum Annealing. *Phys. Rev. Lett.*, 109:050501, Jul 2012. URL https://link.aps.org/doi/10.1103/PhysRevLett.109.050501.

[81] T. Stollenwerk, B. O'Gorman, D. Venturelli, S. Mandrà, O. Rodionova, H. Ng, B. Sridhar, E. G. Rieffel, and R. Biswas. Quantum Annealing Applied to De-Conflicting Optimal Trajectories for Air Traffic Management. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–13, 2019. ISSN 1558-0016. URL http://dx.doi.org/10.1109/TITS.2019.2891235.

[82] M. Stolpe. Truss optimization with discrete design variables: a critical review. *Structural and Multidisciplinary Optimization*, 53(2):349–374, Feb 2016. ISSN 1615-1488. URL https://doi.org/10.1007/s00158-015-1333-x.

[83] Sz. Szalay, N. Dattani, and N. Chancellor. PegasusDraw. https://github.com/HPQC-LABS/PegasusDraw/, 2018.

[84] O. Ubbens. Practical Implementation of a Quantum Algorithm for the Solution of Systems of Linear Systems of Equations. Bachelor's thesis, Delft University of Technology, 2019. URL http://resolver.tudelft.nl/uuid:444580f0-a661-4adc-a937-51c5660916d9.

[85] D. Van Vreumingen, F. Neukart, D. Von Dollen, C. Othmer, M. Hartmann, A.-C. Voigt, and T. Bäck. Quantum-assisted finite-element design optimization. *arXiv e-prints*, art. arXiv:1908.03947, Aug 2019.

[86] D. Venturelli and A. Kondratyev. Reverse Quantum Annealing Approach to Portfolio Optimization Problems. *arXiv e-prints*, art. arXiv:1810.08584, Oct 2018.

[87] Volkswagen. Volkswagen optimizes traffic flow with quantum computers . Online, Oct. 2019. URL https://www.volkswagen-newsroom.com/en/press-releases/volkswagen-optimizes-traffic-flow-with-quantum-computers-5507. Accessed: 2020-09-04.

[88] T. Vyskočil, S. Pakin, and H. N. Djidjev. Embedding Inequality Constraints for Quantum Annealing Optimization. In *Quantum Technology and Optimization Problems*, pages 11–22. Springer International Publishing, 2019. URL https://doi.org/10.1007/978-3-030-14082-3_2.

[89] R. H. Warren. Gates for Adiabatic Quantum Computing. *arXiv e-prints*, art. arXiv:1405.2354, May 2014.

[90] C. Whyte. Google has reached quantum supremacy - here's what it should do next. Online, Sept. 2019. URL https://www.newscientist.com/article/2217835-google-has-reached-quantum-supremacy-heres-what-it-should-do-next/. Acessed: 2019-11-01.

[91] K. Wils. Truss Sizing Optimization: Symbolic Finite Element Method QUBO Repository, 2020. URL https://doi.org/10.34894/PYZGEX.

[92] K. Wils. Truss Sizing Optimization: Direct QUBO Method Repository, 2020. URL https://doi.org/10.34894/OU99WD.

[93] K. Wils. TSP QUBO Repository., 2020. URL https://doi.org/10.34894/QYHGAE.

[94] L. Wossnig, Z. Zhao, and A. Prakash. Quantum Linear System Algorithm for Dense Matrices. *Phys. Rev. Lett.*, 120:050502, Jan 2018. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.050502.

[95] G. Xu and W. S. Oates. Can Quantum Computers Solve Linear Algebra Problems to Advance Engineering Applications? In *ASME 2018 Conference on Smart Materials, Adaptive Structures and Intelligent Systems*. American Society of Mechanical Engineers Digital Collection, 2018.

[96] N. S. Yanofsky. *Quantum Computing for Computer Scientists*. Cambridge University Press, Aug. 2008. ISBN 0521879965. URL https://www.xarg.org/ref/a/0521879965/.

[97] Y. Zheng, C. Song, M.-C. Chen, B. Xia, W. Liu, Q. Guo, L. Zhang, D. Xu, H. Deng, K. Huang, Y. Wu, Z. Yan, D. Zheng, L. Lu, J.-W. Pan, H. Wang, C.-Y. Lu, and X. Zhu. Solving Systems of Linear Equations with a Superconducting Quantum Processor. *Phys. Rev. Lett.*, 118:210504, May 2017. URL https://link.aps.org/doi/10.1103/PhysRevLett.118.210504.