Applications of quantum annealing in combinatorial optimization

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op dinsdag 20 december 2022
klokke 11.15 uur

door

Sheir Yarkoni

geboren te Tampa, FL, USA
in 1990

Acknowledgements

I would like to thank my supervisors, colleagues, friends, and family who supported me throughout my studies. No work of science is ever truly done alone, and everyone in my life has a part in this thesis alongside me.

Contents

Glossary

$\mathcal{T}$	Universal Turing machine.
$\mathcal{Q}$	Universal quantum Turing machine.
$\mathbb{R}$	The field of real numbers.
$\mathbf{B}$	The domain of Boolean values, $\{0, 1\}$.
ψ	Wave function of a single qubit.
$\|0\rangle$,$\|1\rangle$	Qubit computational basis states in bra-ket notation.
$\sigma_x, \sigma_y, \sigma_z$	The 2×2 Pauli-x, -y, and -z spin operators.
$\mathcal{H}$	Generic Hamiltonian operator.
$\mathcal{H}_i$	Initial Hamiltonian operator for Adiabatic Quantum Computing and quantum annealing.
$\mathcal{H}_f$	Final Hamiltonian operator for Adiabatic Quantum Computing and quantum annealing.
τ	Timescale of evolution for a time-dependent Hamiltonian.
x	Vector of N binary variables, $x_i \in \{0, 1\}$.
$\mathbf{Q}$	$N \times N$ QUBO matrix for binary variables.
s	Spin variable $s_i \in \{-1, 1\}$.
h_i	Linear weight for spin variable s_i (also known as bias).
J_{ij}	Quadratic interaction term for spin variables s_i and s_j (coupling strength).
$\mathcal{U}(a, b)$	The uniform random distribution between given points a and b.

$\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$	The standard multivariate normal distribution.
log	Natural logarithm.
$\log_2$	Natural logarithm (base 2).
${}_nC_k$	Binomial coefficient, "n choose k".

CHAPTER 1

Introduction

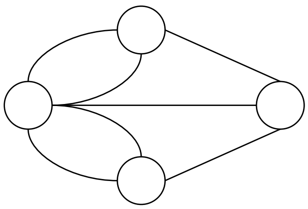
"[...] Nothing at all takes place in the universe in which some rule of maximum or minimum does not appear."

- Leonhard Euler

The field of mathematical optimization has a long and storied history spanning multiple centuries. Dating back to the 1700s, famous mathematicians such as Newton, Fermat, and Lagrange each contributed significantly to the early development of mathematical analysis in finding optima of functions. In recent decades, pioneering works by physicists and mathematicians such as Richard Feynman, David Deutsch, Hidetoshi Nishimori, and Edward Farhi have created exciting new related fields of research: quantum computing and quantum optimization. Here we give a brief history and introduction to the core concepts in these fields, how they relate to combinatorial optimization, and the motivation for the research presented in this thesis.

Now known as "The Seven Bridges of Königsberg", this problem has become synonymous with the birth of the field of graph theory, and is stated as follows: is it possible to traverse the city of Königsberg so that each of the seven bridges is crossed once, and only once? Through trial and error, it is easy to demonstrate that such a walk is not possible for the city of Königsberg and its bridges, Fig. 1.1 (a). However, it was a peculiar curiosity from Leonhard Euler about this problem that spawned completely new branches of mathematics: graph theory and combinatorics. Euler's insight was that there were some innate topological features about the structure and connectivity of the land masses in the city which made such a walk impossible. The notion of a *graph* was invented: each land mass in the city can be represented by a *node*, and the bridges connecting them using *edges*, with the

(a)



(b)

Figure 1.1: **(a)** Depiction of the city of Königsberg in Euler's time with its original seven bridges, circa 1700. Image is public domain, taken from Wikipedia Commons [3]. **(b)** Graph representation of the city of Königsberg, with land masses as nodes and bridges as edges.

number of edges connected to each node being its *degree*. Thus, it was possible to study the features of these graphical structure in an abstract way, rather than any one particular city (shown in Fig. 1.1 (b)). For Königsberg, the solution is that in order to have a walk that crosses each bridge only once, it may only have two odd-degree nodes; one for the start of the walk, and the other for its end. However, as shown in Fig. 1.1 (b), all four nodes in the graph have odd degree, and therefore no valid walk is possible [1]. Euler generalized this notion (later proven rigorously by Carl Hierholzer [2]), asserting that all graphs that have only even degree nodes have such a walk, and that in order for these walks to exist, there must be exactly zero or two nodes of odd degree in the graph. These are now named *Eulerian paths* in his honor.

While relevant historically for its role in graph theory, “The Seven Bridges of Königsberg” is not particularly interesting (or difficult) computationally. For every graph G with nodes V and edges E, Hierholzer's algorithm can find Eulerian paths in time $O(|E|)$. However, the problem serves as a conceptual intersection point between multiple disciplines that are central to this thesis. First is the connection between graph theory and combinatorics. While finding a Eulerian path is simple, and clearly relates to the theory of graphs and their connectivity, a simple related task of finding *all* Eulerian circuits in a graph is more difficult– it now deals with combinations and counting rather than pure graphical structure. In this thesis, similar such connections are presented (although undoubtedly less profound): the results of investigating the link between structures in graphs and the ability of quantum algorithms to solve problems based on them. Secondly, it is known that many optimization problems are fundamentally intractable at large sizes using classical computers, and so solving them may not be possible on practical timescales. Quantum algorithms offer a potential path towards solving these problems more efficiently. Analogously to the graph theory representation of bridges for Euler, we regard quantum algorithms as tools, and the research presented here reports on their use in the context of combinatorial optimization. The conceptual links that need to be made between optimization problems and quantum algorithms is why quantum optimization (and specifically quantum annealing) was investigated in this thesis. Lastly, it is important to appreciate the context in which Euler's original results were reported. While the original article was published in a scientific journal [1], the description of both the problem and solution relates to a problem that manifests in reality: the arrangement of bridges in a city. Therefore, the consequences of the theoretical properties proven in the paper could be observed directly by walking through the city of Königsberg and attempting to cross all bridges only once[1]. Essentially, the Königsberg bridges result could be regarded as a real-world application of graph theory. This intersection between theory and reality is also a cornerstone of the work in this thesis. The goal of the research that will be presented was to gain insight into how algorithms in quantum optimization may be used in practice, and therefore the motivation for many of the results are grounded in the application of these methods for real-world problems.

1.1 Foundations of quantum computing

The theory behind quantum computing (QC) dates to the early 1980s, developed independently by scientists Yuri Manin [4], Paul Benioff [5, 6] and Richard Feynman [7]. Together, these works spawned the fields of modern-day quantum computing and quantum information science. Yuri Manin was the first to envision such a paradigm as a more powerful computation model. In the works by Benioff, the focus was on the representation of a reversible universal Turing machine described by the evolution of a quantum system. The contribution from Feynman was

[1]Some of the original bridges have since been destroyed; today only two of the original seven remain [3]

concerned with the ability of classical computing machines to simulate quantum physics, a computationally intractable task. It was therefore proposed that one could design a programmable quantum system to perform computation. In [8], David Deutsch formalized the notion of a fully quantum model of computation, and specifies the theoretical concepts required to realize a universal quantum computer, $\mathcal{Q}$, the quantum Turing machine. These machines use the quantum equivalent of bits, referred to as quantum bits, or *qubits*. As with classical bits, these can be in either of the binary states 0 or 1. However, as quantum mechanical objects, qubits can also be in states which are combinations of both 0 and 1 simultaneously until measured (or observed). The states of observation are referred to as the *computational basis states*. Computational operations are defined via the use of unitary operators acting on these qubits (in discrete time steps) in order to construct quantum circuits, representing quantum algorithms (or quantum simulations).

In practice, the power of qubits (and therefore quantum computers) exploits three fundamental principles of quantum mechanics.

- **Coherence:** Each of the individual qubits in the quantum system cannot be described via classical physics, but can only be described by the time-dependent Schrödinger wave equation. Thus, the evolution of qubits in a quantum computer is described via quantum mechanical wave dynamics and unitary operations.
- **Entanglement:** The individual components of the quantum system are inseparable; meaning, they cannot be described independently. Only a single physical description of the ensemble of elements (e.g., the whole entangled system) is possible. Consequentially, any operation applied to a qubit in a quantum computer affects all other qubits that are entangled with it.
- **Superposition:** Each qubit in the quantum computer may be in multiple states at the same time. Observing the qubits' states results in the measurement of a single configuration based on the states' individual probabilities. Furthermore, the observation (or collapsing) of the qubits results in the destruction of the other states not measured.

Using these concepts, quantum algorithms have been developed and discovered over the years, some of which significantly outperforming their classical counterparts. In [9], the first such algorithm was presented, now known as the Deutsch-Jozsa algorithm. Other famous examples include Shor's algorithm for factoring prime numbers [10], and Grover's algorithm for unstructured search [11].

Despite the theoretical algorithmic promises presented above, the field of quantum hardware for computation, however, has been slow to progress at the same pace. The first examples of experimental qubits involved NMR technology, where small qubit systems were built to execute simple quantum algorithms [12, 13, 14]. These prototypes were followed by similar experiments with trapped ion qubits [15], and soon after in superconducting flux qubits [16]. Since then, many other possible qubit technologies have been demonstrated for quantum computing, such as diamond cavities [17], quantum dots in silicon [18], cold atoms [19, 20, 21], and photonic quantum computing [22]. However, of these, currently superconducting qubits have proven to be the most scalable architecture choice, with companies such as Google [23], IBM [24], and D-Wave Systems [25, 26] building publicly-accessible platforms to access their quantum processing units (QPUs). These processors have been described as representative of the *noisy intermediate-scale quantum* (NISQ) era [27], describing larger, albeit still noisy (non-error-corrected) qubit systems. Each of these QPUs require different considerations specific to the design and implementation choices made during manufacturing, making them potentially more or less useful under certain conditions. However, they all use the same mathematical formulation and notation in order to execute quantum algorithms. The mathematical representation of the optimization problems presented in this thesis and the quantum algorithms used to solve them are presented next and will remain consistent for the remainder of this thesis.

1.2 Definitions and notation

1.2.1 Problem complexity

Combinatorial optimization is defined as selecting an optimal subset from a finite set of elements, usually subject to some objective. The work presented in this thesis is mainly focused on the tasks of solving such optimization problems using quantum optimization algorithms, in particular the quantum annealing algorithm. Conventionally, optimization problems are defined by their *problem class* (i.e., the type of problem) within a *complexity class* (i.e., complexity of the problems). These complexity classes are defined by evaluating the efficiency of solving specific problems using Turing machines (typically denoted using the symbol $\mathcal{T}$). For example, the aforementioned problem of finding a Eulerian path is solvable by a Turing machine in time polynomial in the graph size. However, a simply-stated related problem, the task of visiting every *vertex* exactly once in a given undirected graph, rather than its edges, currently does not have such a polynomial-time algorithm. Clearly, it is evident that by examining the efficiency of Turing machines in solving different combinatorial optimization problems (and related graph problems), different classes of problem complexity arise, and the field of studying the classification of problem using them is known as "complexity theory". A diagram with common classes based on Turing machines and their conjectured relations are shown in Fig. 1.2. The most pertinent complexity classes are defined briefly here:

- **P.** This problem class is defined as all decision problems which are solvable by deterministic Turing machines with resources (runtime) that scale polynomially with the size of the instance being solved. Typically, problems in this class are referred to as having "efficient" algorithms to solve them, and so efficiency is often equated to polynomial-time algorithms (also known as tractable problems).

- **NP.** The class of decision problems in which the verification of the correctness of a solution to a decision problem can be performed in polynomial time by a Turing machine. Equivalently, this problem class can be defined as the set of all decision problems which are solvable in polynomial time by *non-deterministic* Turing machines. Problems which can be used to represent all other problems in this class (with at most polynomial overhead) form a related class, known as **NP-hard**. Furthermore, problems which are NP-hard and themselves are in NP are referred to as **NP-complete**. It is important to note that this is not restricted to decision problems, and so many NP-complete decision problems have optimization versions which are NP-hard.

- **APX.** This class is defined as the set of NP optimization problems which have a polynomial-time approximation algorithm with bounded constant error. This is not to be confused with the set of optimization problems which have polynomial-time approximation schemes (PTAS), which is defined as optimization problems with polynomial-time approximation algorithms for *every* constant error bound (and thus are all contained within APX).

Furthermore, as in NP, there are APX-hard and APX-complete classes, defined as all problems that have approximation-preserving reductions, and problems which are both APX-hard and in APX, respectively.

Although Turing machines have been traditionally used because they are equivalent to every other model of computation, this does not necessary imply anything in regards to the efficiency of all models of computation. More recently, with the discovery of quantum algorithms and the invention of quantum computation, other problem classes have been developed, defined by the efficiency of quantum algorithms in solving them. These are defined by considering the existence of quantum Turing machines, quantum mechanical extensions of the original Turing machines. A diagram depicting the relationships between the quantum problem classes is in Fig. 1.2. The relevant quantum complexity classes are defined here:

- **BQP.** This class is defined as the set of decision problems which can be solved in polynomial time by a quantum Turing machine with an error probability bounded by 1/3. Since every classical circuit can be simulated with quantum circuits (and can be constructed efficiently), P is contained within this class.
- **QMA.** This is the class of problems which have a polynomial-time quantum circuit that can verify a given polynomial-size quantum state (or proof) with bounded error. As with NP, the set of problems to which all QMA problems are reducible to are known as **QMA-hard**, and those that are both QMA-hard and in QMA themselves are known as **QMA-complete**.

The relationships between many of the complexity classes defined above, both classical and quantum, are still unknown. For example, the question of whether NP is contained in P (referred to as $\mathrm{P} \overset{?}{=} \mathrm{NP}$) is an open fundamental question in computer science (first introduced in [28]), the consequences of which would collapse several of the sets shown in Fig. 1.2. Furthermore, the relations between the quantum complexity classes to the classical ones are also not fully known. For instance, although P is contained in BQP, there are problems solvable by BQP which are conjectured to not be in P. The most well-known example of this is the problem of finding prime factors, which is in NP and has a polynomial-time quantum algorithm (known as Shor's factoring algorithm [10]). Thus, there are still exciting and potentially ground-breaking theoretical results to be discovered.

The ultimate goal of the work presented in this thesis is not to settle these fundamental (and important) theoretical questions. Rather, the context of the research

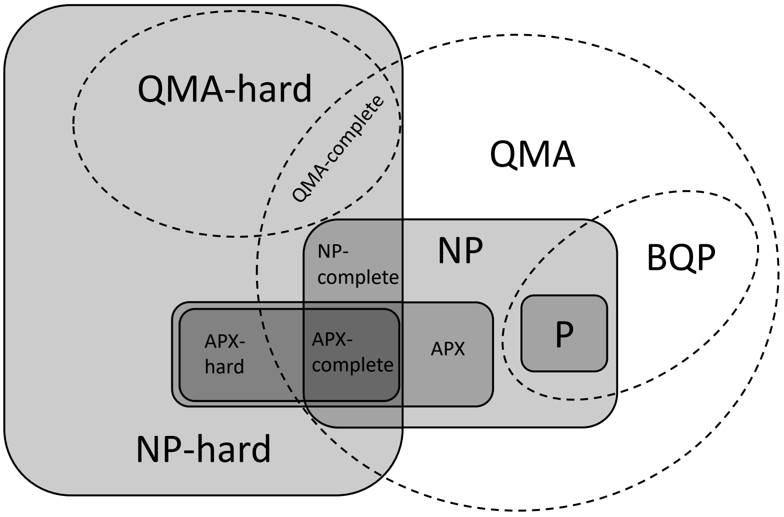


Figure 1.2: A diagram of complexity classes and their relationships, as explained in the text. Classical complexity classes are in rounded squares, and quantum complexity classes are in dashed ellipses.

was to assess the practical relevance of a specific class of quantum optimization algorithms (quantum annealing) implemented on state-of-the-art quantum hardware. The complexity classes defined above were used as a guide in order to better understand what conditions need to be fulfilled, and which problems need to be investigated, to meet such usefulness criteria.

1.2.2 Qubits and operators

The basic elements with which quantum algorithms are built are qubits and their respective unitary operators. The two qubit states which are used to define the vector space of operations are:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}. \tag{1.1}$$

These two states together form an orthonormal basis, called the *computational basis states*. In Dirac notation[1], the single qubit state (typically denoted as the wavefunction ψ) is represented using a linear combinations of the basis states:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{1.2}$$

Here, α and β, which are complex numbers, are the respective amplitudes of each basis state. Consistent with the theory of quantum mechanics, one can only observe each state with a finite probability. As such, the values of the amplitudes are constrained by:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{1.3}$$

Analogous to classical computation, these qubits are manipulated through the use of operators (or gates). In quantum mechanics, unitary operators are used, represented as matrices operating on the vector space of qubits. The central operators for quantum annealing algorithms are the Pauli spin matrices:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.4}$$

In quantum annealing, the Pauli operators are applied to sets of qubits in the quantum system in order to encode the optimization problem, and their magnitudes are varied throughout the computation to perform minimization.

1.2.3 Ising models and QUBO problems

Depending on the specific problems being solved, it is convenient to specify different bases for the target optimization problem. Some problems may use variables $s \in \{-1, 1\}$ called *spins*, while others may use binary variables $x \in \{0, 1\}$. These two are equivalent via a simple change of basis variables, $s = 2x - 1$. Objective functions with spin variables are represented as Ising spin-glass systems (also known as Ising models):

$$\mathcal{H}(s) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j, \tag{1.5}$$

[1]Dirac notation (also known as bra–ket notation) is a method to describe the linear algebra governing quantum mechanics. The derivation and background of this notation is beyond the scope of this thesis, but an introduction to the topic can be found in [29]

where h_i is the linear weight (also known as bias) of each spin variable s_i, and J_{ij} is the quadratic interaction term (also known as coupling strength) for each spin pair. Equivalently, one could specify the objective function using binary variables as a quadratic unconstrained binary optimization (QUBO) problem:

$$\text{Obj}(x, \mathbf{Q}) = x^T \cdot \mathbf{Q} \cdot x. \tag{1.6}$$

Here, x is a vector of N binary variables, and $\mathbf{Q}$ is an $N \times N$ symmetric interaction matrix. Using the property $x_i \cdot x_i = x_i^2 = 1$ for binary variables, the QUBO objective can be re-written in the same form as the Ising model, with the diagonal elements of $\mathbf{Q}$ being the linear terms, and the off-diagonal elements being the quadratic terms[1]. The task performed by the quantum optimization algorithm is to minimize the objective functions above.

1.3 Families of combinatorial optimization algorithms

Given the complexity of many hard combinatorial optimization problems, various algorithms have been developed in an attempt to solve them, both to optimality and approximately. In this thesis we are concerned with specific kinds of algorithms in order to both test quantum annealing directly and to contextualize the results via comparisons to other algorithms. We briefly define and explain the different families of algorithms below.

Exact algorithms. Algorithms which have deterministic endpoints that are guaranteed to provide optimal solutions to problems are known as exact algorithms. Because of the open question of P vs. NP, currently all exact algorithms for NP-complete and NP-hard optimization have exponential runtime in the worst case. Nonetheless, a variety of exact algorithms are used for real-world combinatorial optimization problems, such as CPLEX [30] and Gurobi [31], among others.

Classical (meta)heuristics. Algorithms implemented on classical computers which exploit probabilistic (or non-deterministic) routines within them to solve specific problems are referred to as *heuristics*. More general algorithms which use generic (and often parameterized) routines to solve arbitrary optimization problems

[1]Describing the matrix $\mathbf{Q}$ as an upper- or lower-triangular matrix is also common, since all terms $x_i \cdot x_j = 0$ if either x_i or x_j are 0.

are known as *metaheuristics.* Since (meta)heuristics can solve some instances of problems in NP, they are often used as benchmarks in comparative analyses. Examples of well-known metaheuristics are genetic/evolutionary algorithms [32], simulated thermal annealing [33], particle swarm optimization [34], and many more.

Quantum (meta)heuristics. Equivalently, one can define heuristics and metaheuristic algorithms for quantum computers, which in turn implement probabilistic routines in quantum circuits or Hamiltonians. The quantum annealing algorithm is one such metaheuristic for quantum optimization [35]. The quantum approximate optimization algorithm (QAOA) is a similar quantum optimization metaheuristic for gate-model quantum computers [36].

Hybrid quantum-classical algorithms. Specifically in the near-term era for quantum hardware, the limiting factor in implementing quantum algorithms is often the number of available qubits. As a consequence, the concept of hybrid algorithms has been developed, in which quantum algorithms are used as an inner loop to perform a specific task set by a classical algorithm acting as an outer loop. There are many possible ways to construct such quantum (meta)heuristics, both for quantum annealers and gate-model quantum computers [37, 38, 36, 39]. In the context of this thesis, we refer to any classical algorithm which offloads a sub-task to an adaptive quantum inner-loop as a *hybrid algorithm*, to distinguish from more traditional parameter-tuning techniques.

1.4 Outline

We now outline the remainder of the work in this thesis. Each of the chapters below represents a summary of one or more individual projects addressing the respective research question motivating the research. To address the overall goal of applying quantum annealing to industrial combinatorial optimization problems, we break the topic down into a series of main research questions, each motivating and expanding upon a different aspect in the topic. The chapters are built upon peer-reviewed publications, listed at the end of the description of each chapter.

Q1. How are combinatorial optimization problems expressed for quantum annealers?

Quantum annealers are special-purpose quantum hardware which implement the quantum annealing algorithm to perform optimization, and share some similarities to classical metaheuristic optimization algorithms. However, the process required to perform optimization using quantum annealing has multiple steps which have no analogy in classical optimization. The impact of these steps on the hardware's ability to solve problems is not obvious. To answer **Q1**, Chapter 2 presents the theoretical motivation for the quantum annealing algorithm, and the particularities of its manifestation in hardware. Then, the steps required in order to formulate and submit optimization problems to quantum annealers in practice are reviewed.

[1] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt (2022). Quantum Annealing for Industry Applications: Introduction and Review. *Rep. Prog. Phys.* **85** 104001.

Q2. What is the impact of hyperparameters, tuning, and physical limitations of quantum hardware on algorithmic performance?

Using a representative canonical NP-hard problem (the maximum independent set problem), Chapter 3 explores the setting and tuning of QPU parameters. The trade-off between algorithmic runtime and performance with respect to this parameter tuning process is investigated. To answer **Q2**, best practices and algorithms developed to tune these parameters are implemented and tested on a D-Wave 2000Q QPU. Where relevant, the physics governing the QA algorithm are used to motivate the appropriate schemes for choosing and tuning parameters. The ability to solve optimization problems in practice using these methods is demonstrated. The results from the analysis allow general guidelines and conclusions to be drawn in regards to combinatorial optimization using QA.

[1] S. Yarkoni, T. Bäck, and A. Plaat (2018). First results solving arbitrarily structured Maximum Independent Set problems using quantum annealing. In *Proceedings of the 2018 IEEE Congress on Evolutionary Computation*, CEC '18, Rio de Janeiro, Brazil, pp. 1–6.

[2] S. Yarkoni, H. Wang, A. Plaat, T. Bäck (2019). Boosting Quantum Annealing Performance Using Evolution Strategies for Annealing Offsets Tuning. In: Feld S., Linnhoff-Popien C. (eds) Quantum Technology and Optimization Problems, QTOP '19. Lecture Notes in Computer Science, vol 11413. Springer, Cham, pp. 157–168.

Q3. How are real-world problems different from academic problems, and how does this affect the performance of quantum annealing?

Real-world optimization problems often require mixtures of either variable types, constraints, or other such terms which make them more difficult to represent relative to the canonical problems presented thus far. Chapter 4 introduces methods to formulate such combinatorial optimization problems for quantum annealing, with special consideration towards mathematical modeling of problems. This includes the addition of constraint-based optimization, and the impact such problems have on performance is investigated. The potential and limitations of quantum annealing for solving real-world optimization problems are explored using a variety of use-cases and datasets.

[1] Sheir Yarkoni, Andrii Kleshchonok, Yury Dzerin, Florian Neukart, Marc Hilbert (2021). Semi-supervised time series classification method for quantum computing. Quantum Machine Intelligence **3**, 12.

[2] Sheir Yarkoni, Andreas Huck, Hanno Schulldorf, Benjamin Speitkamp, Marc Shakory Tabrizi, Martin Leib, Thomas Bäck, and Florian Neukart (2021). Solving the Shipment Rerouting Problem with Quantum Optimization Techniques. In: International Conference on Computational Logistics (ICCL 2021): Computational Logistics, pp. 502-517.

Q4. Can we use hybrid quantum-classical to overcome some of the deficiencies of current quantum hardware?

Limited sizes of currently-available quantum hardware is a bottleneck in many application areas. Therefore, Chapter 5 utilizes the previously presented work, and introduces the concept of hybrid quantum-classical algorithms. Various techniques and methods for hybrid algorithms are both reviewed and implemented, developing the necessary functionality to build end-to-end applications with quantum annealing. These algorithms are testing using real-world data and live applications are built using quantum annealers and hybrid algorithms. The capabilities of quantum annealers and hybrid algorithms to solve industrial optimization problems are explored and presented.

[1] Sheir Yarkoni, Florian Neukart, Eliane Moreno Gomez Tagle, Nicole Magiera, Bharat Mehta, Kunal Hire, Swapnil Narkhede, Martin Hofmann. (2020). Quantum Shuttle: Traffic Navigation with Quantum Computing.

In: Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software (22-30).

[2] Sheir Yarkoni, Alex Alekseyenko, Michael Streif, David Von Dollen, Florian Neukart, and Thomas Bäck (2021). Multi-car paint shop optimization with quantum annealing. In: 2021 IEEE International Conference on Quantum Computing and Engineering (QCE): 35-41.

Finally, Chapter 6 discusses the future of quantum annealing for optimization, draws conclusions, and examines the lessons learned from the work presented.

CHAPTER 2

Combinatorial optimization and quantum annealing

Metaheuristic algorithms are often used in practice to solve a variety of traditionally difficult optimization problems. Their power derives from the tunability of their parameter sets, often providing practitioners with trade-offs between global search space exploration and local optimization. Annealing algorithms in particular are popular due to their strong theoretical motivation from physical processes and relative ease of implementation [33]. In this chapter we introduce the basic concepts of the quantum annealing algorithm, from both the theoretical and practical aspects, and present the methodology with which optimization problems are solved using quantum annealing hardware.

2.1 Adiabatic quantum computing and the adiabatic theorem

The Adiabatic Quantum Computing (AQC) model[1] is a computational model for quantum computing which exploits the *adiabatic theorem*: given a quantum system in its ground (minimum) energy state, if the governing Hamiltonian is changed "sufficiently slowly", then the quantum system remains in its instantaneous ground state. It has been shown that, given a register of qubits, this adiabatic theorem can be used in order to simulate quantum Hamiltonians and perform algorithmic computation [40]. Furthermore, it has been shown that this AQC

[1] This model of computation is also known as the Adiabatic Quantum Optimization model, abbreviated AQO.

model is polynomially equivalent to the gate-model of quantum computing, by showing how quantum Hamiltonians can be constructed to simulate arbitrary quantum circuits [41]. However, the ability of AQC to solve (or simulate) arbitrary problems is not perfectly understood. The notion of "sufficiently slowly" depends on the both the specific Hamiltonian and the conditions of evolution, and can be difficult to compute. A generic time-dependent Hamiltonian for adiabatic evolution is given by:

$$\mathcal{H}(t) = A(t)\mathcal{H}_i + B(t)\mathcal{H}_f. \tag{2.1}$$

Here, $\mathcal{H}_i$ is the Hamiltonian in which the system is intialized (referred to as the *initial* or *driver Hamiltonian*), and $\mathcal{H}_f$ is the Hamiltonian at the end of the evolution (referred to as the *final* or *target Hamiltonian*). The boundary conditions on the relative magnitudes of each Hamiltonian are given by $B(t = 0) = 0$ and $A(t = \tau) = 0$, for an evolution on timescale τ. Thus, one could construct $\mathcal{H}_f$ to represent a quantum circuit and use Eq. (2.1) to perform AQC. A visualization of the transition in magnitudes is shown in Fig. 2.1.

The theoretical motivations for AQC can also be used in the context of optimization. Instead of setting $\mathcal{H}_f$ to represent a quantum circuit, it can be used as a Hamiltonian representation of an optimization problem. Thus, successfully evolving to a ground state of $\mathcal{H}_f$ can solve complex combinatorial optimization problem with Hamiltonian representations. In realizable quantum hardware it is not always possible to evolve adiabatically (as any physical system is always coupled to its environment), but it is known that even in the absence of perfect adiabaticity, a quantum system that is evolved "sufficiently slowly" maintains proximity to its ground state [42]. Here we consider only models in which time evolution dependence can be parameterized by a single variable $t \in [0, \tau]$. This parameter is also sometimes normalized by τ, and referred to as *normalized time*, $s = t/\tau \in [0, 1]$. Generally, it is known that the magnitude of τ required to remain near the grounds state is related to the difference between the two lowest energy states, E_0 and E_1. This point at which this energy difference occurs is called the *avoided crossing* or *minimum gap*, and is visualized in Fig. 2.2. The timescale of evolution is typically expressed via the approximation:

$$\max_{t_i \leq t \leq t_f} \frac{\langle \mathcal{H}_f(t) | \frac{d\mathcal{H}(t)}{dt} | \mathcal{H}_i(t) \rangle}{|E_1(t) - E_0(t)|^2} \ll \tau. \tag{2.2}$$

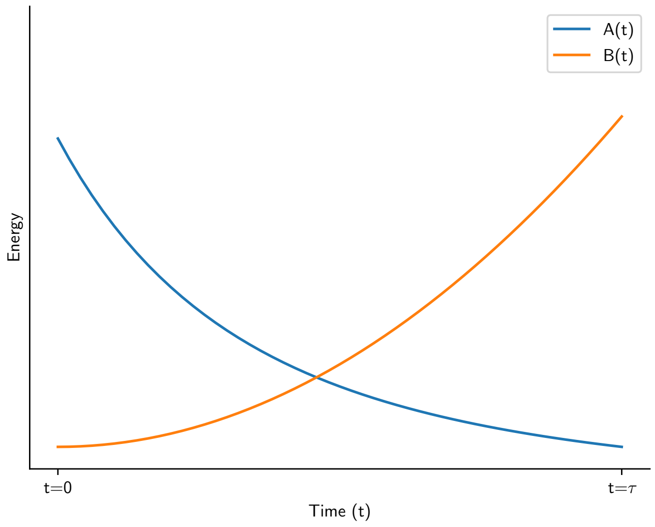


Figure 2.1: Generic evolution of a Hamiltonian as dictated by $A(t)$ and $B(t)$ in the AQC model. In software (meaning, when simulating quantum annealing), the shape of these functions can be programmed and controlled. In hardware, the shapes are hard-coded via the control electronics of the quantum processor and its calibration. To allow for full manipulation of these curves would require controls which are beyond the current capabilities of the technology.

Thus, if the final Hamiltonian $\mathcal{H}_f$ represents a target optimization problem, then an evolution on timescale τ can be used to solve it. However, in practice, calculating $|E_0(t) - E_1(t)|$ requires knowledge of the entire eigenspectrum of the Hamiltonian $\mathcal{H}$ (in the worst case), and is therefore NP-hard in itself.

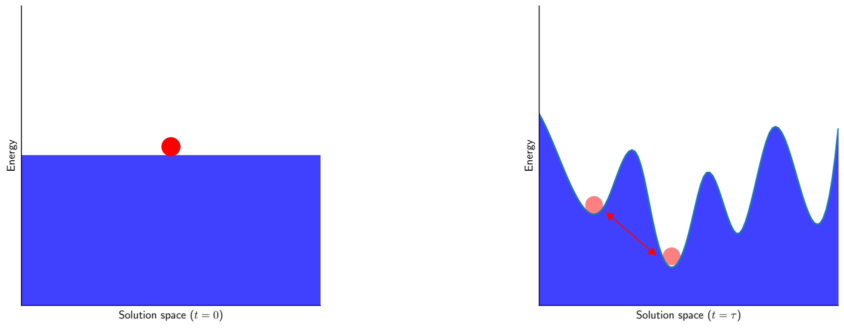
2.2 Theory of quantum annealing

Originally described as a metaheurstic in classical software used to solve combinatorial optimization problems, quantum annealing (QA) is closely related to AQC [35]. Analogous to classical simulated thermal annealing, quantum fluctuations are used in order to tunnel through energy barriers in a combinatorial landscape, as opposed to thermally-assisted hops in simulated annealing. In quantum annealing, these fluctuations induce *quantum tunneling* effects, where the wavefunction is split across energy barriers to different energy minima. At the end of the evolution (ideally) the wavefunction is concentrated at global optima of the energy landscape. However, instead of initializing the algorithm with random initial configurations (like in thermal annealing) the QA algorithm is initialized by setting qubits' states to a simple ground state (generally a superposition over all states), and is evolved over time to target a final Hamiltonian $\mathcal{H}_f$. Quantum annealing can be viewed as a subclass of AQC, where the Hamiltonian type used for $\mathcal{H}_f$ is fixed, and adiabaticity is not guaranteed. Thus, the result is a heuristic quantum optimization algorithm that has a non-zero probability of returning a candidate solution to an optimization problem, rather than a deterministic quantum simulation. In its simplest form, quantum annealing is implemented using the *transverse-field Ising Hamiltonian*:

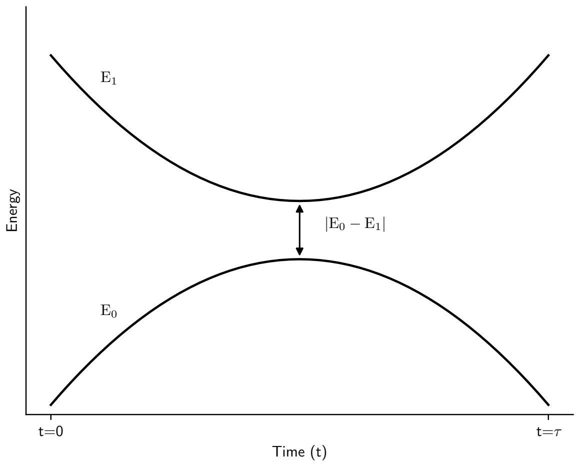
$$\mathcal{H}(t) = A(t)\mathcal{H}_i + B(t)\mathcal{H}_f = A(t)\left[\sum_i \sigma_i^x\right] + B(t)\left[\sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z\right], \quad (2.3)$$

where σ_i^x and σ_i^z are the Pauli-X and -Z spin matrices applied to the ith qubit, and h_i and J_{ij} are used to define the spin-glass representation of the optimization problem. Here, as in Eq. (2.1), $A(t)$ and $B(t)$ are time-dependent functions that dictate the magnitudes of each term in the Hamiltonian. Since it is known that finding the ground state of the 2D Ising spin-glass problem is NP-hard [43], this Hamiltonian definition is sufficient to address many interesting combinatorial optimization problems. However, as with AQC, the probability of obtaining (or remaining close to) a ground state at the end of the computation is bound by the evolution timescale τ and the smallest energy difference between the ground state and first excited state in Eq. (2.2). As this evolution time is a parameter to the quantum annealing algorithm, it is typically referred to as the *annealing time*, t_a. The dependence of QA's performance on time is a known problem in quantum optimization. Particularly, how optimal (minimal) t_a scales as a function of problem size for different problem classes is generally regarded as the true scaling of the algorithm, as the goal is to relate the evolution time of QA to observing the ground state of the system.

Many interesting experiments have been performed to assess the conditions of scaling for quantum annealing, both in simulation and in hardware. Early work with quantum Monte Carlo simulations showed the ability of QA to solve combinatorial optimization problem in some limits [44]. Additional work (in quantum Monte Carlo simulations) investigated the correlation between exponentially small energy



(a)



(b)

Figure 2.2: **(a)** Schematic of the evolution of the energy landscape for a combinatorial optimization problem. On the left, all states are minima, representing the equal superposition of the two basis states for $\mathcal{H}_i(t = 0)$. The red circle represents a single state in this space. On the right, a visualization of a $\mathcal{H}_f(t = \tau)$, where there are well-defined minima at the end of the evolution. The red circle represents a state corresponding to an optimum, having tunneled through an energy barrier from a higher-energy state. **(b)** Here, a diagram of the minimum gap shows the difference between the two lowest energy levels in the quantum system (as a function of evolution time), which defines the adiabatic condition of evolution.

gaps and the probability of observing ground states using maximum independent set instances [45]. A similar experiment testing the scaling of annealing time in the presence of thermal excitations was performed using QA hardware in [46]. For a complete assessment of state-of-the-art theory and a thorough discussion of the scaling of QA in general, see [42].

2.3 Binary and combinatorial optimization

We now focus on the representation of general optimization problems in either of the admissible binary forms so they can be solved using quantum annealing. The first model is the classical Ising spin-glass model:

$$H_f = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j, \tag{2.4}$$

where $s_i \in \{-1, 1\}$ are the individual spin variables, h_i are the real-valued linear weights, and J_{ij} are the so-called quadratic interaction terms. Alternatively, for binary variables $x_i \in \{0, 1\}$ the quadratic unconstrained binary optimization (QUBO) form is used:

$$\text{Obj}(\mathbf{Q}, x) = x^T \cdot \mathbf{Q} \cdot x, \tag{2.5}$$

where $x = (x_0, x_1, \ldots, x_{N-1})$ is a vector of N binary variables, and $\mathbf{Q}$ is an $N \times N$ real-valued matrix of interaction terms (diagonal elements are the variable weights, and off-diagonal elements are the quadratic interactions). Many canonical examples of combinatorial optimization have been studied extensively in literature using both of these forms, and often for research in computer science and quantum computing: the traveling salesperson problem (and related vehicle routing problem) [47], max-cut [36], satisfiability [48], graph coloring [49] and more. In general, all of these analyses in combinatorial optimization exploit binary programming, also known as $0-1$ binary programming, or binary optimization (and in some cases pseudo-Boolean optimization). The task is to assign the optimal value for each binary variable in a set such that a particular objective function is minimized. Originally, as one of Karp's 21 NP-complete problems, the objective function was represented as binary integer problem [50]. Formally, we are interested in a more general formulation, namely *pseudo-Boolean objective functions.* We define these as a family of functions whose domain is the set of N binary variables, which are then mapped to a real number:

$$f : \mathbf{B}^N \to \mathbb{R}. \tag{2.6}$$

As an NP-hard problem itself, Boolean optimization can be used to represent all other NP-hard problems, and therefore is sufficient to describe these canonical combinatorial optimization problems (not just binary optimization problems), with some polynomial overhead. Therefore, the initial step for using QA in practice is that of problem definition: the objective function of the optimization problem must be represented entirely with binary variables.

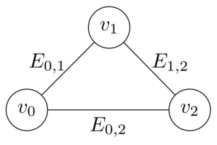
It is both common and useful in the context of quantum annealing (and quantum computing in general) to study combinatorial optimization from a graph representation perspective. Every Ising/QUBO problem can be represented as an undirected weighted graph, with every variable represented by a vertex $v_i \in V$ and pairwise interaction term represented by an edge $e_{ij} \in E$ between nodes v_i and v_j. In the Ising (QUBO) model, each h_i in Eq. (2.4) (diagonal elements of Q in Eq. (2.5)) is the corresponding weight for v_i in the graph representation, and likewise the J_{ij} (off-diagonal elements of Q) for e_{ij}. A simple demonstration of the equivalence between QUBO, Ising, and weighted undirected graphs is shown in Fig. 2.3.

$$\mathcal{H} = h_0 s_0 + h_1 s_1 + h_2 s_2 + \\ J_{0,1} s_0 s_1 + J_{0,2} s_0 s_2 + J_{1,2} s_1 s_2$$

(a)

$$\text{Obj} = x^T \begin{pmatrix} Q_{0,0} & Q_{0,1} & Q_{0,2} \\ 0 & Q_{1,1} & Q_{1,2} \\ 0 & 0 & Q_{2,2} \end{pmatrix} x$$

(b)



(c)

Figure 2.3: **(a)** A simple objective definition of a three-variable Ising model. **(b)** Equivalent representation of a QUBO matrix in upper-triangular form. The terms in the QUBO matrix which correspond to (a) can be derived using the change of basis $s_i = 2x_i - 1$. **(c)** A graph network representation of same system, with variables as nodes and interaction terms as edges. The specific weights in the graph depend on the choice of basis, (a) or (b).

Collectively, these are referred to as *binary quadratic models* (BQMs). To generalize binary models to arbitrary variable types, specific mathematical techniques are used in practice. The relevant approaches used in the studies of this thesis (and the related works) are introduced next.

2.4 Generalizing QUBO and Ising

Although binary optimization is NP-hard, the polynomial overhead incurred when modeling more general problems means that certain classes of optimization are better (or conversely, less) suited for quantum annealing. Furthermore, different classes of real-world optimization problems require different categories of variables. Here we review the types of variables and general terms that are often used for quantum annealing in practice.

2.4.1 Constrained optimization

In QUBO/Ising form, the real-valued constant variable coefficients are unconstrained. As quantum annealing is an analog process, there is also no way to explicitly constrain the variables during the evolution. Therefore, all equalities (and inequalities) must be implemented in a quadratic objective form in order to be included in the model. This means that constrained optimization cannot be performed directly, but is instead addressed by including an additional term scaled by a constant (known as a penalty factor) to separate *feasible* and *infeasible* solution spaces in the optima of the optimization problems. One example of an important constraint– the one-hot constraint, where exactly one binary variable is 1 and the rest are 0– is transformed as follows (with binary variables x_i):

$$\sum_i x_i = 1 \longrightarrow \left(1 - \sum_i x_i\right)^2 = 0. \tag{2.7}$$

The left hand side of the equation above represents the equality constraint with a simple summation over the variables. Adding these factors to a QUBO would result in a minimum where all $x_i = 0$, obviously violating the purpose of the one-hot constraint. Thus, the right hand side is implemented to constrain the variable space by expanding the square, which has only linear and quadratic terms that can be added to the QUBO objective function. This now has the correct minimum with one binary variable set to 1 and the rest to 0. Arbitrary linear constraints are expressed in QUBO form as:

$$\lambda \left(b - \sum_i a_i x_i \right)^2 = 0, \tag{2.8}$$

where λ, b, a_i are all real-valued numbers, and x_i are a subset of all binary variables in the problem. Because the contribution of a satisfying configuration of the binary variables is zero, the cost of violating this condition can be set by the parameter λ. In pure constraint satisfaction problems, this isn't strictly necessary, since (by addition) the individual contributions of the constraints are zero, and thus the objective function value of a satisfying solution will also be zero. When mixing optimization terms and constraints in the objective function, it is necessary to set λ appropriately such that it is never energetically favorable to violate a constraint in favor of reducing the value of the objective function.

These concepts can be further extended to address inequalities as well, by introducing *slack variables*. In general, linear inequality constraints (on binary variables x) are presented as:

$$\sum_i a_i x_i - b \leq 0, \tag{2.9}$$

where a_i, b are integer coefficients. To transform this inequality to QUBO, we must create an degenerate objective function such that all minima satisfy the inequality. We start by adding auxiliary variables such that the total slack is accounted for:

$$\lambda \left(\sum_i a_i x_i + \sum_j^W w_j y_j - b \right)^2 = 0. \tag{2.10}$$

The number of slack variables W and their coefficient w_j can be derived from the coefficients a_i and b, where at most $W = b$ slack variables are needed[1]. Now the equality constraint is equivalent to the original inequality constraint, since it can be fulfilled for all values of the binary variables x_i satisfying the original inequality by choosing some appropriate values for the slack variables, which remain unconstrained. For values violating the original inequality constraint, the equation can never be fulfilled regardless of the values of the slack variables. The equality constraint written in the quadratic form is then added as a penalty term to the QUBO cost function.

[1]In general, it is possible to use binary encoding schemes such that only $\lceil \log_2(n) \rceil$ auxiliary qubits are needed.

2.4.2 Discrete variables

Non-binary discrete variables can also be represented in Ising/QUBO, and have been investigated in the context of quantum annealing in scheduling problems [51, 52] and graph colouring [53, 54, 55], among others. Here, each discrete variable can be encoded by using multiple qubits to represent a logical integer variable subject to a single constraint [56, 57, 58]. Binary encoding, analogous to classical binary encoding, is efficient in the number of qubits in the sense that one can encode d discrete states using only $\lceil \log_2(d) \rceil$ qubits. However, the binary encoding is not used much in practice for application problems, since the interactions necessary to enforce the validity of the encoding as well as the realization of the couplings between logical variables are rather complicated to implement with the binary encoding. Additionally, this encoding has been investigated in previous works and has been shown to be detrimental to the QPU's ability to find ground states [59, 49].

The one-hot encoding is a standard technique where a logical variable with d possible states is represented by d qubits. Each qubit's state corresponds to one possible value of the discrete variable if set to $|1\rangle$, and all other qubits are $|0\rangle$. The corresponding constraint is typically implemented as quadratic interaction terms between all d qubits, shown in Eq. (2.7). The advantage of this technique is the fact that the one-hot case can be trivially extended to the $k-$hot case, where a subset of exactly k qubits are $|1\rangle$ and the rest $|0\rangle$– however, this requires all-to-all connectivity between the qubits. This is expressed in the form:

$$\left(k - \sum_i x_i\right)^2 = 0. \tag{2.11}$$

An alternative to the one-hot constraint is the *domain-wall encoding*, which can encode discrete variables with one fewer qubit per variable, i.e., $d-1$ qubits for d states, and requires only a chain of couplings rather than all-to-all. It has been shown to be more efficient in different test problems [59, 49, 60]. However, the main drawback of this method is that it cannot extend directly to the $k-$hot constraint using the same technique. Rather, k copies of the $d-1$ chain must be connected (non-trivially) to enforce the same combinatorial landscape, which is no longer efficient in the number of qubits.

2.4.3 Continuous variables

Representing continuous numbers using binary variables is also possible using QUBO/Ising, although due to the binary encoding necessary it is not often used in practice. Much like in classical computing, these numbers are represented by binary encoding schemes. A single decimal variable $\tilde{x}$ with N bits of precision would be encoded as follows (using binary values x):

$$\tilde{x} = \sum_{i=0}^{N-1} 2^i x_i. \tag{2.12}$$

There are several drawbacks to using this encoding scheme. Firstly, this requires high precision in encoding the optimization problem, which makes the solution landscape more difficult to explore due to thermal noise in quantum hardware. Furthermore, the individual energy levels (or local optima) are exponentially spaced, making them more less likely to be explored. Lastly, the connectivity between the qubits is again quadratic, increasing the density of the problem being solved. However, expressing continuous variables is sometimes unavoidable, and this technique has been used in practice [61].

2.5 Quantum annealing in hardware

The core of a quantum processing unit (QPU) is a layout of qubits which are connected via a system of couplers. The QPUs used in the research in this thesis were from D-Wave Systems, which implement superconducting flux qubits to build their processors [62]. While the component physics is beyond the scope of this thesis, a technical description of the quantum processors can be found here [25]. In the absence of logical encoding schemes, the physical layout of the qubits is fixed in the QPU and is referred to as the *hardware graph*, denoted by U. This naming convention comes from the graph-theoretical description of the QPU, where each qubit (coupler) is represented by a node (edge) in an undirected graph. The exact topology of the hardware graph dictates the structure of graphs and possible $\mathcal{H}_f$ that can be represented natively by the qubits' connectivity. The QPU topology is an engineering artifact arising from the design choices for the QPU. In general, due to engineering difficulties, connectivity between qubits comes at the cost of the total number of qubits in the QPU. Due to various technological limitations in manufacturing, calibration, or other anomalies, a small portion of the qubits and couplers in the QPUs may be defective and hence not programmable. The percentage of qubits and couplers that remain functional once fully exposed to users is referred to as the *hardware yield.* For D-Wave QPUs, the qubit yield is typically above 97% for current processors [63]. The graph representing the functional qubits and couplers is referred to as the QPU's *working graph.*

The two topologies used for studies presented in this thesis are the D-Wave 2000Q and Advantage QPUs. The topology of the D-Wave 2000Q (and earlier generations of QPUs) are referred to as *Chimera* topology. These graphs are composed of a 2D lattice of small complete bipartite graphs (each called a tile, or unit cell). These graphs are denoted as C_X (where X is the length of one side of the square lattice, for a total of X^2 tiles), and each tile is in itself a $K_{N,N}$ bipartite graph (full connectivity between left and right partitions). The D-Wave 2000Q has a size of C_{16} and $K_{4,4}$ tiles. Each qubit in the Chimera topology has six couplers, where four couplers are inside the unit cell and two are to different unit cells. The Advantage QPUs implement a novel topology, called *Pegasus*, which differs in two major aspect from the Chimera graph: the graph degree is 15, and the hardware graph is not bipartite. Instances of the Pegasus topology that contain $N \times N$ unit cells are referred to as P_N and consist of $24N(N-1)$ qubits. This results in a significantly more complex structure, allowing for denser graph structures of $\mathcal{H}_f$ to be solved by the QPUs, and hence more difficult optimization problems (a technical description comparing Chimera and Pegasus can be found in Ref. [64]).

2.5.1 Minor-embedding for fixed topologies

Obviously not all interesting optimization problems can be defined by the hardware graph Chimera or Pegasus directly. However, it is possible to reformulate arbitrarily-structured Hamiltonians $\mathcal{H}_f$ into a new hardware-compatible $\mathcal{H}'_f$ via the technique of graph *minor-embedding*, a well-studied problem in graph theory (see, e.g. [65]). This process produces a mapping between one graph to another such that the relevant topological properties of the original graph are preserved. For QA, this involves encoding logical problem variables, or nodes in a graph, as *chains* of multiple physical qubits on a QPU so that they act as a single logical qubit [66, 67]. Different families of topologies may produce very different structures in hardware for the same input graph represented by $\mathcal{H}_f$. The impact of this, specifically in terms of canonical problems and the various chain requirements when embedding certain graph families in both Chimera and Pegasus graphs are explored in [64].

There exist several difficulties when using minor-embedding techniques. First, deciding whether a graph can be minor-embedded into another is a known NP-complete problem, and so polynomial-time heuristics (or deterministic algorithms on constrained subspaces) are used in practice [68, 69]. Secondly, the use of embeddings requires an additional set of constraints to be imposed on the qubits representing a logical qubit (the magnitude of this constraint is called the *chain strength*). These enforce that the minimum energy of $\mathcal{H}'_f$ is obtained only when all physical qubits representing a logical qubit are in the same state. It has been shown that the magnitude of the minimum chain strength increases with the degree of the graph of $\mathcal{H}_f$ [67, 70], which can create distortions in the resulting Hamiltonian $\mathcal{H}'_f$. Generally, determining the best suited chain strength is a nontrivial task, and it is one of the parameters explored in more depth in the results of this thesis.

2.5.2 Noise and mitigation strategies

Building a quantum processor inherently involves the implementation of an open quantum system– meaning, the qubits can never be truly perfectly isolated from their operating environment. This makes it difficult to draw universal conclusions about the power of programmable QA [71]. However, many investigations into how these quantum system interact with the environment (known as background noise) have been published in the past. Dickson et al. [46] were able to demonstrate the effects of thermal noise and diabatic transitions with D-Wave QA processors. In this context, resilience against background noise was defined as the ability of the quantum system to yield the correct solution with acceptable probability within a time comparable to the closed-system adiabatic timescale. It was demonstrated that in the limit of weak coupling to the environment (i.e., relatively low levels of thermal noise), annealing diabatically across the minimum gap did not hinder QA, but rather enhanced its performance. This was demonstrated by manually raising the operating temperature of the chamber in which the QPU was operating, thus allowing for a controllable amount of noise to perturb the system. The specific point demonstrated was that thermal noise does not necessarily inhibit obtaining ground states at the end of the annealing, even with small minimum gaps and non-adiabatic conditions. However, the caveat remains that the annealing times need to be sufficiently long, and the system must be only weakly coupled to the environment[1]. Additional experimental studies of quantum annealing processors have investigated the effects of noise and its role in computation of solutions to optimization problems. It has been demonstrated that finding quantum speedups may be elusive in quantum annealing hardware with fixed topologies (tested on spin-glass instances) [72]. Furthermore, it has been shown in such tests that classical optimization algorithms can often match and exceed the performance of quantum annealing hardware on such test instances due to physical limitations of the devices [73]. Nonetheless, it has also been demonstrated that small clusters of qubits do indeed remain coherent for some duration of the quantum annealing protocol, which can contribute to solving specific kinds of (contrived) optimization problems [74]. The extent to which this contributes to hardware performance for large-scale problems remains an open question.

Noise can also directly perturb problem formulations in the analog system, i.e., $\mathcal{H}_f$. The Hamiltonian parameters are subject to noise that can be modeled using a Gaussian distribution over the terms in the Hamiltonian. This noise (specifically, thermal noise whose timescale is much slower than the annealing dynamics) may cause a reshuffling of the energy levels in the problem Hamiltonian, which results in an incorrect encoding of the logical optimization problem in the eigenstates of the Hamiltonian. There are multiple ways to mitigate some of the effects of noise in quantum annealing. The most common technique is *spin-reversal transforms*, which are used to effectively average over the asymmetric final Hamiltonian distortion due to noise. This is done by creating random spin vectors $S = \{-1, 1\}^N$ and recalculating a new $\mathcal{H}'_f$ subject to this shift, where $h'_i = h_i \cdot S(i)$, and $J'_{ij} = J_{ij} \cdot S(i)S(j)$. These transformations do not change the distribution of ground states in the system, only the signs of a subset of terms. This new Hamiltonian $\mathcal{H}'_f$ is then sampled using a QPU. To recover the samples subject to the original Hamiltonian $\mathcal{H}_f$, the solutions are multiplied by the vector $S(i)$. Typically, many such transforms are used when solving a single $\mathcal{H}_f$. While this method is effective in reducing the error due to static noise, each new $\mathcal{H}'_f$ results in another programming cycle, increasing the total wall-clock time of using QA. Furthermore, it has been shown that increasing the number of transforms has

[1]Due to thermal relaxation, in the worst case the time to reach equilibrium in an open quantum system can grow orders of magnitude with respect to the time needed in a closed system.

diminishing returns for a fixed number of samples [75]. However, because of the technique's effectiveness in averaging over the static noise, this method is often used in practice.

2.5.3 Workflow of solving problems with QPUs

In QA, as implemented in quantum hardware by D-Wave Systems, the process of solving a combinatorial optimization problem can be divided into discrete stages. It is important to realize that very little in the procedure can be changed, and in practice the freedom is in the tuning of parameters exposed to users within this system. It is then up to the users of QA to interpret the results relative to the problem. In the context of optimization, we can conceptually model the QPU as a black-box optimization algorithm with very specific parameters and constraints. The step-by-step process is as follows:

- **Definition of a QUBO or Ising formulation and graph representation.** QUBO problems and Ising models have become the standard input format for quantum annealers, to which the optimization problems of interest are converted. This problem is represented as an undirected weighted graph, where each node represents a variable and each edge denotes the interaction term between a pair of variables. Thus the problem statement is given as finding the correct assignment of either $\{0, 1\}$ or $\{-1, 1\}$ (depending on the choice of basis) to minimize the quadratic objective function.
- **Minor-embedding.** The logical interaction graph is translated to the physical hardware graph of the QPU. It is necessary to select sets of physical qubits to represent a single logical node and to identify the couplings between the physical qubits to realize the correct interactions between the logical variables. Provided the correct parameters are chosen to enforce the chains of physical qubits, this procedure does not alter the minimum energy solution landscape, and the QPU's ability to optimize the embedded problem solved the original optimization problem of interest.
- **Programming and initialization.** Programming the quantum annealer requires setting the parameters that define the embedded problem to be solved, $\mathcal{H}_f$. This involves setting the weights for each qubit biases (controlling the magnetic field acting on the qubit) and coupler strengths (the interaction between qubits). To initialize the system, the qubits are set to a superposition

of the two computational basis states. This is the lowest energy configuration of the easy-to-implement initial Hamiltonian.

- **Annealing process.** In this step, the Ising model is solved. The system transitions from the initial to the final Hamiltonian according to predefined annealing functions in an attempt to minimize the energy. The way that the functions $A(t)$ and $B(t)$ are evolved (called the *annealing path*) through this search space is not fully controllable, but rather the total physical time taken is set by the user of the QA QPU.
- **Readout of the solution.** At the end of the annealing phase, the qubits are measured in the computational basis, and their configuration represents a candidate minimum of the final Hamiltonian. The individual spin values of the final configuration are read out and stored externally representing a candidate solution to the original problem.
- **Resampling.** Because quantum annealing is a heuristic, there is only ever a non-zero probability that the computation results in a ground state of the system. Therefore, the anneal-readout cycle is repeated many times per problem to acquire multiple candidate solutions. The number of times this is performed is determined by the user.

It is important to note that, as a heuristic optimization routine, the *time* used by the quantum annealing algorithm must be distinguished from the time used by the QPU. Total runtime is a combination of engineering-specific timing that cannot be altered, user-specified parameters, and the number of independent trials. To represent wall-clock time, $t_{\text{wall-clock}}$, the most general timing model can be regarded as follows:

$$t_{\text{wall-clock}} = t_{\text{prog}} + N_{\text{reads}} \cdot (t_a + t_{\text{readout}})\,. \tag{2.13}$$

Here, t_{prog} is the programming time required to set the initial values of the qubits and couplers, t_a is the user-specified annealing time[1] (physical time of evolution from $\mathcal{H}_i$ to $\mathcal{H}_f$), t_{readout} is the time to read out all the qubits' states at the end of the annealing cycle, and N_{reads} is the number of trials (i.e., the number of samples obtained from the QPU). The importance of this model is exemplified

[1]This is essentially the same as τ from Sec. 2.2. However, to distinguish the theoretical aspects of adiabatic evolution and the user-specified parameter for fixed annealing paths as implemented in D-Wave QPUs, we use t_a to refer to the annealing time in practice.

by the analysis of "runtime" for quantum annealing: clearly, the more samples are taken (N_{reads} is increased), the higher the probability that one of the samples is a ground state. However, the probability of the proportion of a single sample obtained at the end of the annealing is a ground state is controlled by t_a. For the research presented in this thesis, the distinction is made clearly in the appropriate contexts.

CHAPTER 3

Annealing control parameters and tuning

In the scope of this thesis, we are particularly interested in the applicability of quantum annealing implemented in quantum hardware. Therefore, while some combinatorial optimization problems may be simple to formulate conceptually, the process of attempting to solve these problems with quantum hardware is not straightforward. To understand this, we selected one representative class of NP-hard problems with a known simple QUBO formulation to study, the unweighted maximum independent set (MIS) problem. We start by solving these problems using standard techniques for QA, and compare the results to standard classical competition algorithms. Then, we attempt to improve the performance of the QPU by analyzing a subset of control parameters designed to mitigate some of the bottlenecks in QA. Thus, we use MIS to highlight the properties of the QPU in a targeted way, and deeply explore the role of parameter tuning in QA in general.

3.1 Combinatorial optimization on a quantum annealing processor

The advantage of choosing the MIS problem for an initial investigative analysis is multifold. First, the problem is APX-complete. Meaning that, since no (classical) PTAS exists, quantum algorithms may be a candidate for speedup on this problem class. Furthermore, the QUBO representation of this problem is constraintless. This significantly simplifies the tasks needed to solve these problems with QA in a number of ways. Since there is no way to directly constrain the search space in quantum annealing hardware, constraints are expressed by adding penalty terms to the objective function, as discussed in Sec. 2.4.1. Constraints have the detrimental effect of added complexity to the problem formulation, both in terms of increased number of terms in the problem (which in turn increases the connectivity of the problem), as well as the precision required to encode it, a known issue with quantum annealing hardware [70, 76, 74]. Constraintless problems also mean that there are no *infeasible* candidate solutions, which allow for direct examination of QA's ability to explore the solution space. Lastly, MIS is a particularly good starting point for the investigation of combinatorial optimization: every graph has a (sometimes trivial) MIS. Thus it is trivial to formulate MIS instances using random graphs, allowing tunable difficulty for the problems investigated. It allows us to generate test bed instances without relying on other data sources.

3.1.1 The maximum independent set problem

The formal definition of the unweighted MIS problem is as follows: given an unweighted undirected graph G with vertices V and edges E, find the largest subset of vertices $V' \subseteq V$ for which no two edges in V' have an edge in E. A straightforward QUBO formulation of this problem is given in [77], with the following objective function:

$$H = -B \sum_{i} x_i + A \sum_{ij \in E} x_i x_j. \tag{3.1}$$

Here, x_i are binary variables, and A and B are constants. In order to ensure that the minimum of H solves the MIS problem, it is required to set $B < A$ [77]. In our experiments, we used $B = 1$ and $A = 2$. We briefly demonstrate that this is sufficient: for each vertex v_i that is added to the candidate independent set V', we add -1 to the objective function value. Adding a node v_j that has an edge $e_{ij} \in E$ raises the objective function value by 1 and is therefore not favorable. Thus, the minimum of H represents the maximal set of unconnected nodes $v_i \in V'$, which is the definition of solving the MIS problem.

To create a suitable instances of the MIS problem, we used the Erdős-Rényi model to generate a family of random graphs, typically denoted as $G(n, p)$[1]. Here, a

[1]The Python package NetworkX has built-in functionality to generate such graphs given the appropriate parameters. More information about this package can be found here: `https://networkx.org/`.

graph G is generated with n vertices, with each pair of edges having a probability p of being connected. Since the purpose of this investigation is to understand the QPU's ability to solve a canonical NP-hard problem, we attempt to generate graphs with properties that probe the limits of quantum annealing performance. The goal is to identify the maximum point of difficult empirically in a self-consistent manner: by exploiting the edge probability p to tune the density of the random graphs, it is possible to control some aspects of the difficulty for the QA algorithm as a function of problem size n. The point p at which the QPU has the lowest probability of finding the putative optimum (largest independent set for these graphs) therefore corresponds to the maximum point of difficulty. For this initial experiment, we generated random graphs of $n = 40$ at different edge probabilities $p \in [0.05, 0.425]$ in steps of 0.025, with 50 random graphs at each point of p. These problems were then solved using a D-Wave 2000Q processor, with 100,000 samples (candidate solutions) collected for each problem. The results of this experiment are shown in Fig. 3.1. Qualitatively, the QPU behaves roughly as expected: for small p, random graphs are sparse, and therefore have easy-to-find independent sets. As p increases, the graphs become denser, and the MIS becomes harder to find. However, as p continues to increase, the graphs slowly approach being a fully-connected graph (clique), at which point the MIS sizes decrease substantially (for a clique, the MIS size is 1), which is again easy to find. Therefore the maximum point of hardness is expected to lie at some point in the middle. Quantitatively, the observed point of hardness for the QPU was at $p = 0.2$, which is also consistent with previous studies of MIS with random graphs [78]. Therefore, this edge probability was used for the remainder of the MIS experiments.

3.1.2 Minor-embedding and parameter settings

A crucial step in solving arbitrarily-structured graphs using a QPU is the process of minor-embedding. As explained in Sec. 2.5, each QPU has a fixed topology defining the connectivity between qubits. Therefore, in order to map arbitrary graphs to the connectivity of the QPU, each node in the source graph must be represented as a *chain* of connected physical qubits in the QPU. While in principle this incurs only a polynomial overhead in the number of variables (qubits) used to describe a problem for quantum annealing, it has been previously observed that D-Wave QPUs have a lower probability of finding global optima for problems which require embedding [70]. Thus, most interesting and difficult problems must be

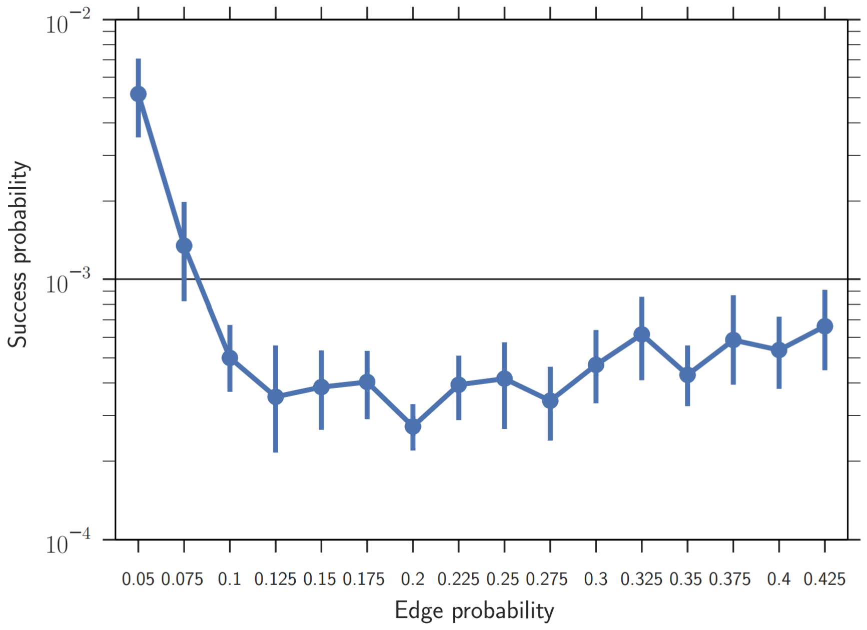


Figure 3.1: Distribution of success probabilities of the tested D-Wave 2000Q QPU as a function of edge probability p for randomly generated graphs (50 graphs at each point of p). The maximum point of difficulty (e.g., lowest mean success probability) was observed at $p = 0.2$. Error bars are bootstrapped 95% confidence intervals.

minor-embedded to match the QPUs topology[1]. In order to understand this effect more concretely, we performed another initial experiment: 50 random graphs were generated at $n = 50$ and $p = 0.2$, and the `minorminer` Python package was used to generate 25 different embeddings for each problem. Each of these embeddings was used and submitted to the QPU, and the respective sizes of the independent sets for each problem (and embedding) were recorded. To analyze the effect of embedding on the performance of the QPU, we performed linear regression to check the correlation between embedding size (total number of qubits used in the embeddings) and the quality of the results (distance of each independent set from the putative optimum). The slopes of these linear regressions are the major point in this analysis, as it shows the marginal degradation of the results as a function of additional qubits used in the embedding. The distribution of the slopes for each problem is shown in Fig. 3.2. Since the vast majority of the points shown are above zero, we can conclude that larger embeddings are on average worse for such combinatorial optimization problems, which is consistent with results from literature. Thus, we allocate some pre-processing time to generate a number of candidate embeddings, and select the one which uses the smallest number of qubits.

The last step in solving embedded problems using a QPU is deciding on the optimal *chain strength* which will be used to encode each logical qubit. While there are some rough rules of thumb and bounds [67, 79], the chain strength is a tunable free parameter of the embedded problem. In practice, chains act as additional constraints on the problem being solved, with the chain strength being the magnitude of the constraint. It is therefore necessary to set a chain strength such that all qubits in the chain act as a single logical qubit, so that it is never energetically favorable to have oppositely valued qubits within a single chain in order to reduce the overall value of the objective function (such a result is referred to as a *broken chain*). Unfortunately, this value is not always known *a priori*, and may change from one problem to the next. Furthermore, using a chain strength which has too large a value has the effect of compressing the logical problem being solved, which leads to a degradation in performance of the QPU. Thus, an optimal trade-off must made such that the smallest possible chain strength which correctly encodes the problem is used. In the MIS case, we performed a sweep over many

[1]D-Wave Systems provides an open-source Python package which can be used to find minor-embeddings as per the algorithm presented in [68]. The Python package can be found here: `https://github.com/dwavesystems/minorminer`.

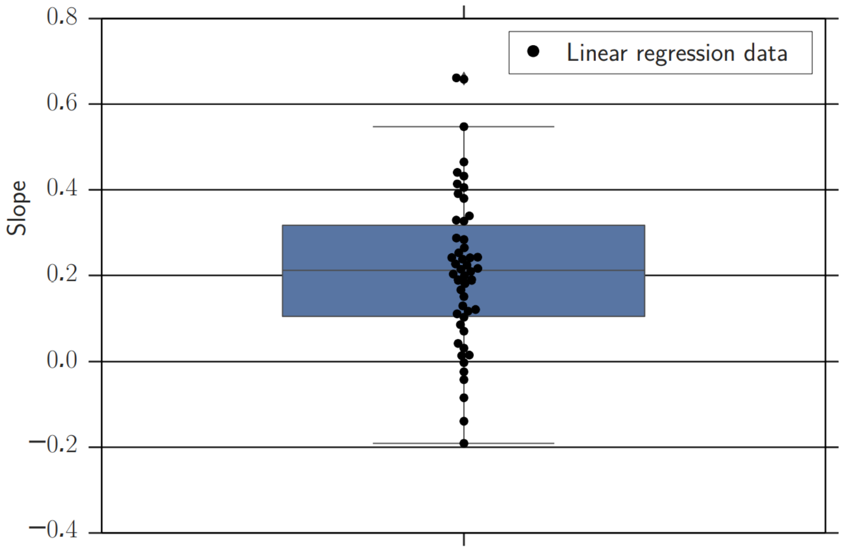


Figure 3.2: Boxplot showing the distribution of linear regression slopes of 50 MIS problems at $n = 50$ and $p = 0.2$ (25 embeddings used for each point), all solved using a D-Wave 2000Q QPU. The majority of the points lie above zero, indicating the larger embeddings cause the QPU to find worse optima.

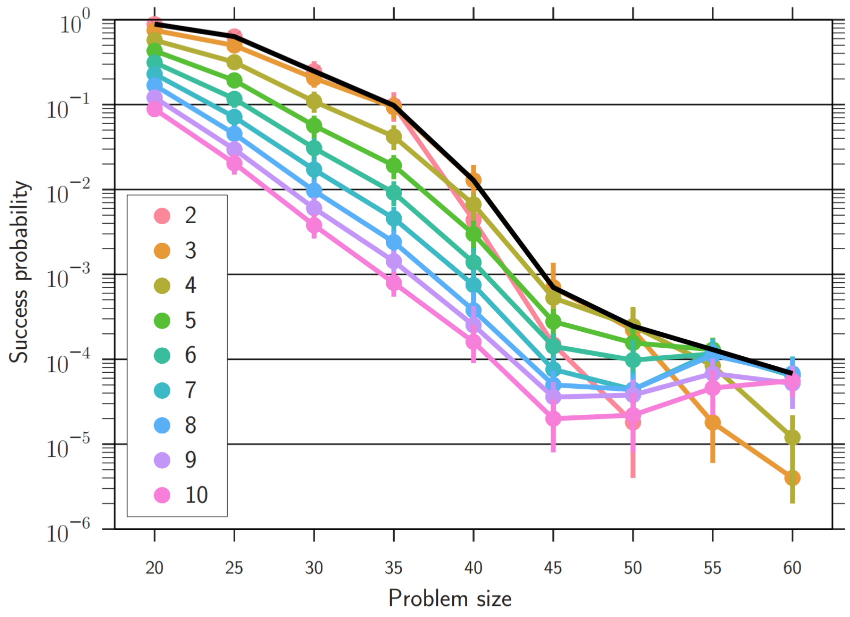


Figure 3.3: The probability of the D-Wave 2000Q QPU finding the putative optimum of each problem for different chain strengths. The black line shown is the envelope of optimal chain strengths at each problem size n. As n increased, larger magnitude chain strengths were optimal.

such chain strengths, solving each problem at different chain strengths to observe the effects of this parameter. In these experiments, we generated 50 random graphs at $n \in [20, 60]$ in steps of 5 (all at $p = 0.2$). We solved each of the 50 problems at every size using a variety of different chain strengths $\in [2, 3, \ldots, 10]$, and measured the QPU's performance for each. The results of this experiment are shown in Figure 3.3. As expected, larger problems required stronger chain strengths in order to correctly encode the problem. This is due to the average node connectivity (or degree) in the logical graph as n increases, which in turn requires additional qubits to be coupled in a single chain. Interestingly, Fig. 3.3 also demonstrates the consequences of exceedingly large chain strengths for the smaller problem sizes: the stronger chain strengths (for example, 10) caused up to two orders of magnitude degradation in performance at $n = 40$, further demonstrating the importance of correctly tuning this parameter.

While these results are only useful for this specific test bed (MIS problems with

$p = 0.2$), we can use the black line in Fig. 3.3 as a guide for the remaining experiments involving the MIS problems. The QPU is therefore considered to be optimally tuned in regards to chain strength for each of the problem sizes which will be tested.

3.1.3 Benchmarking quantum annealers against classical algorithms

For the main MIS experiments, we generated a new set of Erdős-Rényi graphs at edge probability $p = 0.2$ and $n \in [20, 60]$ in steps of 5 (50 instances at each n). Each problem was embedded using the methodology in the previous section, and solved using a D-Wave 2000Q QPU. A total of 100,000 samples (candidate solutions) were collected for each problem instance. A standard noise-mitigation strategy, *spin-reversal transforms*, was used. These spin-reversals involve using random vectors of spin states $\{-1, 1\}^N$ (N is the length of the vector) to reverse the signs of terms in the problem being submitted to the QPU, but leaves the underlying structure of the problem intact. It has been shown that this technique averages over the systematic noise present in the QPU, overall enhancing the results [75]. In these experiments, we used a total of 100 spin-reversals per problem (1 spin reversal per 100 samples), as it has been shown that more reversals has diminishing returns [75]. The annealing time for each sample was set to the default minimum, $t_a = 20\ \mu s$. Since the objective of this analysis is to understand the raw performance of the QPU given the embedded MIS problem, we define the amount of time taken by the QPU to solve each problem as the raw computation time (t_{compute}) needed to obtain the putative optimum:

$$t_{\text{compute}} = N_{\text{reads}} \cdot t_a, \tag{3.2}$$

where N_{reads} is the total number of samples. Thus, the maximum possible t_{compute} per problem was 0.2 s. All other parameters of the QPU whose values are not mentioned here were set to their default values.

Two classical competition algorithms were chosen to compare against the QPU results. The first is a well-known heuristic optimization algorithm, simulated thermal annealing (SA). D-Wave Systems has an open-source Python implementation of SA which was used for these experiments[1]. The SA algorithm was originally

[1]The Python package can be found here: `https://github.com/dwavesystems/dwave-neal`.

inspired by the annealing process of metals [33], and has been used extensively for quantum annealing benchmarking studies in the past [72, 73, 76]. The algorithm starts in a random binary string configuration (corresponding to infinite temperature), and is manipulated through a "cooling" schedule until a candidate solution is reached in a user-specified number of steps, N_{sweeps}. In these experiments, $N_{\text{sweeps}} \in \{10, 100, 1000\}$ were tested, and it was found that $N_{\text{sweeps}} = 10$ was sufficient for all problem sizes. At every step of the algorithm, a possible bit-flip is proposed, and accepted with a probability P, defined as:

$$P = \max\left\{1, \exp\left(-\beta \Delta E\right)\right\}. \tag{3.3}$$

Here, $\beta = 1/T$ is the inverse temperature parameter and ΔE is the energy (objective value) difference between the two system states before and after the flip. The "cooling" of the system is reflected in the way in which the β parameter is slowly increased, effectively lowering the probability of accepting energetically unfavorable bit-flips[1]; choosing such a scheme is known as a schedule. In our experiments, we used the schedule $\beta \in [0.01, 3]$, interpolated geometrically with N_{sweeps} points.

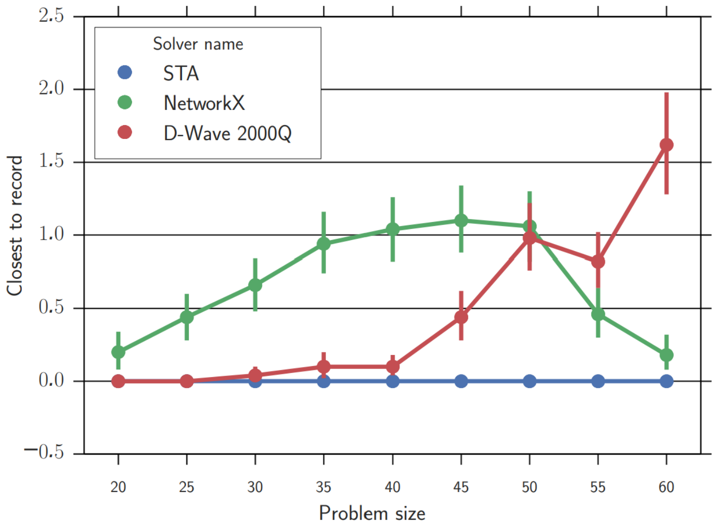
The other algorithm is a polynomial-time approximation to the MIS problem provided by the Python package NetworkX, based on the algorithm presented in [80]. This algorithm provides a solution with quality bounded by $O\left(|V|/\log\left(|V^2|\right)\right)$ in the worst case. The purpose of this approximation is to examine the average hardness of the test bed instances in the "native" domain of MIS instances (meaning, without going through the translation to QUBO).

The goal of the comparison was to identify the ability of the QA algorithm (as implemented on a D-Wave QPU) to solve a canonical NP-hard optimization problem relative to classical optimization algorithms. In general, heuristics are always in tension with respect to three different criteria: solution quality, runtime, and problem size. Ideally, a powerful heuristic is able to provide good solutions to an optimization problem quickly, and for a large range of problem sizes. In practice, a compromise between these must be made. For example, longer runtimes may be permissible in favor of higher-quality solutions. Thus, to make the trade-off

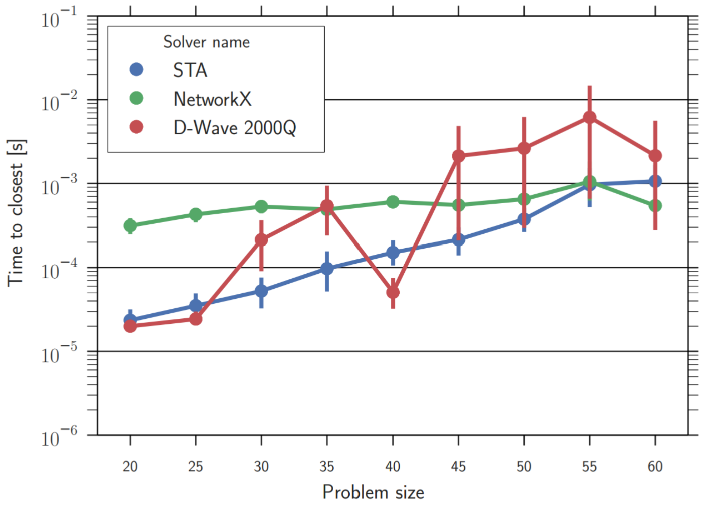
[1]Moves that improves the candidate solution are always accepted. As the algorithm progresses, however, these are exponentially unlikely and thus unfavorable moves may be beneficial in finding new subspaces to explore.

between the different solvers more apparent in our experiments, we consider the quality of solution and runtime of the algorithms separately, in Fig. 3.4 (a) and (b), respectively. Because verifying that the putative optima are global minima is NP-hard in the worst case, we evaluate each of the solvers based on their performance in a self-consistent method. After running all solvers on all instances, we consider the largest independent set found among any of the solvers as the putative (or "record") optimum for each instance. Thus, we can consider how well each solver performed by considering how close each solver was (on average) to the putative optima, shown in Fig. 3.4 (a). For all problem sizes tested, SA was always able to find the putative optimum. The D-Wave 2000Q QPU was only able to consistently find the same solutions for the smallest problem sizes tested $(20-25)$. However, in the mid-range $(30-40)$, the QPU was not able to find global optima for all problem. At the largest sizes $(45-60)$, the QPU's performance degrades significantly, and the independent sets found are consistently smaller than both SA and the approximation algorithm. These results can be explained by examining the polynomial approximation's performance: at small graph sizes, the problem is sparse, and so the MIS is easier to find for SA (single variable flips are more common) and the QPU (smaller embeddings are sufficient). As the problems reach intermediate sizes, the polynomial approximation degrades, as is expected for such an algorithm. However, at the largest problem sizes, the density of the graph grows, which makes the MIS problems easier to approximate. From the QPU perspective, the density also increases the size of embeddings (and correspondingly, the chain lengths and strengths) needed to execute the QA algorithm. With increasing embedding sizes and chain strengths, it has been observed that the dynamics of the qubits "freeze" earlier in the annealing schedule, which results in settling in local rather than global minima since the chains stop reacting to the evolving problem Hamiltonian [81]. Thus, even though the MIS problems are "easier" classically, this does not translate to improved QPU performance due to the overhead required to solve the problems in hardware.

In Fig. 3.4 (b), the time required by each solver to find its record solution is compared. As expected, all the algorithms tested were observed to have increasing runtime with increasing problem size. However, at the two smallest problem sizes $(20-25)$ the QPU had the fastest time to record. This is likely due to the fact that these instances were classically easy, but also in a regime where the hardware and embedding effects were not too prominent for the QPU. As the problem sizes increased, these effects became more prominent, resulting in unfavorable time



(a)



(b)

Figure 3.4: **(a)** The performance of each algorithm represented as difference between best solution found and putative optima, relative to increasing (logical) problem size. **(b)** Mean runtimes for each algorithm tested, in seconds. In both, error bars are bootstrapped 95% confidence intervals.

scaling for the QPU relative to the classical algorithms. It is important to note that these results are not reflective of the scaling of the QA algorithm in theory (e.g., the optimal annealing time t_a required to solve these instances, as discussed in Sec. 1.1), but simply the scaling of the routine as set by the parameters discussed above.

By selecting MIS as a representative NP-hard problem class, several conclusions can be drawn from the analysis that apply to combinatorial optimization with QA in general. This was done by constructing QUBO instances that were entirely constraintless, allowing the examination of the QPU's ability to perform heuristic optimization directly. The task of minor-embedding and setting the individual parameters of the QPU by hand allowed thorough testing of the performance of QA with respect to the basic necessary steps required to pose problems to the quantum annealer. Based on the results presented, it is evident that the difficulties encountered by the QPU in finding global optima are different from those of classical algorithms. The process of transforming a canonical NP-hard problem so that it can be solved with QA necessarily involves steps that result in different manifestations of computational bottlenecks (embedding, precision, noise, etc.). When evaluating QA, it is important to identify which of these potential bottlenecks are impacting performance in order to correctly understand the implication of the results. This is evidenced by the observation that the QPU was the best-performing algorithm in both runtime and solution quality for the smallest MIS instances ($20-25$ nodes), but at the largest problem sizes was the worst performing algorithm. Furthermore, it is apparent that wall-clock time plays a unique role in QA. The basic annealing time (t_a) determines some aspects of the underlying physics of quantum annealing and therefore affects the ground state probability. However, much like some classical metaheuristics, in practice attempting to solve problems (even simple constructions such as MIS problems) requires many candidate solutions to be obtained, regardless of the relative simplicity or complexity of the optimization problem. Therefore, the onus is on the user of QA to find the optimal trade-offs between tuning QPU parameters and executing the algorithm in order to gain some practical use.

3.2 Advanced annealing controls

As observed in the results from the previous section, implementing QA in programmable hardware has obvious limitations when applied to combinatorial optimization problems. The rigid topology of qubits in the layout of a QPU necessitates the process of minor-embedding in order to solve arbitrarily-connected graph structures, which result in a multitude of consequences in practice. To address this, there are a set of control features which have been developed and implemented in successive generations of D-Wave QPUs processors which have been specifically designed to mitigate some of these issues. While full annealing path programmability is still impossible, it is possible to manipulate *some* elements which contribute to the annealing path using these controls. Here we review a subset of such controls, along with existing known methods on tuning them (and where applicable, their known physical effects on performance). The impact of annealing offsets, in particular, on the QPU performance is tested explicitly with small toy problems. Then, new algorithms using these controls are developed and tested with D-Wave QPUs. We show how to tune and use these controls in practice to solve maximum independent set problems, expanding upon the previous work and improving the overall QPU performance.

3.2.1 Tunable annealing control parameters

We briefly review the set of features which were tunable in the D-Wave 2000Q generation of QPUs used in the MIS experiments. Where applicable, the relevant physical effects are explained and related to practical applications of QA.

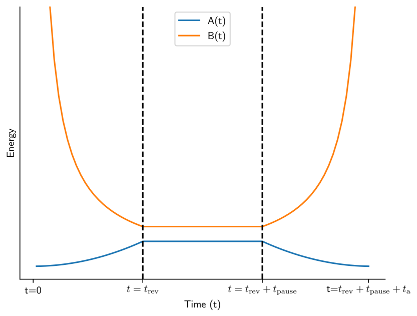
Reverse annealing. As opposed to standard (forward) annealing, where all qubits are initialized in a simple ground system of equal superposition, reverse annealing (RA) implemented in D-Wave QPUs performs a fundamentally different task. Each qubit in the system is initialized in a computation basis state, the ensemble of which represents a classical state, in effect seeding the heuristic optimization with a starting point. Then, the annealing process proceeds in reverse, and the magnitude of the initial Hamiltonian (also known as the *transverse field*) is slowly raised. The transverse field, as in forward annealing, still enables tunneling between qubit states. The strength of the transverse field (or equivalently, the point in the annealing process where the reverse annealing is stopped) is a parameter to this protocol, and dictates the degree of "locality" in which the QPU is able to explore [82]. A schematic diagram of this annealing protocol is shown in Fig. 3.5 (a). Because this feature initializes at the end of the forward annealing schedule, the reverse evolution of the Hamiltonian may result in following different annealing paths than in strictly forward annealing, which may aid in finding optima to optimization problems [83].

This RA protocol was the first implemented in D-Wave QPUs that expanded upon standard forward annealing, and as such has been investigated thoroughly since its construction. Reverse annealing was first used to conceptualize a mutation operator in a quantum-assisted genetic algorithm [84], which was then subsequently implemented and demonstrated to be effective [38]. It has also been demonstrated that this protocol can be used as a Markov chain operator to sample from mixed quantum states in frustrated lattices [85]. In the context of optimization problems, reverse annealing has been used to benchmark portfolio optimization in financial models [86, 87], scheduling optimization problems [88], and for the inner optimization routine in non-binary matrix factorization [89, 61].

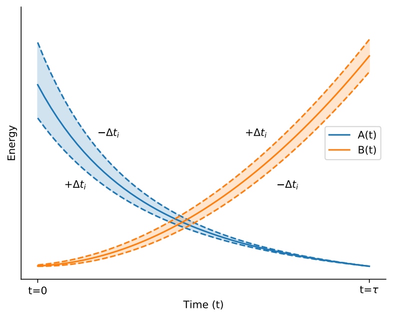
Anneal pause. It has been well-established that thermal noise in quantum annealers exhibit certain signatures [90] which can be measured experimentally [46, 91]. Specifically, it has been shown that under open quantum system conditions, qubit states can be driven to a higher eigenstate due to coupling to the environment (known as excitation) before returning to a lower energy state (known as relaxation). This effect is typically observed when the timescale of dynamics during the quantum annealing evolution is much longer than the timescale of thermal fluctuation in the system, known as the *relaxation time*. To investigate this effect, the novel control parameter of anneal pause was introduced, which allows the modification of the annealing schedule by adding a constant amount of physical time (t_{pause}) to each annealing cycle where the evolution is stopped. The schematic of anneal pause in the annealing schedule is also shown in Fig. 3.5 (a). Experimental investigations have shown that pausing can increase the probability of finding optima by orders of magnitude (under the correct conditions) in ways that strictly increasing the annealing time can not [92].

Annealing offsets. One further attempt at implementing qubit-specific control features is the annealing offset parameter. This feature is theoretically motivated by the qubit freeze-out dynamics, where the qubits' dynamics stop responding to the evolving energy landscape [25]. With annealing offsets, the shape (or path) of the anneal scheduling is not modified, but only shifted in time by a constant amount, Δs_i (in normalized time units) for qubit q_i. Because this time shift can be applied to each qubit in the QPU independently, this has the effect of shifting the freeze-out point of individual qubits relative to each other. This is particularly useful in embedded problems (such as MIS), since chains of different lengths and strengths may freeze at different points in the schedule. A diagram exemplifying annealing offsets on a two qubit example is shown in Fig. 3.5 (b).

As with other performance-enhancing protocols, finding optimal such offsets depend on the specific Hamiltonian being solved, which therefore poses a difficult problem in general. There has been some experimental demonstration of effective offset tuning using a variety of methods. Analysis based on perturbation of the driver Hamiltonian was shown to increase probability of finding optima through iterative evaluation of an objective function [26]. In applications, a grid-search technique was used to boost performance of an integer factoring routing using a D-Wave QPU [93, 94]. Depending on the instances being studied, it was shown that the probability of finding the ground state could increase by up to multiple orders of magnitude.


A(t)
B(t)
Energy
t=0
$t = t_{\text{rev}}$
$t = t_{\text{rev}} + t_{\text{pause}}$
$t = t_{\text{rev}} + t_{\text{pause}} + t_{\text{a}}$
Time (t)

(a)



(b)

Figure 3.5: (a) Diagram of a simple reverse annealing protocol (with duration t_{rev} in reverse and t_a forward) with a pause (of duration t_{pause}) in the middle. **(b)** Diagram of possible annealing schedules using an anneal offset of maximum magnitude $|\Delta t_i|$.

3.3 Heuristic tuning of QPU controls

A major drawback in the optimization of QPU control parameters is the complex nature of the optimization task in itself. Whether exhaustive or iterative schemes are used (such as those presented in previous sections), significant resources are often devoted to finding optimal schedule shifts to improve QA performance. In practice, this leads to significant bottlenecks in the time devoted to tuning the QA parameters. Therefore, we now introduce methods to attempt to improve QA performance using some of these advanced controls without causing additional bottlenecks in computation. We employ the use of a black-box evolutionary algorithm in order to tune the anneal offsets in a D-Wave 200Q QPU. We develop these methods particularly to solve embedded instances of the MIS problems, again as a representative class of canonically hard optimization problems. This is then used to draw comparative conclusions regarding the resources needed to tune QA in a more general context.

3.3.1 Covariance Matrix Adaptation Evolution Strategy for offset tuning

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [95] is a stochastic optimization algorithm for continuous black-box optimization. To tune the annealing offsets, we adopt the so-called $(1 + 1)$-CMA-ES variant [96], which generates only one candidate search point in each iteration. The $(1 + 1)$-CMA-ES algorithm exhibits both fast convergence and global search ability. The choice of the optimization algorithm is made based on the following considerations: firstly, QPU time is considered an expensive resource, and we wish to spend as little QPU time for the tuning as possible. Secondly, the QPU is an analog device, meaning that problems (one candidate set of annealing offset parameters in this case) can only be tested sequentially. Therefore, search strategies that generate multiple candidate points are not advantageous as it is not possible to parallelize those points in our case. For the experiments reported in this paper, we run the $(1 + 1)$-CMA-ES with its default parameter settings, since it is well known that such a setting shows quite robust behaviors across many benchmark functions [97]. Pseudocode outlining the algorithm used to tune the offsets is shown in Alg. 1 and the appropriate terms (along with the values used in our experiments, when applicable) are defined in Tab. 3.1. Essentially, the proposed algorithm optimizes the annealing offsets using the so-called mutation operation (Line 10 and 11), where the current annealing offset candidate $\mathbf{x}$ is perturbed by a Gaussian random vector $\sigma\mathbf{Az}$ (which is transformed and rescaled from the standard normal vector $\mathbf{z}$, Line 10). The resulting mutation $\mathbf{x}'$ is evaluated with the QPU (Line 12) and is passed onto the next iteration if its objective value $f(\mathbf{x}')$ is better than $f(\mathbf{x})$ (Line 14 and 15). In addition, two procedures, UPDATE-STEP-SIZE and UPDATE-CHOLESKY are adopted to control the step-size σ and the matrix $\mathbf{A}$. The details of those two procedures are presented in [96].

Recall that in the case of MIS problems, to solve them directly with the QPU, minor-embedding is necessary for each graph. Therefore, the objective of the (1+1)-CMA-ES is to tune the annealing offsets of each qubit chain collectively, thus attempting to mitigate the freeze-out effect of the quantum system by mitigating each chain independently. To calculate the offsets for each chain, we must then consider the *collective* minimum and maximum offset ranges that can be used for the chain to act as a single logical qubit. We therefore find the highest minimum and lowest maximum shared value between all qubits in every chain, and use those as the boundary for each logical qubit (chain). Every qubit within every chain is then advanced/delayed by the same amount, resulting in a singular collective shift of the entire qubit chain.

In our implementation of (1+1)-CMA-ES, we considered two starting points for the evolution strategy: *uniform* and *null*. The random uniform initialization was selected due to its known performance advantages by allowing a fair exploration of the search space within the offset bounds of each chain. The initial points are selected uniformly for each chain c_i from the range of feasible values $[l_i, u_i]$ (common lowest/upper bound for each chain). The fair exploration is particularly important in this model due to the non-convex nature of the problem, where we have a high-dimensional search space and each point is possible optimum. The alternative approach (null), is set by selecting all zeros for each initial offset value. This is equivalent to starting with a standard quantum annealing protocol as a seed to the CMA-ES. We consider this a “not bad” starting point (essentially a local optimum in the search space) and attempt to improve upon it. This is also the default value for standard forward annealing.

At the core of the (1+1)-CMA-ES tuning routine is a necessary step of evaluating a fitness function, testing the current quality (fitness) at every iteration of the procedure. The fitness (objective) function function used in this case was the *mean energy* of the samples returned from the QPU at every iteration, with a sampling rate of 100 samples per iteration. Thus, the goal was to drive the CMA-ES search towards lower mean energies of the problem Hamiltonians in an attempt to find larger MIS solutions. In theory, other objective functions could be used; in preliminary experiments 25% percentile energies and minimum energies or candidate solutions were tested, but mean energy proved to be the most stable metric, and was the best performing fitness function. In Fig. 3.6 we present one example of tuning the offsets for a 40 node graph.

Algorithm 1 Pseudocode for the routine used to tune the annealing offsets parameters using $(1+1)$-CMA-ES.

```
1:  procedure TUNE-OFFSET($\mathbf{l}, \mathbf{u}, B, \texttt{StepCost}$)
2:      Initialize: $\sigma \leftarrow \max\{\mathbf{u}-\mathbf{l}\}/4,\ \mathbf{C}=\mathbf{I},\ c \leftarrow 0$
3:      if $\texttt{InitialOffsets} = 0$ then
4:          $\mathbf{x} \leftarrow \mathbf{0}$                         ▷ offset: zero initialization
5:      else
6:          $\mathbf{x} \leftarrow \mathcal{U}(l_i, u_i)$                  ▷ offset: uniform initialization
7:      end if
8:      $f(\mathbf{x}) \leftarrow$ CALL-QPU($\mathbf{x}, \texttt{StepCost}$)
9:      $\mathbf{A}\mathbf{A}^\top \leftarrow \mathbf{C}$                        ▷ Cholesky decomposition
10:     while $c < B$ do
11:         $\mathbf{z} \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$                 ▷ standard normal distribution
12:         $\mathbf{x}' \leftarrow \mathbf{x} + \sigma\mathbf{A}\mathbf{z}$
13:         $f(\mathbf{x}') \leftarrow$ CALL-QPU($\mathbf{x}', \texttt{StepCost}$)
14:         $\sigma \leftarrow$ UPDATE-STEP-SIZE($\sigma$)
15:         if $f(\mathbf{x}') < f(\mathbf{x})$ then
16:             $\mathbf{x} \leftarrow \mathbf{x}'$
17:             $\mathbf{A} \leftarrow$ UPDATE-CHOLESKY($\mathbf{A}, \mathbf{z}$)
18:         end if
19:         $c \leftarrow c + \texttt{StepCost}$
20:     end while
21:     return $\mathbf{x}$
22: end procedure
```

B	Total budget (amount of resources) for tuning QPU annealing offsets (in units of total number of samples drawn from the QPU; In total, 10,000 samples are used in our experiment per instance.).
`InitialOffsets`	The initial value for each offset of qubits in the problem; we test either all set to 0 or uniformly between their min/max range.
`StepCost`	The cost of each step of the fitness evaluation of the offsets (in number of samples from the QPU; we used 100 samples per call).
$\mathbf{x}$	The current annealing offsets.
$f(\mathbf{x})$	Fitness value of the current offsets, measured in units of mean energy of the samples returned by the QPU.
c	Counter for the budget (measured in number of samples from the QPU).
CALL-QPU	The objective function that calls the QPU, takes $\mathbf{x}$ and `StepCost` as arguments.
σ	The step-size that scales the mutation of offsets.
$\mathbf{C}$	The matrix of covariances between the annealing offset values
$\mathbf{A}$	The Cholesky decomposition of $\mathbf{C}$
UPDATE-STEP-SIZE	The procedure to control the step-size σ. Please see [96] for the detail.
UPDATE-CHOLESKY	The procedure to adapt the Cholesky decomposition $\mathbf{A}$ of the covariance matrix $\mathbf{C}$. Please see [96] for details.
$\mathcal{U}(a, b)$	Uniform random distribution in $[a, b]$.
$\mathcal{N}(\mathbf{0}, \mathbf{I})$	Standard multivariate normal distribution.

Table 3.1: Table explaining the variables and procedure used in Alg. 1.

3.3.2 Solving optimization problems with parameter tuning

The problem instances used for these experiments were from the same problem class as those in the last section, MIS problems of random graphs (20-60 nodes, edge probability $p = 0.2$). To test the efficacy of the (1+1)-CMA-ES procedure, we allocate an initial budget of resources B (number of samples queried from the QPU) with which we tune the offsets. Then, we sample another B samples with the terminal annealing offset values $\mathbf{x}$, and record the best solution found in this *second* part of the procedure. The goal is to allocate a sufficient amount of resources for calibration, and then solve the MIS problems with the remaining QPU time. In these experiments, we used $B = 10,000$ samples, for a total of 20,000 QPU samples per instance. For each problem size tested, 50 problems were generated. A D-Wave 2000Q was used to sample each MIS problem with three sets of conditions: (i) *uniform* initial offsets tuned with (1+1)-CMA-ES, (ii) *null* initial offsets tuned with (1+1)-CMA-ES, (iii) no offsets used (default). The no offsets setting is used here as a baseline for comparison. For the minor-embedding, as in the previous section, several embeddings were mined, and the smallest embedding was used to solve the problems. For setting the chain strengths, Fig. 3.3 was used as a reference guide.

We present the results from the QPU in Fig. 3.7. In Fig. 3.7 (a), we show the probability of finding the minimum. However, to fairly compare the effects of tuning the offsets, we show the mean of these probabilities *only* for the instances where the configuration was able to find the putative optimum[1]. As expected, the probability of the ground states decrease with increasing problem sizes. Furthermore, it is evident that for the instances where the QPU found the putative optimum, both offset tuning strategies provided a higher probability of ground states.

[1]In the cases where the QPU was not able to find the optimum, the null probability would heavily skew the mean, especially at the larger problem sizes.

In Fig. 3.7 (b), we show the number of instances where each tuning setting could not find the putative optimum. As expected, the no offsets strategy was the worst-performing setting at the largest problem sizes, with over half of the instances remaining unsolved. At the two largest sizes ($n = 55, 60$), both the null and uniform initial offsets solved more instances than no offsets. This means that for some instances, the largest independent sets found by the tuned offsets were *larger* than those found by the default settings of the QPU. At the smaller instance sizes, surprisingly, there were a few instances that remained unsolved after tuning for each problem size. On average, the null initial offsets outperformed the uniform setting. This can be explained by the fact that the null configuration represents a "not bad" starting point; the uniform initial offsets, by contrast, have no such guarantee. It is therefore possible that a random initial offset value was "bad", and the amount of samples budgeted to the tuning algorithm was insufficient to overcome this.

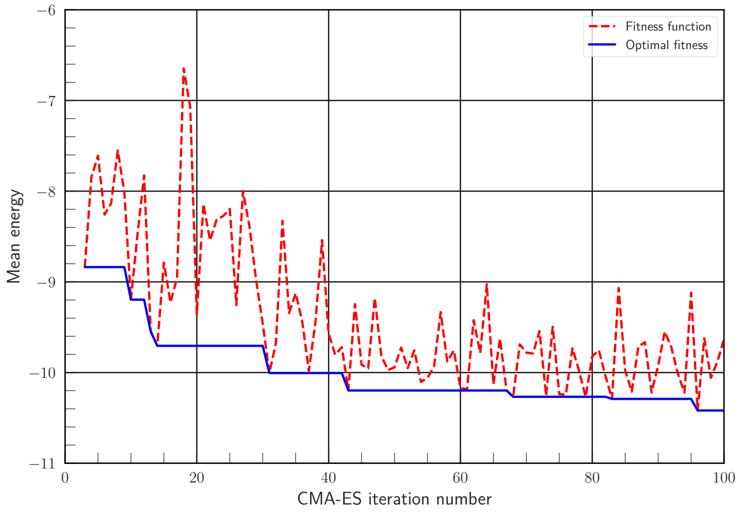
In order to quantify the degree of improvement due to the tuning, we take the ratio between the probabilities of finding the ground states (of each of the two strategies) relative to the baseline of no offsets. This is shown in Fig. 3.8. The improvement obtained by using the two configurations is qualitatively similar, and peaks at problem size 45 for both: 12.4 improvement for null initial offsets, and 8.2 times improvement for the uniform initial offsets. The improvement gained by tuning the annealing offsets steadily increases with problem size, until reaching its peak, after which the gains mostly disappear. This behavior indicates that at small problem sizes there is little to be gained from tuning, but at larger problem sizes the annealing offsets can have a significant impact on performance. The decay observed in success probability in problem sizes larger than 45 implies that insufficient resources were allocated to the tuning procedure, and more than 10,000 samples are needed to tune the offsets. Given the results from the previous section (where the QPU was able to outperform even SA for small problems), it is evident that the need for tuning the anneal offsets is particularly pronounced at the larger problem size.

In practice, exhaustive tuning of hyperparameters is prohibitive when using metaheuristics. Typically rules-of-thumb are used, or only basic optimization is performed. Furthermore, it is important to consider the amount of resources necessary to perform the tuning, relative to the cost of running the metaheuristic itself. Therefore, we devise a figure of merit in order to adequately compare the CMA-ES tuning method explored here with other techniques used in the past. The figure of merit (FoM) is defined as the number of samples needed by the QPU divided by the improvement in probability of ground states. Essentially, this figure represents a sort of *resource efficiency*, where the goal is to minimize this number, so that a minimum number of samples are needed to maximally improve the probabilities. We present a comparison with some previous literature (and calculate the FoM for those techniques) in Table 3.2. For completeness, we calculate our FoM for both initial offsets configurations for CMA-ES. By comparing the FoMs for each of the tuning methods, it is evident that improving the probability of ground states is not sufficient to judge the applicability of each method. For example, the grid search technique produced an improvement of 10^3 in probability of ground state, but required millions of samples in order to find. Thus, while theoretically demonstrable as proof that annealing offsets can be used to improve performance, it would be unrealistic to use such techniques in application to problems larger than toy instances. Under these conditions, we find that the CMA-ES method developed here is the most efficient technique (with FoMs of 806 and 1219 for the null and uniform settings, respectively). Since both cases used the same number of samples to tune (10^4), the only difference between them was that the null starting offsets resulted in larger improvements (factor of 12.4 as opposed to 8.2).

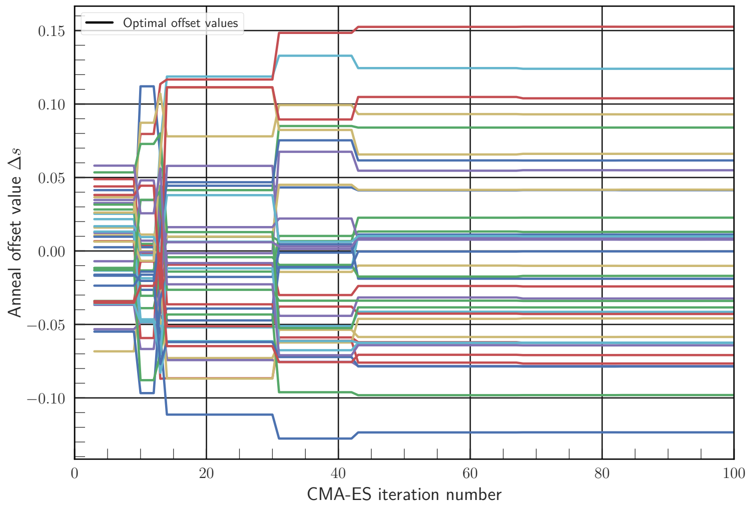
Method	**Samples**	**Improvement**	**Figure of Merit**
D-Wave (perturb.) [26]	$3.15 \cdot 10^5$	1.37	$2.3 \cdot 10^4$
D-Wave (grid) [93]	$2.5 \cdot 10^6$	1000	2500
CMA-ES (uniform)	10^4	8.2	1219
CMA-ES (null)	10^4	12.4	**806**

Table 3.2: Table comparing the different methods used to exploit the annealing offsets. Stated are the the number of samples used to tune the annealing offsets, the success probability improvement ratio, and the figure of merit, calculated by dividing the first two columns for each method. The lowest figure of merit (and hence most resource-efficient method) was the CMA-ES routine with null offsets, in bold in the table.

Considering the results presented in Fig. 3.7, we conclude that annealing offsets may provide some benefit to solving combinatorial optimization problems using direct embedding approaches in quantum annealing hardware. In particular, using an evolutionary strategy such as (1+1)-CMA-ES, which is well-suited for continuous parameter black-box optimization, we were able to improve the performance of the D-Wave 2000Q QPU tested for these MIS experiments by over an order of magnitude in probability of obtaining ground states. Furthermore, we demonstrated that by allocating a pre-defined budget of samples, the tuning procedure can (on average) allow the QPU to access better solutions than without tuning, and more efficiently than with other methods used for annealing offsets in the past. The figure of merit used to calculate resource efficiency was the lowest for the null initial offsets, which implies that the default configuration is both a local optimum for the annealing offsets search space, and also a good starting point for tuning procedures in general. However, the methods developed in this section are not guaranteed to improve performance across all problem sizes, and seem best suited for combinatorial optimization problems under the following two conditions: (i) problems that can be expressed via simple QUBOs (like MIS), and (ii) problems which require long chains. In practice, real-world combinatorial optimization problems are not as simple to express as the MIS case (which we show in detail in the next chapter), and so other mitigation techniques must be used to improve QPU performance.

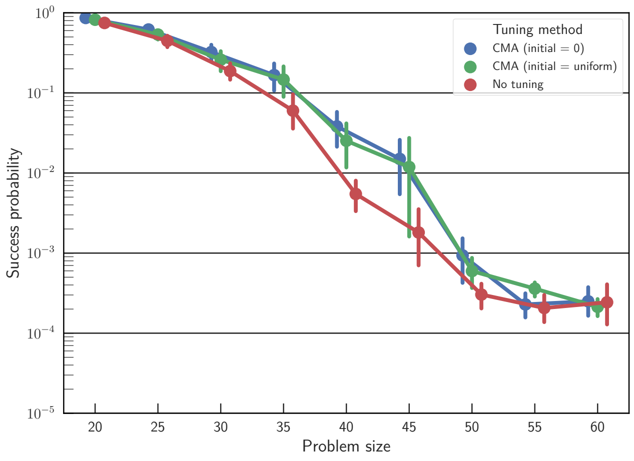


(a)

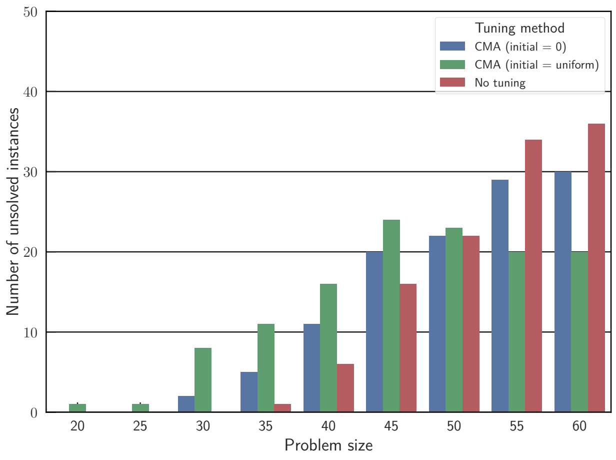


(b)

Figure 3.6: (a) A demonstration of the fitness function evolution for one 40 node MIS problem (100 samples taken per iteration). The red line is the fitness function value at every iteration of the CMA-ES algorithm, whereas the blue line is the cumulative minimum and represents the (presumably) best-known configuration. **(b)** The evolution of the offsets for each chain in the same 40 node MIS problem, updated every time a new minimum is found in the objective function.



(a)



(b)

Figure 3.7: **(a)** Probability of the QPU returning the ground state using B samples for each tuning setting, after the tuning procedure was completed ($2B$ samples used for the no offsets case). Means are calculated only for instances where the QPU was able to find the putative optima using that setting. Error bars are bootstrapped 95% confidence intervals. **(b)** The number of instances that remained unsolved (i.e., best solution found by the QPU was worse than the putative optimum).

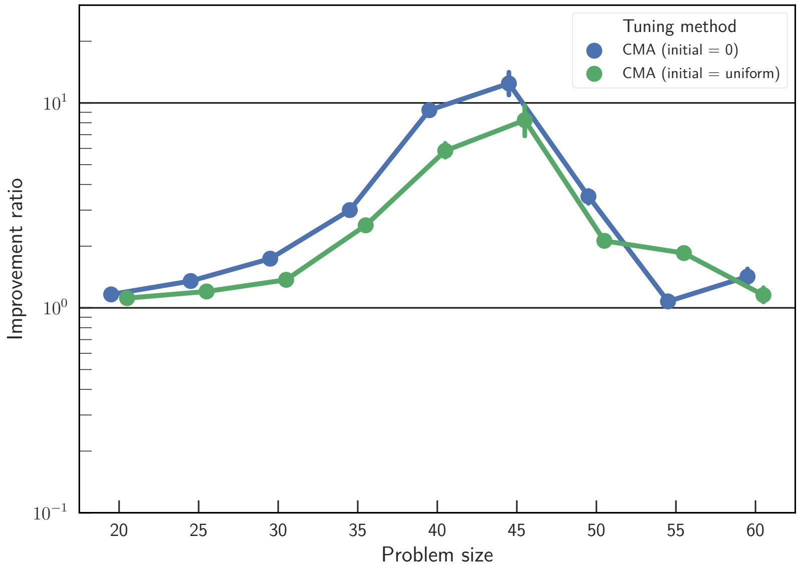


Figure 3.8: Ratio of mean success probabilities, before and after tuning, for each of the two strategies. All points are above 10^0, and indicate an improvement in probability. The peak is observed at $n = 45$ for both initial settings, with an improvement of 12.4 for the null configuration and 8.2 for the uniform configuration.

CHAPTER 4

Real-world combinatorial optimization

There are many differences between real-world optimization problems and canonical problems in NP. While the former may build on the latter, in many cases real-world problems convolve multiple problem classes, or add additional constraints and terms required to take into account real-world scenarios. For example, the Vehicle Routing Problem is a well-known NP-hard optimization problem, which in its canonical presentation requires only a given set of locations and a number of agents to describe completely. However, in real-world applications, road networks, scheduling time windows, and availability times all must be taken into account to correctly represent the problem in practice. These real-world conditions may impose additional constraints on the search space, yet often result in equally complex problems. For instance, the Capacitated Vehicle Routing problem with Time Windows, a more realistic representation of the problem where vehicle capacities and delivery time windows are included, is still NP-hard. Thus, to address solving such combinatorial optimization problems with QUBO/Ising formulations and quantum annealing, we must introduce methods to incorporate such real-world conditions in our modeling.

In this chapter we investigate two such techniques of solving real-world problems with quantum annealing methods, each of which are fundamentally different. The first involves deriving a QUBO from a real-life optimization problem in logistics (the less-than-truckload problem). We introduce generic methods to model the various constraints and optimization terms in QUBO form to accurately capture the real-world nature of the problem. The specific limitations imposed by using quantum annealing as a method to solve these QUBOs is addressed, and techniques are proposed, implemented, and tested in this context. The (sometimes unfavorable) scaling requirements of the problem description as a limiting factor is discussed in detail.

The second method takes an opposite approach to solving real-world optimization problems. Instead of deriving a QUBO to model the optimization problem directly, we utilize a canonical NP-complete problem– the set cover problem, which has a well-known QUBO form– as an oracle for a semi-supervised classification algorithm. We consider a variety of open source datasets of different types, and utilize various techniques to transform the data so that the task of data reconstruction (from a reference database) can be performed by the set cover QUBO. We show how the QUBO size and connectivity used to perform this task scale, and how these affect the performance of our method. Finally, we show how to use this reconstruction method to classify our data, and compare this to similar traditional classification algorithms.

4.1 Combinatorial optimization with real-world constraints

4.1.1 The shipment rerouting problem (SRP)

In order to introduce the idea of real-world applications of QUBO/Ising problems and quantum annealing methods, we derive a QUBO formulation of a real-world optimization problem as a case-study. We focus on a well-known problem in logistics: the less-than-truckload network service design. The term less-than-truckload (LTL) denotes shipments not exceeding a maximum weight significantly below a full truck load. This application is a common problem in the logistics industry, where complex delivery networks must be serviced regularly; for example, a country-wide mail delivery service is a kind of problem in this class. The transport of a single shipment in these scenarios can be described as a three step process: (a) a collecting truck run (where shipments are picked up from service locations), followed by (b) one or several linehaul truck runs (where shipments are passed through a network of connecting distribution locations) and ending with (c) a distributing truck run (a delivery to the shipments' final destination locations from a service hub). The work presented here focuses on the design of the linehaul network, or step (b). The linehaul network for LTL is made up by the set of terminals and timetable based truck runs, connecting the terminals and thereby producing the long-hauls of all the shipments entering the network. Taking limitations on transport times into account, the forwarding of the shipments is meant to be as cost efficient as possible, measured here as the total distance each shipment must travel from origin to destination. One key factor for cost efficiency is the consolidation of multiple shipments in jointly utilized trucks, at least regarding parts of their individual linehaul paths through the network. Meaning, it may be favorable to merge multiple partially-filled trucks which service similar routes, to reduce the total distance being traveled in the network. This saving measure is represented by an increase in truck utilization. However, the consolidation of multiple shipments with different origins and destinations in jointly utilized trucks requires detours of shipments, thereby possibly increasing the distance traveled for any *individual* shipment in the network. As detours come at a cost, the network design problem searches for an optimal trade-off between detour costs and the benefit of increased truck utilization. We focus on this central trade-off decision and call this problem (the middle step in the process) the *shipment rerouting problem* (SRP). We provide an illustrative example with two shipments in Fig. 4.1.

The input to our description of the SRP includes a set of terminals, their distances between each other, and a maximum number of available trucks between any two terminals. We have a set of shipments, each with a set of possible routes of intermediate terminals throughout the network. These possible routes already comply with constraints like maximum transport time or maximum detour factor, and we consider the (weighted) graph network they represent as a fixed input to the problem. These candidate routes include the direct route from the origin to the destination of the shipment, which are also used as the default for all shipments. Other candidate routes for rerouting are constructed in a pre-processing step based on the graphical structure of the terminals and the distances between them. Thus, a subset of shipments may be rerouted through alternate routes in order to reduce the overall distance all trucks travel to deliver the shipments. Each shipment has a size metric (volume, weight, etc.), and each truck has a corresponding capacity, i.e. an upper bound for the total shipment size that can be loaded. For our purposes, we denote the shipment sizes and truck capacities with respect to volume, and refer to them as such throughout the rest of this chapter. We note that the mathematical formulations we use to derive the final QUBO equally admit other quantities.

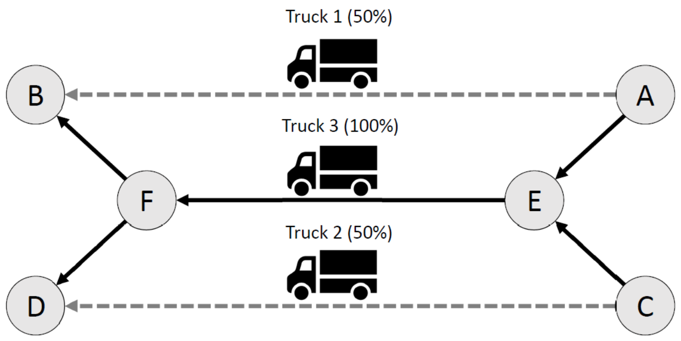


Figure 4.1: An example of the SRP with two shipments. The default routing (Trucks 1 and 2 carrying their respective shipments at 50% capacity each) is optimized by replacing Trucks 1 and 2 with a single truck (Truck 3) which can be fully utilized. The cost of rerouting each shipment to the route serviced by Truck 3 is offset by the removal of Trucks 1 and 2, thereby reducing the overall distance travelled to deliver the shipments.

In our case, a single shipment cannot be split across different routes. However, for transporting a shipment between two terminals, we may split it to distribute it on multiple trucks along the same route (this is necessary especially for shipments with large volume). Given the input to the problem, the optimization task is to decide on a route for each of the given shipments, which may include partial (or entire) overlaps between shipments. Consequently, the result includes the number of required trucks in the network, and which terminals are connected by truck runs in which frequency. Conceptually, this is similar to [98], where it was shown how to form a QUBO representation of a simple traffic flow combinatorial optimization problem. In that work, individual vehicles are given multiple candidate routes whose intersection needs to be minimized. This route-generation procedure is used here as well, but with opposite intent: our objective is to consolidate as many routes as possible. The SRP application we consider here is different from other QUBO formulations found in literature as not only the selection of routes for each shipment is variable, but also the number of trucks used on each edge along the path is selected by the optimization. Typically, such parameters are inputs to the QUBO construction which are then used to construct the appropriate QUBO. Our formulation thus incorporates elements from both scheduling (route selection in [98]) and packing problems (canonical problems in NP [77]).

4.1.2 Constructing a MIP for the SRP

Representing the SRP as a mixed-integer program (MIP) is straightforward, as we can use multiple kinds of variables (binary, integer, real) and constrain both the search space and solutions explicitly. Therefore, we start with a MIP representation which we then later will transform to a QUBO.

First, we represent the connectivity of the terminals as a weighted directed graph G where the vertices V are the terminals and the edges E between them represent the ability to transport shipments from any single terminal to another; in other words, we have an edge $e \in E$ from a terminal a to a terminal b if there are trucks available to drive from a to b. These trucks are called the *trucks on* e and the maximum available number of trucks is denoted by $t_{\max}(e)$. The weight of e is the distance from a to b and is denoted by $d(e)$. For each shipment s, $v(s)$ denotes its volume and $R(s)$ denotes the set of all routes that can be used to transport s (*candidate routes* of s). For each edge e, $R(e)$ denotes the set of all candidate routes which pass through e. A shipment s is *scheduled* on some edge e if s is transported using an associated candidate route r containing e.

In our scenarios, we assume all trucks have the same volume capacity, which we denote by c_{vol}. Moreover, all shipments have different origin-destination pairs so that no two different shipments have common candidate routes (however, their candidate routes may overlap). Therefore, for each candidate route r, we have a unique shipment $s(r)$ that can be transported using r. These choices simplify our scenario, but result in no less of a general mathematical representation of the problem.

Our objective is to transport each shipment entered in the network along an associated candidate route such that the total distance of *all* used trucks in the network is minimized. To represent this problem by a MIP, we introduce a binary decision variable y_r for each candidate route r that is 1 if r is used to transport $s(r)$, and 0 otherwise. For each edge e, we introduce a non-negative integer variable t_e

with maximal value $t_{\max}(e)$ representing the number of used trucks on e. Thus, we represent the problem by the following MIP:

Objective: Minimize the total truck distance

$$\sum_{e \in E} d(e) \cdot t_e \tag{4.1}$$

with respect to the following constraints:

Route-shipment constraints: For each shipment s, exactly one associated candidate route is used:

$$\sum_{r \in R(s)} y_r = 1. \tag{4.2}$$

Capacity constraints: For each edge e, the total volume of all shipments scheduled on e does not exceed the total volume capacity of the used trucks on e:

$$\sum_{r \in R(e)} v(s(r)) \cdot y_r \leq c_{\text{vol}} \cdot t_e. \tag{4.3}$$

The capacity constraints ensure that on each edge e, enough trucks are used to transport all shipments scheduled on e because we can split shipments to optimally exploit the truck capacities. Note that in an optimal solution, each truck number t_e is as small as possible, namely $\left\lceil \sum_{r \in R(e)} v(s(r)) \cdot y_r / c_{\text{vol}} \right\rceil$. In that case, for each edge e, we can completely fill all used trucks on e except possibly one truck that is partially filled.

4.1.3 From MIP to QUBO

As discussed thoroughly in Sec. 2.4, contrary to a MIP, a QUBO contains only binary variables and an objective function to be minimized without explicit constraints. Thus, additional variables and constraints must be modeled with penalty factors. Here, we show how to derive such factors to emulate the MIP constraints.

Our QUBO formulation also uses the binary variables y_r for the candidate routes r. In replacement of the integer variables t_e for the edges e, we use modified binary representations of their values in the QUBO based on a concept in [77]: for each edge e, we define $T(e)$ to be the set of all powers of two less than or equal $t_{\max}(e)$, and for each $n \in T(e)$, we introduce a binary variable $t_{e,n}$ to represent the number of used trucks on e by $\sum_{n\in T(e)} n \cdot t_{e,n}$. In this way, we can represent at least each number up to $t_{\max}(e)$, i.e. each allowed truck number. However, the highest number that can represented is $2n_{\max} - 1$ where $n_{\max}$ is the maximal value in $T(e)$. Therefore, to avoid representations of numbers greater than $t_{\max}(e)$, we reduce the coefficient $n_{\max}$ in $\sum_{n\in T(e)} n \cdot t_{e,n}$ by the surplus $s := 2n_{\max} - 1 - t_{\max}(e)$. The new expression is denoted by

$$\sum_{n\in T(e)} \overline{n} \cdot t_{e,n}, \tag{4.4}$$

i.e. we have $\overline{n_{\max}} = n_{\max} - s = 1 + t_{\max}(e) - n_{\max}$ and $\overline{n} = n$ for each $n \neq n_{\max}$. Now we can still represent each number up to $t_{\max}(e)$ but no other numbers. In our QUBO, we reformulate the total truck distance (4.1) as

$$\sum_{e\in E} d(e) \cdot \sum_{n\in T(e)} \overline{n} \cdot t_{e,n}. \tag{4.5}$$

To encode the route-shipment constraints (4.2), since that they are linear equalities, they can be added directly as $\lambda \cdot (A - B)^2$ where λ is a large penalty factor ensuring that the constraint is fulfilled at least in all optimal solutions of our QUBO. The capacity constraints (4.3), however, cannot be implemented in the same way (after reformulation using the representations (4.4)) because they are inequalities of the form $A \leq B$. However, as discussed in Sec. 2.4.1, such a constraint can be transformed into an equality $A + \ell = B$ by using a slack variable ℓ. In the context of the SRP, this slack of the capacity constraint for each edge e represents the wasted volume in the used trucks on e (*volume capacity slack* on e). However, determining the number of slack variables required must be determined from the other coefficients in the inequality. Here we consider the "packing" constraint in the QUBO formulation for the knapsack problem with integer weights in [77]:

$$H = A\left(\sum_{n=1}^{W} n y_n - \sum_{\alpha} w_\alpha x_\alpha\right)^2. \tag{4.6}$$

Here, w_α is the weight of item α, with associated decision variable x_α. The y_n variables function as unconstrained slack variables, allowing the total weight of the knapsack to be any integer less than or equal to W, as required by definition. Notice, however, that because integers are used for both the item weights and the slack variables, then all partial sums of item weights are represented by the slack variables. In the SRP problem, the volume $v(s(r))$ is not restricted to integers, and therefore needs more consideration. A naive attempt to use the standard inequality constraint would require us to consider all partial sums of shipments through every edge e in the graph network:

$$\left(\sum_{n=1}^{|R(e)|} \left(\sum_{m=0}^{{}_{v(s(r))}C_n} y_{n,m} \right) - c_{\text{vol}} \sum_{n \in T(e)} \bar{n} \cdot t_{e,n} \right)^2 = 0. \tag{4.7}$$

Here, $y_{n,m}$ would be a slack variable representing m possible sub-summations of n shipments (with ${}_{v(s(r))}C_n$ being the binomial coefficient). Clearly, this is not tractable and would result in significantly more QUBO variables than intended[1]. To overcome this problem, we discretize the shipment volumes into *bins*: We divide the capacity of each truck into the same number c_{bin} of equally sized bins. We call this c_{bin} the *bin capacity* of the trucks. Each bin can only be used for transporting one shipment and has the volume capacity $c_{\text{vol}}/c_{\text{bin}}$. Hence, for each shipment s, the number $b(s)$ of bins needed to transport s is given by $b(s) = \lceil v(s) \cdot c_{\text{bin}}/c_{\text{vol}} \rceil$. Instead of the volume capacity slacks, we now have to represent the *bin capacity slack* on each edge e, i.e. the number of unused bins in the used trucks on e. These slacks are more tractable because they are integers that can be assumed to be less than c_{bin}.

On the other hand, we must consider the case where c_{bin} is too small, and the bin volume capacity $c_{\text{vol}}/c_{\text{bin}}$ is large so that we may obtain several partially filled bins in the trucks, especially if shipments exist that are smaller than the bin volume capacity. Hence, we may not optimally exploit the truck capacities any more which may increase the number of used trucks. We can improve the situation by subdividing each bin into the same number of smaller bins.[2] Therefore, c_{bin} is a crucial parameter for the QUBO construction: more bins may lead to a better exploitation of the truck capacities, but at the cost of larger bin capacity slacks to be represented. In our experiments, we used the bin capacity 10, which was an empirically-determined compromise.

[1]Even considering duplicates in the partial sums, this is still factorially many terms in the worst case for every edge in the graph.

[2]Simply increasing the bin capacity may worsen the situation. For instance, suppose that $v(s) = c_{\text{vol}}/2$ for each shipment s so that $b(s) = \lceil c_{\text{bin}}/2 \rceil$. If $c_{\text{bin}} = 2$, then $b(s) = 1$ so that we can put two shipments into a truck. But if $c_{\text{bin}} = 3$, then $b(s) = 2$ so that we can put only one shipment into a truck.

For each edge e, we introduce a non-negative integer variable ℓ_e representing the bin capacity slack on e. Each bin capacity slack ℓ_e is less than c_{bin} so that we can represent these values in the QUBO by using a binary encoding scheme, introduced in Sec. 2.4.3. We define L to be the set of all powers of two less than c_{bin}, and for each edge e and for each $m \in L$, we introduce a binary variable $\ell_{e,m}$ such that the bin capacity slack of e is

$$\sum_{m\in L} m \cdot \ell_{e,m}. \tag{4.8}$$

In this way, we can represent at least each number less than c_{bin}, i.e. each relevant bin capacity slack. Finally, we can reformulate the capacity constraints as follows:

Capacity constraints: For each edge e, we have

$$\sum_{r\in R(e)} b(s(r)) \cdot y_r + \sum_{m\in L} m \cdot \ell_{e,m} = c_{\text{bin}} \cdot \sum_{n\in T(e)} \overline{n} \cdot t_{e,n}. \tag{4.9}$$

Similar to the route-shipment constraints, these capacity constraints are implemented in the standard way, by introducing them as a large penalty term of the form $\lambda \cdot (A - B)^2$. Putting all components together, we obtain the following formulation of the QUBO:

$$\begin{aligned}\text{Obj} = &\sum_{e\in E} d(e) \cdot \sum_{n\in T(e)} \overline{n} \cdot t_{e,n} + \lambda \cdot \sum_{s\in S} \left(\sum_{r\in R(s)} y_r - 1 \right)^2 \\ &+ \lambda \cdot \sum_{e\in E} \left(\sum_{r\in R(e)} b(s(r)) \cdot y_r + \sum_{m\in L} m \cdot \ell_{e,m} - c_{\text{bin}} \cdot \sum_{n\in T(e)} \overline{n} \cdot t_{e,n} \right)^2 .\end{aligned} \tag{4.10}$$

Here, all variables are as before, and S is the set of all shipments in the problem. We must now choose a penalty factor λ to ensure that only feasible solutions are present in the global optimum of the QUBO objective, so that it is never energetically favorable to violate one of the constraints in favor of minimizing the total truck distance. In general, we may choose any λ greater than the total truck distance d(*feas*) of any known feasible solution *feas* (for instance, the solution transporting each shipment on its direct route). To see the correctness of this choice, consider an optimal solution *opt* and suppose that *opt* violates a constraint. Then the *opt*-value of the QUBO objective is at least λ and thus greater than d(*feas*).

But since *feas* is feasible, $d(\mathit{feas})$ is also the *feas*-value of the QUBO objective, contradicting the optimality of *opt*.

The resulting QUBO representation now contains all the necessary terms to correctly model the SRP problem. However, it requires many more variables than the MIP in Section 4.1.2. This makes the problem more difficult to solve, both in sense of increasing the search space (in terms of number of variables) as well as the possibility of obfuscating optima via the choice in discretization (meaning, we may accidentally cause over/under filling in the trucks due to discretization). For each truck number variable t_e in the MIP, we have $|T(e)| = \lceil \log_2(t_{\max}(e) + 1) \rceil$ variables $t_{e,n}$. Additionally, we have $|L| \cdot |E| = \lceil \log_2 c_{\text{bin}} \rceil \cdot |E|$ variables $\ell_{e,m}$ to represent the bin capacity slacks. Thus, we can already see the overhead associated with conversion of a real-world optimization problem to QUBO form.

4.2 Solving real-world QUBO models

4.2.1 Generating SRP QUBOs from data

The inputs used in this work were generated from a real-world network of delivery hubs in Europe. The specific locations and distances between hubs were abstracted to comply with data protection laws, but are nonetheless representative of the original real-world network. Connections between hubs correspond to serviced routes between hubs. We used one graphical model to represent the entire hub network, and generated multiple inputs based on different numbers of shipments: 30, 50, 80, and 100 shipments. In all inputs, every shipment s_{ij} travels from one hub (v_i) to another (v_j). The direct route, $v_i \rightarrow v_j$ along e_{ij}, is always the first candidate route for s_{ij}. The other candidate routes are generated by a staggered *k-shortest path* approach: shipments are categorized by their origin-destination distance, and for each category the k shortest paths are calculated where k increases with respect to the origin-destination distance of the category. For example, shipments up to 200 km have one alternative route while shipments over 1000 km have up to 10 routes. The volume of the shipments is randomly generated using an adapted exponential distribution, resulting in many smaller shipments and few larger shipments. We show in Table 4.1 the number of QUBO variables and the number of total terms in the QUBO for each of the problems we generated.

Shipments	Routes	QUBO variables	QUBO terms
30	223	787	4856
50	428	1526	16315
80	752	2305	40594
100	925	3318	59014

Table 4.1: Number of QUBO variables and terms needed to describe the SRP instances.

4.2.2 Comparing QUBO and MIP solvers for the SRP

To understand the impact of our choices in modeling the SRP as a QUBO, we employed the use of multiple other solvers. Here we provide a brief introduction and motivation for each solver. It is important to note that in our inputs, all shipments have different origin-destination pairs. Therefore, two different shipments cannot have common candidate routes. However, candidate routes of different shipments may overlap in some edges.

Direct shipments. We consider the "direct shipment" solution to the SRP as a simple baseline for the other solvers to beat. The direct solution is computed by routing every shipment (s_{ij}) along its most direct path (e_{ij}). Since every shipment origin/destination is unique in our instances, this equates to using one truck per edge for every shipment.
Simulated thermal annealing. We use the same simulated thermal annealing algorithm used for experiments in Ch. 2 for the MIS instances. The specific implementation of simulated annealing in this analysis was from the D-Wave Python package here [99].
Tabu search. This algorithm is another metaheuristic for combinatorial optimization, operating on the principle that searching already-discovered solutions should be actively discouraged (a "tabu list"). Individual variables' states are flipped based on their likelihood of importance in the global optimum [100]. Solutions which worsen the objective function value may be explored by the search if no other variable flip is possible, which allows for both global and local refinement of solutions. The Python package used for Tabu can be found here [99].

Gurobi. Optimal solutions and optimality bounds were produced by solving the MIP in Section 4.1.2 using Gurobi, an exact branch-and-bound solver. The benefit of using Gurobi is that a bound on the optimality of the solutions is provided. Given that the objective function units are the same for all solvers, this optimality gap can be used for all solvers in this analysis. The runtime allocated to Gurobi was 24 hours per input to obtain good bounds for each instance.
D-Wave Hybrid Solver. As mentioned in Sec. 1.3, it is possible to construct algorithms which use QPUs in their inner loop, thus leveraging both classical and quantum resources. The smallest instance in our test set required 787 QUBO variables. While small for the SRP application, this is larger than could be solved on D-Wave QPUs at the time of experiments, which therefore necessitates the use of a hybrid algorithm in order to be solved with quantum annealing. We used a proprietary hybrid quantum-classical algorithm offered by D-Wave Systems, called the Hybrid Solver Service (HSS), which admits QUBOs with up to 10k binary variables. The HSS uses a QPU to optimize clusters of variables, allowing one to leverage the use of a quantum processor without the overhead of embedding. However, this hybrid algorithm does not allow direct access to control the QPU in its inner loop. Therefore, we consider the HSS as a black-box optimizer, and measure the performance as a function of the timeout parameter, similar to Gurobi and other black-box solvers.

We present the timing information allocated to each solver in Table 4.2, and the corresponding parameters in Table 4.3. For the D-Wave HSS, we limit the 30 and 50 shipment instances to only 5 minutes of runtime. We note that these 5 minutes were sufficient for the problems tested. Because we could not control the usage of the QPU in the D-Wave HSS, we report the QPU runtime in the timing results rather than a parameter. All software solvers were executed using single-threaded programs. To attempt a fair comparison, each QUBO solver was given roughly the same amount of time per test instance. However, the specific parameter choices corresponding to such times were found and set by hand.

We present the consolidated results from all solvers in Figure 4.2. While the total runtime of Gurobi was set to a 24 hour timeout (to obtain good lower bounds), good solutions with an optimality gap of less than 10 percent were already found after a few minutes for all instances. For the 30 and 50 shipment instances, we also obtained provably optimal solutions within the first few minutes of optimization. The solutions from Gurobi were significantly better than those obtained by solving

Instance	Simulated Annealing	Tabu	HSS
30	1 hr	1 hr	5 min (QPU: 3.0s)
50	1 hr	1 hr	5 min (QPU: 1.4s)
80	1 hr	1 hr	1 hr (QPU: 3.61s)
100	1 hr	1 hr	1 hr (QPU: 4.34s)

Table 4.2: Table of runtime allocated to each solver in the experimental setup.

Instance	Simulated Annealing	Tabu	HSS
30	2500 samples, 50000 sweeps	1 hr timeout	5 min timeout, $use\_qpu = True$
50	1600 samples, 50000 sweeps	1 hr timeout	5 min timeout, $use\_qpu = True$
80	1000 samples, 50000 sweeps	1 hr timeout	1 hr timeout, $use\_qpu = True$
100	500 samples, 50000 sweeps	1 hr timeout	1 hr timeout, $use\_qpu = True$

Table 4.3: Parameter sets used for each solver. Parameters not mentioned were set to default values.

the QUBO formulation. However, this is possibly due to both the fact that Gurobi is an exact solver and the way in which the MIP is discretized to form the QUBO, as explained in Section 4.1.3. The lack of discretization for Gurobi may result in more efficient packing of shipments along each edge, which would result in fewer trucks, and therefore a lower objective function value (truck km). Tabu search was able to find a near-optimal solution for the 30 shipment instance, but was unable to find even feasible solutions for any of the other instances. Simulated annealing was able to find feasible solutions, but only in the largest case of 100 shipments was the solution better than the direct shipment approach. The D-Wave HSS was able to find better-than-direct solutions for the 30, 50, and 80 shipment instances.

Throughout our initial experiments, we found that increasing the number of possible routes for each shipment does not directly correlate with improved solutions to the original problem (lower total truck km). This is due to the fact that each additional route creates more minima and a more rugged landscape. It is important to note that given the way we construct the QUBO– no trucks along an edge is a

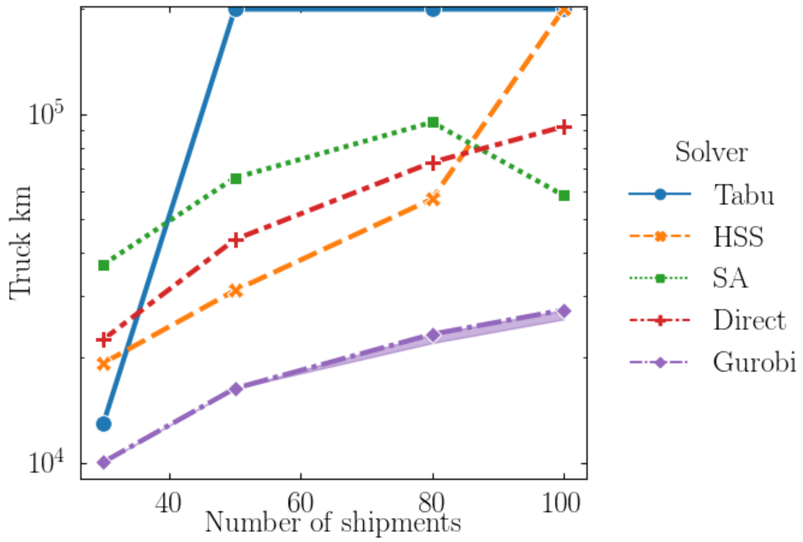


Figure 4.2: Performance of all solvers used in the experiments. We display the results in units of truck kilometers for ease of comparison. Simulated annealing (SA), Tabu, and the D-Wave HSS are QUBO solvers, Gurobi is a MIP solver, and the direct solutions are the simple baseline of one truck per shipment.

valid solution– increasing the number of possible routes can only create additional minima, not remove minima that have already been created. Furthermore, given that all constraints are implemented as penalty factors in the QUBO, this created an increased difficulty for the QUBO solvers to find optima. Given this insight, it is even more important to consider the number of QUBO terms (in Table 4.1) when modeling combinatorial optimization problems as QUBOs.

In general, both the SRP (in its original form) and the QUBO model the distance minimization of a simple objective function– total number of truck kilometers used to send shipments between nodes in a graph. The majority of our work focused on deriving methods to translate the MIP representation of SRP to a QUBO using both simple minimization objectives (truck kilometers as weights on decision variables), and hard constraints (packing constraints on edges in the graph) to test both quantum and classical optimization algorithms. In reality, it is evident that there is a significant amount of work required to find such valid QUBO representations for complex optimization problems inspired by real-world constraints. Despite the relatively straightforward description of the problem, correctly modeling the solution landscape requires a more subtle approach, and required multiple iterations of derivations, as explained throughout the text. Of the algorithms tested, Gurobi performed the best despite being an exact branch-and-bound algorithm. Of the heuristics, we found that the D-Wave HSS was able to find better than greedy solutions for the smaller problem sizes tested. We stress that given our small test bed we cannot conclude any one solver being the best relative to the others, nor was this the intention. The context of the work presented was to assess the work that was required to transform a real-world optimization problem to a clearly-defined QUBO equivalent. We found that the bar defined as "acceptable" (finding solutions that are better than direct shipments) was surprisingly difficult for the heuristics to beat. This is important to note since simulated annealing was able to find valid solutions for all the problem sizes, but better-than-direct for only one problem (the largest). Furthermore, long runtimes were required to find these better-than-direct solutions, and yet were still far from optimality. The significance of this result is that despite QUBO being an NP-hard problem, the overhead in transforming any single optimization problem to QUBO may be detrimental to the performance of optimization algorithms which then solve the QUBO, and therefore it is not always worth the effort of the transformation. Furthermore, throughout the studies conducted in this section we found that naive and straightforward transformations to QUBO using known techniques is sometimes impractical. In particular, the inequality constraints required to correctly pack the shipment volumes in the truck capacities required such an increase in the number of variables, that additional data transformation was required to encode the constraint in the QUBO (discretization of the capacities).

4.3 Adapting real-world optimization problems to known QUBOs

We now turn our attention to a different approach to solving optimization problems. Here we perform the task of data reconstruction and classification using a QUBO model. Given some set of time series (TS) data, and a reference database, the task of reconstructing the candidate time series from features in the database is a hard problem. In our approach, we use a combinatorial optimization "oracle" in the form of a QUBO problem to model the task of the time series reconstruction, thus solving the underlying problem. In particular, we use the QUBO representation of the set cover problem as the oracle performing this task. In contrast to the previous section, where the majority of the work was in deriving a valid QUBO, here we instead focus our work on finding methods to transform the data so that it can be solved effectively by the set cover problem without oversimplifying the original data. This is also in contrast to other quantum machine learning techniques, where it has been shown how to reformulate parts of classical classification algorithms as quantum subroutines that can be executed on error-corrected gate-model QPUs [101, 102, 103, 104]. In quantum annealing, similar numerical approaches have been shown in which the objective function of the classification task (minimizing distance metrics between high-dimensional vectors) has been directly translated to a QUBO, with each vector's possible assignment represented via one-hot encoding to physical qubits [105, 106].

By reformulating the critical task in our classification algorithm as a set cover problem, we introduce two novel ideas to quantum classification algorithms: (i) we avoid representing single vectors with polynomial numbers of qubits, instead representing the features within the data as the qubits, and (ii) we perform the classification task by transferring the core concepts of classification (and reconstruction) to the quantum algorithm for set cover, as opposed to a direct translation of a distance-based minimization procedure. This results in an algorithm that avoids a classical "learning" procedure, therefore requiring significantly fewer computational resources compared to other classical and quantum methods.

4.3.1 The set cover problem

The set cover problem is defined as follows: given a set of symbols (called a *universe*) $U = \{1, ..., n\}$, and a set of subsets V_i, such that $U = \bigcup_{i=1}^{N} V_i$, $V_i \subseteq U$, find the smallest number of subsets V_i whose union is U. This is a well-known NP-hard optimization problem, and is one of Karp's original 21 NP-complete problems [50]. There is a known QUBO formulation for the set cover problem provided in [77]. We start by defining the following binary variables:

$$x_i = \begin{cases} 1, & \text{if set } V_i \text{ is included,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.11}$$

We let $\alpha \in U$ denote an element of the universe set, and m signify if element α appears in m subsets. Then, we have binary variables:

$$x_{\alpha,m} = \begin{cases} 1, & \text{if the number of } V_i \text{ which include } \alpha \text{ is } m, \\ 0, & \text{otherwise.} \end{cases} \tag{4.12}$$

We consider the full QUBO as a sum of two components:

$$H_A = A \sum_{\alpha=1}^{n} \left(1 - \sum_{m=1}^{N} x_{\alpha,m}\right)^2 + A \sum_{\alpha=1}^{n} \left(\sum_{m=1}^{N} m x_{\alpha,m} - \sum_{i:\alpha \in V_i} x_i\right)^2, \tag{4.13}$$

and

$$H_B = B \sum_{i=1}^{N} x_i. \tag{4.14}$$

The complete QUBO is given by $H = H_A + H_B$. The first summation in H_A imposes that exactly one of $x_{\alpha,m}$ must be selected in the minimum via a one-hot encoding. The second summation in H_A represents the number of times α is selected, and that this is equal to the number of selected subsets α appears in (m, as only one $x_{\alpha,m}$ can be 1 in the minimum). This is similar to the packing constraint from the previous section. The final term, H_B, serves to minimize the number of V_i needed to cover the universe U. The total number of variables required is $N + n(1 + M)$, where M is the maximal number of sets that contain given element of U. The limiting case where each element of V_i included covers only one element of U constrains the coefficient of H_A and H_B to $0 < B < A$. The closer the coefficients B and A, the more weight is given to (4.14), minimizing the number of elements selected from V.

4.3.2 Time series reconstruction as a QUBO

Classification techniques generally require specific data representation, similarity measure definitions, and algorithm selection. Similarly, in our QUBO approach, we represent the time series data as encoded strings from which we formulate semi-supervised classification and optimal reconstruction as a set cover problem, and provide metrics based on solutions to the set cover problem. While different than classical approaches [107, 108, 109, 110], we do not attempt to simplify the complexity of the problem, and introduce a method that is based on latent features within the data. The only assumptions we make about the time series data is that it is separated into two categories: a training set and a test set. The training set we assume is labeled and is used as a reference database with which each time series in the test set is reconstructed from.

In order to reconstruct given time series data, we start by discretizing both the training and test data, and compare the encoded strings to generate the elements of our universe to form the set cover. This technique is crucial to allow feature-wise comparison of the data, as well as arbitrary reconstruction using existing (or training) data. There are many ways to discretize time series data, and exploring the trade-offs between the various methods is beyond the scope of this thesis. For our purposes, we use the symbolic Fourier approximation (SFA) method [111], as it provides differentiation between separate time series classes and features in high-dimensional data sets, allowing us to use these representative symbols for our set cover problem. Nevertheless, the exact discretization is data-dependent, with various hyperparameters (such as number of letters in the alphabet, length of each encoded string, etc.) present in the method. We therefore assume, under simple conditions, that we can treat the SFA method as a black-box that takes time series data as an input and returns symbolic strings encoding the data features as output.

Given the encoded strings, we introduce a pair-wise method to compare the time series features using what we call a “pulling procedure”, illustrated in Figure 4.3. This pair-wise comparison is considered a pre-processing step necessary to formulate our set cover problem. Starting with one fixed string (red in the figure), we consider each encoded character as an independent element in the universe set[1] ($U = \{0, 1, 2, 3, 4\}$ in the figure). A second string (green in the figure) is compared element-wise by successively moving the second string along the first, as illustrated. At every iteration, all character matches between the two strings are recorded as a new set. In the example from Figure 4.3, the set of sets is $V = \{\{0\}, \{\emptyset\}, \{0, 2\}, \{\emptyset\}, \{1, 2, 3\}, \{\emptyset\}, \{\emptyset\}, \{3\}, \{\emptyset\}\}$.

The procedure is repeated for the rest of the encoded training tie series to form the set of sets V. In this setup, the SFA routine needs to be performed only once per time series, and that the pair-wise comparison is then performed in $O(n^2)$ time. The set which is a union of all subsets obtained via the pulling technique now represents the all features in common between the target (or test) time series

[1]It is important to note that by using the same encoding scheme for all time series data, we ensure that all string characters belong to the same alphabet.

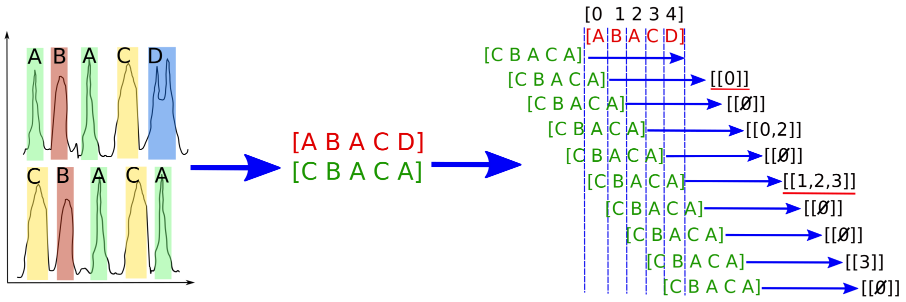


Figure 4.3: Schematic illustration of time series encoding and pulling procedure to produce subsets of set $V = \{\{0\}, \{\emptyset\}, \{0,2\}, \{\emptyset\}, \{1,2,3\}, \{\emptyset\}, \{\emptyset\}, \{3\}, \{\emptyset\}\}$. The optimal selection to cover $U = \{0,1,2,3,4\}$ in this case would be underlined subsets $V = \{\{0\}, \{1,2,3\}\}$ with item numbers 0 and 4.

candidate and all other time series in the reference data set. Given this aggregate set, the goal is now to select the minimal subset that most closely reconstructs the universe, which is the NP-hard set cover problem. In other words, the task is to select the features in the reference database which correctly reconstruct the given test time series. In the case illustrated in Figure 4.3, the optimal selection of subsets is underlined in red. In principle, solutions of this set cover problem do not preserve order of elements, and allow the use of the same element multiple times. This feature is useful for time series comparison, as elements of the time series data can be permuted and duplicated without affecting our reconstruction method.

The final size of the set cover QUBO is heavily dependent on our choices during discretization. For example, the number of binary variables is equal to $N_{\text{train TS}} (2L-1)(L+1)$, where $N_{\text{train TS}}$ is the number of time series in the training set used for reconstruction, and L is the length of string that encodes the time series. Increasing the string length to encode each time series changes the size of the universe U. Allowing longer encoded strings to represent the data creates more subsets V_i. Therefore, there exists a trade-off between the granularity of the encoded strings and the ability to solve the set cover representation of the problem. Including more characters in our alphabet for discretization changes the non-empty sets V_i, which the number of quadratic elements in the QUBO depends on. The general trend is, however, that the number of the quadratic element decreases with the increase of the characters used in our alphabet. This is explained by the properties of the pulling procedure described above, since a smaller alphabet produces more non-empty elements V_i which could be used for reconstruction of the universe U. In Figure 4.4 we show how varying these hyperparameters of the discretization affects the size of the QUBO problem, based on 20 test samples from one of our data sets used in experiments [112].

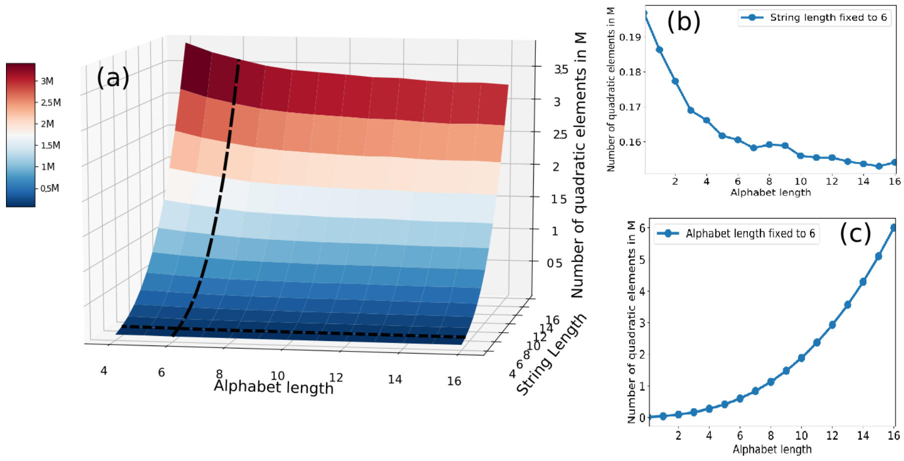


Figure 4.4: (a) The number of quadratic terms in millions as a function of string and alphabet length. (b) Quadratic elements as a function of alphabet length, with string length being fixed to 6. (c) Quadratic terms as function of string length, with alphabet length being fixed to 6. The corresponding isolines (b) and (c) are shown with dashed line on surface plot (a). Analysis was performed using 20 test samples from the BeetleFly data set [112].

4.4 Using QUBOs to perform classification

We can now combine the methods described in the previous sections– constructing the sets U, V from discretized data and the QUBO representation of the set cover problem– to perform semi-supervised classification from reconstructed data sets. In our case we use training data sets with known labels, and the task we solve is to use the labeled data to assign labels to the test set, given a valid reconstruction. Normally, the training set with labeled data is significantly smaller than unlabeled test set, which we exploit in our method.

We encode both the training and test data sets into strings using the pulling method described previously. We then perform the reconstruction procedure for every time series in our test set using the entire training set. Each time series from the test set is assumed to individually form a universe U, and is to be reconstructed using the sets V_i, obtained via the pulling procedure. Explicitly, using Figure 4.3, the red string is the time series from the test data set, and all strings in the training set are pulled through (green strings) to obtain the V_i's. This allows us to compare every test time series to the full training set in one-versus-all manner. Then, using the universe U and V_i's from the pulling procedure, we formulate the set cover problem outlined in Section 4.3.1. Thus, a single solution to that set cover problem (even sub-optimal in the worst case) allows us to reconstruct each time series from the test set using a set of discretized features obtained from *all* elements which appear in the training set. Furthermore, since we employ metaheuristics to solve the set cover QUBO, various optima could yield different ways to reconstruct the test time series using the training set. Due to this, it is therefore the users' task to use these reconstructed strings to associate each test time series with a label from the training set.

To classify the reconstructed test time series data we evaluated three different similarity metrics using set cover solutions: largest common subset V_i, highest number of common subsets V_i, and largest sum of common elements in selected V_i. We briefly explain how each metric is calculated, and discuss the performance of each.

- **Largest common subset.** Given a candidate solution to the set cover problem, the label corresponding to the V_i which contains the most elements is selected. The label is then assigned to the test time series. This metric captures the longest continuous set of features from the training time series data, and assumes that is sufficient to determine the label.
- **Number of common subsets.** Frequently, multiple V_i's from the same training time series are used to reconstruct a test time series. In this metric, we count the number of V_i subsets used to cover the universe. The test label is assigned the same label as the training time series which appears most frequently in the set cover solution.
- **Largest sum of subsets.** This metric is a combination of the previous two. For every training time series that is used to reconstruct a test set, the total number of elements used by each is counted (summed over all V_i's). The

label which corresponds to the training time series with the largest sum is assigned to the test time series.

These metrics allow us to quantify the accuracy of our semi-supervised classification method. The first two metrics, being based on large sets of common features between the time series, performed the best (experiments and results shown in the next subsection). There was no significant difference between the two metrics, and the superiority of one metric over the other varied between data sets. The third metric, which was a combination of the first two, performed worse than either of the first metrics in the majority of the cases tested. While unexpected to begin with, this observation could be explained by the fact that because the third metric admits matches with many small subsets V_i that are selected in the set cover, this metric could miss significant signatures present in the time series data. Therefore, the largest common subset metric was selected for the experiments presented in the next section. It should also be noted that the use of labeled training data is not designed to not reach the accuracy of supervised learning methods. Moreover, there are modifications that could be made to the methods presented to improve the accuracy, for example increasing the word length and/or using a larger train set. Both are constrained in our use-case to prohibit excessively large QUBOs from being constructed. The goal of this method is to allow for relatively high accuracy using small sets of training data.

4.4.1 Classifying real-world time series data

We validate our method by using the set cover QUBO to reconstruct and classify a variety of open-source, real-world time series data. The benchmarking experiments performed here used labeled time series data available publicly [113, 114]. These data sets were used as-is in the experiments presented below. Validation was performed by measuring the classification rates of each methods on the labeled test data. We restricted the analysis to univariate time series data with two classes and small training set size. However, this method of semi-supervised classification can be used with any number of classes, at the cost of QUBO size. Since both the number of time series in the training data and the word length used to encode the data contribute to the number of variables in the QUBO, we select data sets that have small numbers of time series in the training set. The test and training sets used in these experiments are already determined and labeled by the source, allowing us to easily calculate the classification rate of our method and avoid the step of selecting a training set. To benchmark the performance of our classification method, we compared the accuracy of our labeling method to semi-supervised and unsupervised classical classification methods. The results of these experiments for the various data sets are summarized in Table 4.5. To test the robustness of our method we collected a variety of data sources of different types. We briefly review each source and provide a literature reference for further details. We note that in the data sources' accompanying cited works, higher classification rates than our methods are reported using supervised algorithms. In this analysis we do not consider supervised classification algorithms, and instead compare our semi-supervised quantum-based approach to similar classical algorithms.

SonyAIBORobotSurface1 [115] data is sensor data collected from a small, dog-shaped, quadruped robot. It is equipped with multiple sensors, including a tri-axial accelerometer. In the experiments we classify between roll accelerometer measurements on two classes of surfaces: soft carpet and hard cement.

GunPoint [116] data includes motion tracking of actors' hands during gun-drawing and gun-pointing actions. For both classes the X-component of the actor's right hand centroid is tracked and used to distinguish between the two classes.

TwoLeadECG [117] and **ECG200** [117] are electrocardiogram data sets available at the PhysioNet database [118]. The first includes long-term measurements from the same patient using two different leads. The classification task aims to differentiate between each lead signal. In contrast, the second ECG200 set contains electrical activity recorded during one heartbeat. The two classes are the normal heartbeat and a Myocardial Infarction records.

BeetleFly [112] time-series data is generated from binary images developed for the testing of shape descriptors. The external contour of these images is extracted and mapped into the distance to the image center. The two image classes are contours of beetles and flies.

Chinatown [119] data is collected by an automated pedestrian counting system in the city of Melbourne, Australia. The classes are based on weekday or weekend traffic.

The QUBOs generated by our methods were too large to be embedded and optimized using the largest available QPUs (D-Wave 2000Q at the time of experiments). The exact sizes of the QUBOs for each data set are shown in Figure 4.5. To solve the QUBOs we used the simulated thermal annealing metaheuristic which was used for previous experiments in this thesis. The specific implementation of SA was from the D-Wave Python package for classical QUBO optimizers [99]. We found that 20,000 samples and 1000 SA sweeps (with geometric interpolation of the inverse temperature) were sufficient to ensure that low-energy local minima were sampled within reasonable times per QUBO. We use the default SA settings in the package for initial and terminal inverse temperature selection (for more information about the implementation of SA we refer the reader to [99]).

The specific parameters used for the time series encoding to generate the QUBOs are shown in Table 4.4. In general, the longer the time series are and the fewer time series are in the training set, the finer the discretization method required to accurately classify the test data. In all data sets we were able to reconstruct each test time series with elements from the training set, as explained in Section 4.3.1. The distributions of the number of variables in each QUBO for all data sets is shown in Figure 4.5.

We provide an illustrative example of our QUBO-based reconstruction and classification in Figure 4.6 using the BeetleFly data set. The task is to reconstruct the data in Figure 4.6 (a) using (b) and (c). For this example, an alphabet of size 5 was used for encoding, color-coded in the figure. The results of the set cover problem, formulated using the methods explained in previous sections, are three sets, shown as v_1, v_2, and v_3 in Figure 4.6. Meaning, each box (representing a fifth of the time series data per box) that appears in one of the subsets forming the solution is designated as such. Specifically, $v_1 =$ [‘A’, ‘E’], $v_2 =$ [‘E’, ‘B’], and $v_3 =$ [‘C’]. Therefore, the union $v_1 \bigcup v_2 \bigcup v_3 = U$, where $U =$‘ACEEB’, the test time series data to reconstruct. For classifying the reconstructed sample, we refer to the classes of the training data used for the reconstruction, and note that the training samples in Figure 4.6 (b) and (c) belong to two different classes. Using the similarity metrics defined above, it is easy to determine that v_1 and v_2 both originate from the time series (b), whereas only v_3 (which contains only a single element) is obtained from (c). Therefore, (a) is assigned the same label as (b). This example is representative of the majority of cases encountered during classification, with components of the reconstructed time series varying across multiple training samples, and often also across multiple classes.

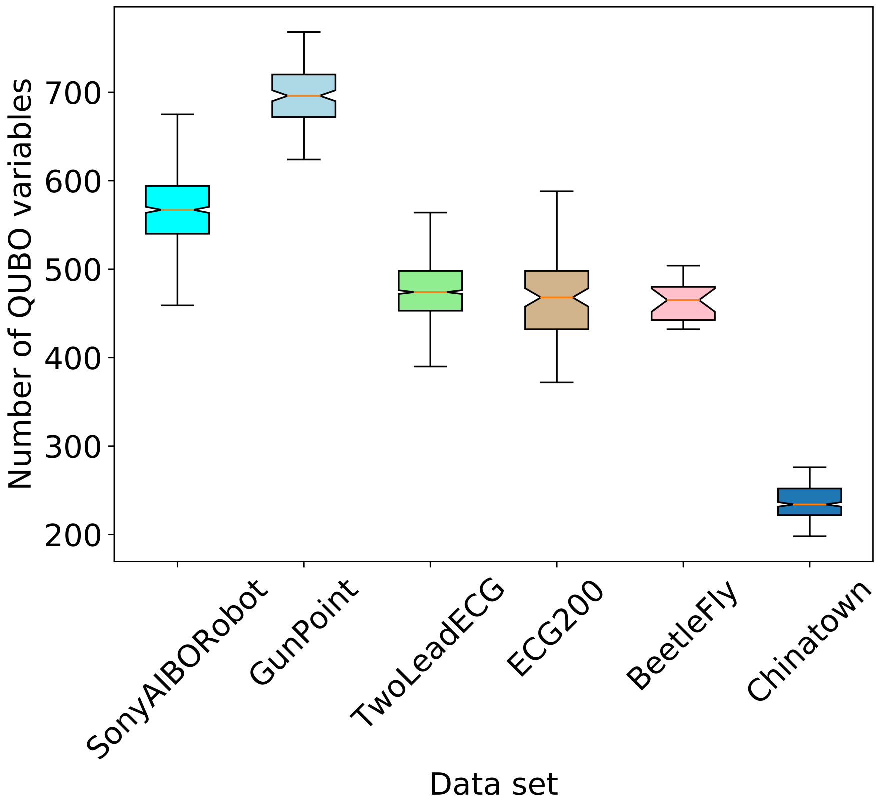


Figure 4.5: Distribution of number of QUBO variables for all data sets in Table 4.4.

4.4.2 Classification benchmarking

For the purposes of evaluating our QUBO-based classification method quantitatively, two classical time series classification algorithms were compared based on dynamical time wrapping (DTW) [120] measures: k-means classification and a classical analogue of the semi-supervised method described above. The motivation for using these specifically is that both are based on pair-wise similarity metrics as in the approach presented here. DTW applied to temporal sequences aligns the pair series in a non-linear way to minimize differences and calculate Euclidean distance afterwards. The DTW measure could be applied directly in unsupervised k-means classification or similarly to the method described here in the semi-supervised fashion. We use k-means classification with pairwise DTW metrics calculated

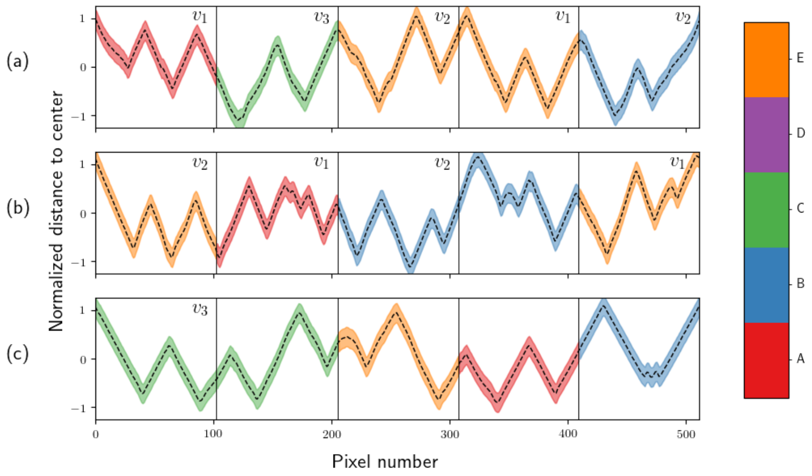


Figure 4.6: An illustrative example of reconstruction and classification from the BeetleFly data set. (a) A test time series sample (encoded as 'ACEEB') reconstructed from two training TS. Each box in the sub-figure is encoded as a single letter in a string, as per the color bar. The subsets v_i obtained from the pulling procedure and used to reconstruct this data are shown both in the reconstructed (test) time series and in the training time series. (b) The first training data used for reconstruction and classification (encoded as 'EABBE'). (c) A second time series used for reconstruction (encoded as 'CCEAB').

on the original TS (before encoding), with the labels being assigned based on belonging to one of two clusters. The second method assigns the test TS labels are by the DTW metric directly, calculated pairwise between each training and test TS (without encoding). We use these two methods to calculate classification rates for all data sources in the experiments.

As expected, the semi-supervised QUBO-based method outperforms classical unsupervised methods. We note however, that the QUBO-based method operates on a reduced dimensionality in contrast to the classical methods which use the original TS, where full information is preserved. Even under this consideration the accuracy of QUBO-based method is comparable with the semi-supervised DTW methods, and could be improved still by enriching the set V, i.e. by augmenting the training set or increasing the discretization granularity. The worst performance of the QUBO-based algorithm is observed on the TwoLeadECG data set. This

Data set	**Data type**	**Train/Test size**	**Time series length**	**Word/Alphabet length**
SonyAIBORobotSurface1	Sensor	10/601	70	8/8
GunPoint	Motion	30/150	150	5/5
TwoLeadECG	ECG	20/1139	82	5/5
ECG200	ECG	20/100	96	5/5
BeetleFly	Image	20/20	512	5/5
Chinatown	Traffic	20/345	24	5/5

Table 4.4: Table with data set description, number of time series in training and test sets, length of time series, and length of each encoded string and number of different letters used to encode data set.

could be explained by the nature of our method, as well as the sensitivity of the ECG data. By using the set cover problem, we allow for permutations of subsets of TS data in the reconstruction of the test TS. It is likely that this permutation of TS segments, and similar representation in Fourier space of the signals from the two leads in the ECG measurements, makes our method not suitable for this kind of data. The highest accuracy is obtained using the BeetleFly and Chinatown data sets. In the first case, many permutations of the training set to construct the test set are permissible, which our method takes advantage of. The accuracy of our method is additionally improved by the relative size of the training set, further augmenting the combinatorial space of permutations. This robustness can also be explained by the dimensionality reduction technique for this data set: the 2D BeetleFly images (with different orientations) were mapped to 1D series of distances to the image centre, which again is beneficial for permutation-based methods. The Chinatown data set, for comparison, contained significantly shorter TS than BeetleFly. Encoding the Chinatown TS data with the same word length as BeetleFly resulted in higher granularity representations, and ultimately higher accuracy. This provides additional evidence that the accuracy of our method can be improved by increasing the granularity of the encoding.

Among the advantages of our method is the utilization of significantly less data with respect to conventional classical methods, as well as a one-versus-all comparison that allows the selection of segments of data from multiple sources to reconstruct a single time series. This provides an additional robustness in the method with respect to permutations of time series segments during the reconstruction. In order to formulate this problem as a QUBO we apply time series dimensionality reduction by encoding each time series as a separate string. This encoding procedure and

Data set	**QUBO method**	**k-means**	**DTW**
	c1/c2/weighted	**c1/c2/weighted**	**c1/c2/weighted**
SonyAIBORobotSurface1	0.7/0.9/0.78	**0.85/0.97/0.92**	0.97/0.63/0.83
GunPoint	**0.76/0.79/0.78***	0.53/0.51/0.52	**0.82/0.77/0.79***
TwoLeadECG	0.6/0.62/0.61	0.65/0.7/0.68	**0.86/0.94/0.9**
ECG200	0.61/0.82/0.75	**0.62/0.8/0.79**	0.87/0.51/0.64
BeetleFly	**0.85/0.89/0.87**	0.64/0.83/0.73	0.62/1.0/0.82
Chinatown	0.72/0.91/0.86	0.37/0.78/0.67	**0.89/0.98/0.94**

Table 4.5: The classification accuracy measured on two classes and weighted average reported for QUBO-based and classical DTW-based methods. Bold text signifies the most effective classification method (based on the weighted average of the two classes) for each data set tested. Asterisk denotes a tie between the methods within statistical variance.

selection of comparison metrics define the hyperparameter space of the problem. The QUBO-based classification method performed the best on image and traffic data, which is consistent with our method's inherit ability to utilize permutations of features/data within the time series to perform reconstruction.

Time series reconstruction and classification has a wide variety of useful applications, such as: management of energy systems, factory process control, sensor systems, and many more. The methods introduced in this section show how to reformulate the tasks of reconstruction and classification of real-world data so they can be solved as QUBOs. This is a fundamental departure from the traditional methods used in solving optimization problems with QUBOs, and so we consider this a novel contribution to the field of optimization of real-world problems which can be built on in the future.

CHAPTER 5

Hybrid quantum algorithms for real-world optimization

While the potential benefits of quantum computing and quantum annealing are well-motivated, it is clear from the previous sections that there are several limitations when trying to solve optimization problems directly in current-generation QPUs. In recent years, classical software development has complemented the quantum hardware in an attempt to bridge the gap between the restrictive quantum hardware and more real-world applications of quantum annealing. One of the proposed ways to use quantum annealing in practice is the concept of *hybrid quantum-classical* algorithms. With these, algorithms are constructed so that the quantum annealing QPU is used in some way by an inner loop (as a sub-solver/sampler, a mutation step in a genetic algorithm, etc.), thus offloading some difficult critical task to the QPU. These kinds of algorithms have been used in the past to tackle arbitrarily-structured optimization problems, in an attempt to solve larger and more realistic combinatorial optimization problems [37, 38, 39]. To support the development of these algorithms, D-Wave Systems released a Python package dedicated to constructing such hybrid algorithms, called `dwave-hybrid` [99]. In this chapter we investigate the construction and use of hybrid algorithms using these tools, as well as the black-box hybrid optimization algorithm provided by D-Wave System. We motivate and contextualize their use through real-world combinatorial optimization problems, and build custom optimization routines to solve QUBO/Ising problems in real-time.

5.1 Motivating a real-world traffic optimization use-case

One of the first real-world applications demonstrated for quantum annealing in practice was traffic flow optimization [98]. This was motivated by the increased focus on autonomous driving, smart cities/infrastructure, and the need for cutting-edge computational resources to handle the complex task. In order to test this in practice, a pilot project was conducted in which the QUBO outlined in [98] was solved in real-time to navigate a fleet of buses, providing a turn-by-turn navigation service for the Web Summit 2019 conference in Lisbon, Portugal. The motivation for this project comes from observations in the automotive industry, that as cities around the world continue to grow in both size and population, traffic congestion becomes an increasingly prevalent problem [121]. This is especially apparent during events that congregate large numbers of people for specific periods of time. For example, conferences, sporting events, and festivals can cause temporary but significant disruption to the cities' transportation systems, resulting in delays for the residents of those cities [122, 123]. A key issue is that permanent transportation infrastructure, such as rail lines or roads, are costly and slow to modify given the temporary nature of these events. In light of this, the advent of smart traffic management systems offers possible improvements to existing transportation systems with minimal overhead in regards to implementation. Some requirements for such systems include the management of the mobility flows in real- or near to real-time using flexible and modular software components. The goal of this project was therefore meant to address two very specific questions related to applying quantum annealing in a real-world scenario:

1. How do we design customized bus routes to avoid traffic congestion during big events?
2. How would one build a real-time production application using a quantum processor to manage such a traffic system?

In order to address both of these questions, we separate the work into two separate phases: the first is concerned with understanding the input required for such a live navigation service to run in practice, and the second phase presents the construction of hybrid optimization service that complies with the demands of the first. Concretely, the goal is to adapt the model presented in [98] to a live setting, and test the assumption that using quantum annealing to avoid overlaps between vehicle routes reduces traffic congestion. Thus, we are able to construct meaningful optimization problems in an application setting, and address the application needs using hybrid quantum techniques.

To answer the first question, we needed to consider real-world road networks and movement data as input to our navigation service. This required us to find and fix (possibly multiple) start and endpoints of a custom bus service, where each bus had its individual route customized by considering all other buses in the fleet in real-time, as described in detail in [98]. By analyzing origin/destination (OD) matrices of peoples' movement data, detailing the volume of movement streams from the Web Summit conference venue (Altice arena) to different zones of the city of Lisbon on an hourly basis, we were able to identify locations of interest for our navigation service. The results showed a total of 225 OD matrices in a study area of 93 zones throughout Lisbon, shown in Figure 5.1. This data included demographic census data, mobility behavior from surveys, Lisbon traffic counts, floating car data, mode of choice and network models from the city, for both dates within and outside the Web Summit time frame.

Three express bus lines were proposed to serve the demand from the selected zones to the Web Summit: a red, green, and blue line, with a total of 23 bus stops. One express line was dedicated for the return traffic from the conference venue to the city center (the black line). The red and green bus lines operated only during the morning period, whereas the black and blue lines operated both in the morning and evening. Bus departures were scheduled every 30 minutes for all lines. For the morning, two lines (red and green) covered the demand in the northern part of the city center, picking up visitors along 7 dedicated bus stops and meeting at the farthest point of the line at the Saldahna roundabout. From this point to Web Summit, the visitors were no longer picked up and the bus was navigated solely using the quantum navigation service. For the black and blue lines, the portion of the routes between the Web Summit and Alameda station were navigated using the quantum navigation service.

To determine the schedules of the fleet, we identified the peak traffic demand in the selected zones was from 09:00-10:00 and 10:00-11:00, with 6314 trips towards Web Summit. For the evening demand (the return trips to the city center), peak hours were from 16:00-17:00 and 17:00-18:00 with 7600 trips leaving the Web Summit.

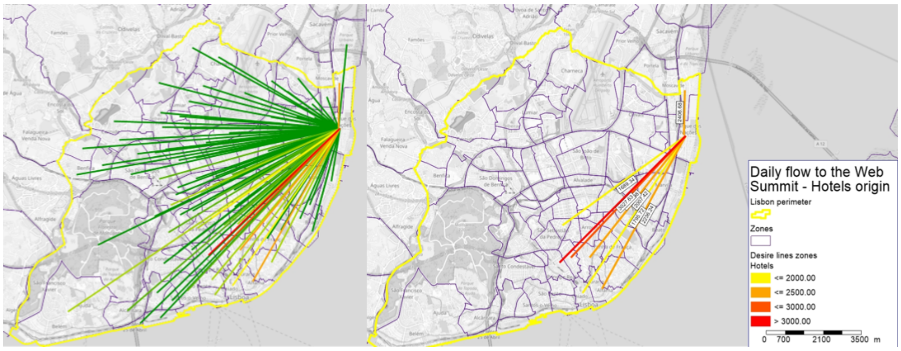


Figure 5.1: Left: Origin-destination matrices from the Web Summit conference to the city center. Right: Selected OD matrices for hotel- and room rental-related trips. Visualization provided by PTV Visum software.

The results also highlighted that the zone with the lowest public transport usage was zone 75 (Santa Maria Maior-Castelo), with 45% of total trips being public transit. This indicated that the modal split in this zone can be heavily improved relative to other zones with higher average public transit usage (65% and above). Given the performed analysis for each of the four selected zones, and considering the estimated demand in both the morning and evening, a total of 9 buses was proposed to Carris as a recommendation for the Quantum Shuttle fleet volume.

5.2 Building a quantum optimization service

Building a functioning web service that uses a quantum processor while providing meaningful navigation optimization imposes a specific set of both conditions and constraints. By setting the goal of navigating buses in real-time, we require that a live connection between three different services be consolidated simultaneously– the navigation app (run on an Android tablet), the traffic data, and the hybrid quantum optimization. We briefly describe the content and scope of each component, then explain how they were combined in the final optimization service, which we call the Quantum Web Service (QWS).

5.2.1 Bus navigation Android app

Development of the Android app was divided into two parts: the front-end (visualization for the bus drivers) and the back-end (server-side communication to the QWS). We explain separately how the two parts were implemented, then consolidated, to satisfy the demands of the navigation service. The front-end development was outsourced to an external supplier, while the back-end optimization (including the API to access it and database management) was developed and implemented internally.

Front-end. The front-end of the bus navigation comprised of an Android application, visualizing the turn-by-turn navigation, operated on an Android tablet. The main role of the app was to plot the custom routes provided by the QWS and initiate turn-by-turn navigation with voice instructions. Additional functionality was built into the app to allow bus drivers to start/stop the current and next trips they were meant to follow, as well as to track the current location of the buses in the fleet to allow for the optimization of the route selection. Trips were also ended automatically once a vehicle was within 15 meters of its defined destination. The visualization portion of the app was built using the Mapbox SDK[1] to render the custom routes obtained by the QWS.

Back-end. The back-end of the navigation app acted as an interface between the Android app and the QWS. The back-end was used to send the collected data of all the vehicles in the fleet at fixed time intervals to the QWS over HTTP. Responses from the QWS were also distributed by the back-end to their respective vehicles.

The most important role of the back-end was to keep the data flow in sync between the vehicles and the QWS, as both synchronous and asynchronous communication protocols were used. This was necessary in order to properly construct correct traffic-flow QUBOs which represented the live navigation conditions. The QWS was designed in such a way that it accepted a single consolidated request consisting of the location information for all the vehicles, both live and projected. It was the role of the back-end software to consolidate all the locations it received from all the vehicle devices at different frequencies, and send it to the QWS. This meant that the back-end maintained an index of each vehicle's request and response. Because of

[1] https://www.mapbox.com/

the different requirements of the live location tracking and the projected locations, the requests submitted by the back-end to the QWS were separated between two destinations on the QWS: /update and /optimize. The /update request submitted the live location of each Quantum Shuttle vehicle to the QWS on a 30 second interval, while the projected location (and in return the customized route returned) was submitted to the /optimize URL at an interval of 120 seconds. The necessity of splitting the requests to separate URLs and their respective timing was discovered during the testing phase, explained in Section 5.3. Lastly, the back-end was also responsible for determining the difference between two subsequent optimized routes for the same vehicle.

5.2.2 Constructing live traffic-flow QUBOs from data

The approach used for custom navigation of the vehicles followed an approach based on [98]. At the start of every trip and at regular time intervals until completion of each trip, multiple candidate routes needed to be generated between the current location of each bus in the system to its assigned destination. These routes also needed to be traffic-aware to reflect the current conditions of the city road network. To accomplish this, we used a live traffic services provider, HERE Technologies [1]. Using their routing API [2], we were able to generate between 3-5 traffic-aware candidate routes per vehicle at every optimization step with minimal overhead. It is important to note that since the vehicles were operated in parallel in different parts of Lisbon, different routes were likely to be suggested for each vehicle in the system at every optimization step. Often, however, subsets of these suggested routes overlapped, necessitating the optimization of the routes' selection to minimize congestion. In this scenario, identical GPS points were returned from the HERE API describing the shape of the overlapping portion of the routes. Therefore, the GPS points were used directly to form the optimization problem, instead of the road segments as in [98]. The QUBO formulation of the objective function is then:

$$\mathrm{Obj} = \sum_{p \in P} \mathrm{cost}(p) + \lambda \sum_{i} \left(\sum_{j} q_{ij} - 1 \right)^2 , \tag{5.1}$$

[1]https://www.here.com/
[2]https://developer.here.com/

where q_{ij} are the binary decision variables associated with vehicle i taking route j, P is the set of all GPS points that overlap in the suggested routes, λ is a scaling factor ensuring only one route is selected per vehicle in the QUBO minimum, and $\mathrm{cost}(p)$ is the cost function associated with each GPS point p in P:

$$\mathrm{cost}(p) = \left(\sum_{q_{ij} \in B(p)} q_{ij} \right)^2 . \tag{5.2}$$

Here, q_{ij} is as before, and $B(p)$ is the set of all binary variables that contain the GPS point p. Thus, the final selection of routes in the optimum of the QUBO represent the routes that minimally overlap with all other selected routes.

5.2.3 Solving the traffic-flow QUBOs with quantum annealing

For live traffic navigation, our quantum optimization service needed to meet specific conditions. Because of the time-sensitive nature of traffic navigation, our solution needed to respond with valid solutions to the optimization problem quickly. The algorithm also needed to handle varying sizes and complexity of the traffic-flow optimization problem, as it needed to optimize the route selections of the vehicles automatically at regular intervals. We used the D-Wave 2000Q QPU and its respective software stack to deploy our solution on quantum hardware. Three different methods of using quantum annealing was tested to solve the traffic flow optimization QUBOs. We briefly explain each method, how it was implemented, and evaluate them based on our navigation application.

Direct embedding. The most straightforward approach to solving the QUBOs is by minor embedding the graph directly to the topology of the QPU. The benefit of this approach is speed– even with the overhead of transforming the traffic-flow problem to a QPU-compatible graph, using the QPU at the fastest annealing time ($1\mu s$ to obtain a single sample) still returned valid solutions to the problem. The minor-embedding process can be performed on the order of tens or hundreds of milliseconds for small-sized problems. However, the drawbacks of this approach have already been explored and presented in detail in Ch. 2. During the development phase of the Web Summit project we tested various configurations of the direct embedding approach, and found that this method was suitable for up to 10 cars with 5 routes each.

Custom hybrid algorithms. One of the primary goals of this project was to assess whether hybrid quantum algorithms could assist in solving QUBOs in practice. The advantage of this approach is that the use of the QPU in the algorithm can be tailored to the QPU's strength, making better use of a limited resource. However, run-time is sacrificed in waiting for the QPU's response to continue the iterative classical procedure. The original traffic-flow work in [98] employed a hybrid algorithm that used a 64 variable fully-connected graph as the inner loop for the QPU to optimize sub-problems [37]. However, as mentioned previously, minor-embedding dense problems significantly degrades the QPUs performance, which in this case still came at the cost of waiting for the QPU responses. Furthermore, new embeddings needed to be generated for each problem being submitted. In light of this, we developed a custom hybrid algorithm to make better use of the QPU in a timely manner. Our algorithm performed the same Tabu search in the outer classical loop as in [37], but instead of using a single 64 variable sub-problem, we found natural sub-graphs within the TFO problem that were already Chimera structured, thus circumventing the embedding issue. We were able to deploy a hybrid algorithm similar to [39] but without employing chains in the sub-problem. Our method allowed us to increase the throughput of sub-problems to the QPU. However, due to time constraints, we were not able to parallelize the implementation so that it could continually run independent of the request/response portion of the QWS. Therefore, the hybrid algorithm was restarted every time a new route optimization was requested. This incurred significant overhead time, delaying the response to an unacceptable level for live use.

Hybrid Solver Service. As part of D-Wave's online cloud service, in addition to direct QPU access, a state-of-the-art hybrid algorithm is also offered. This service, named the Hybrid Solver Service (HSS), is tailored to solve large, arbitrarily structured QUBOs with up to 10,000 variables. The disadvantage of this method is that we cannot control the exact method with which the QPU is used, instead we use the HSS as an optimization black-box. Access to the HSS was provided through the same API as the D-Wave QPU, which allowed us to integrate it in to the QWS in a modular way. By offloading the overhead associated with starting the hybrid algorithm to the D-Wave remote server, we were able to reduce the response time significantly compared to the other approaches. That the HSS can solve problems significantly larger than the QPU also allows us to seamlessly scale up our QWS to handle hundreds of vehicles with tens of route options for future applications. The HSS provided the best compromise of speed, accessibility, and performance, and was used for the live navigation service during the Web Summit conference.

5.2.4 Creating the live navigation service

We now present an overview of the navigation service as a whole, detailing the various components comprising of the system and how they communicate, for review and clarity. A diagram of the system is presented in Figure 5.2. The components are color-coded according to their roles in the service: blue components relate to data-processing steps, which can be considered as input to the hybrid optimization routine. Gray components comprise the core of the QUBO construction and optimization to solve the traffic-flow problem. The white components are black-boxes provided by external parties.

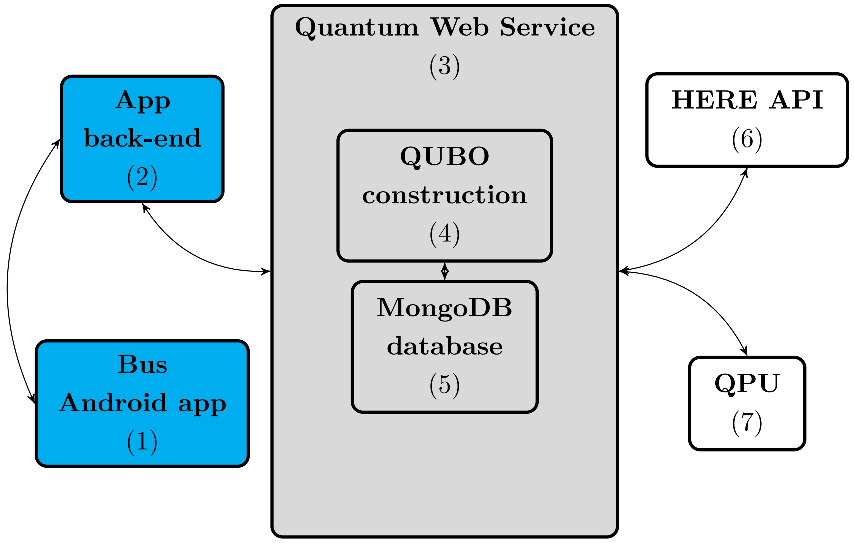


Figure 5.2: Diagram detailing the QWS and its interactions with the other components in the hybrid navigation system, as per the text.

Blue components.

- Component (1), the front-end Android tablet application, provided visual turn-by-turn navigation with vocal instructions to the Quantum Shuttle bus fleet and their drivers. The bus locations were sent every 30 seconds to Component (2).

- Component (2) was the Android app back-end, and submitted the POST requests to the QWS, Component (3). Component (2) consolidated the locations of the vehicles and submitted them to (3) via the /update URL every 30 seconds; this component also formed the POST request to the Component (3) /optimize URL for the route optimization, interpreted the response, and sent the new routes back to (1).

Gray components.

- Component (3) is the framework hosted on AWS (exposed using Flask[1]) and served as the central consolidation point of the other components. In the event of an /update POST request from (2), the QWS updated Component (5), the MongoDB database used to store the data. In the event of an /optimize POST request from Component (2), Component (6) was accessed to request the suggest routes, and that data was passed to Component (4) to construct the traffic flow optimization QUBO problem, then stored in the database using (5). The QUBO was submitted to the D-Wave HSS, Component (7). Component (3) then interpreted the results of the optimization, stored them via (5), and pushed the selected routes back to (2). The API used to communicate with the QWS was custom-built for this application.
- Component (4) is the Python module that implemented the traffic flow optimization QUBO formulation described in Section 5.2.2.
- Component (5) is the Python module wrapped around the MongoDB database; accessing and writing data from the QWS to the database was performed by this component.

White components.

- Component (6) is the HERE Technologies traffic/routing API.
- Component (7) is the D-Wave HSS (or other QPU-based services), accessible via HTTPS.

[1]Flask is a minimalist Python framework for making web-apps, and can be found at: https://www.palletsprojects.com/p/flask/.

5.3 Deploying a quantum optimization service

In preparation for the Web Summit conference, multiple trial runs were performed in both Wolfsburg, Germany (Volkswagen AG headquarters) and Lisbon, Portugal. In this section we describe the two different test scenarios and the lessons learned from each.

5.3.1 Initial tests: Wolfsburg

The first testing phase occurred in Wolfsburg in August 2019, where a small number of cars (1-3) were driven between various origin/destination pairs using the Android app. The goal of this testing phase was to ensure the proper integration of the Android navigation app, QWS API, and the MongoDB database used to keep track of active trips and record results. During testing, several key observations were made, which were used to modify the system.

Location tracking. As mentioned in Section 5.2, it was necessary to both track the live positions of the vehicles, as well as specify a new "origin" per vehicle for every optimization request. In order to ensure that the locations used for optimization didn't result in infeasible or unrealistic route selections, a projected location was used for each vehicle. Specifically, given a route in the form of a sequence of GPS coordinates, a GPS point further along along the route was passed as the origin for the next optimization step. During the initial testing phase, this position was also used to track the locations of the vehicles. However, this proved to be problematic, since the update frequency was faster than the change in the projected location, making it impossible to track the locations of the vehicles. To solve this, the live location and the location used to determine the new routes were separated: location updates were sent to a different URL independently from the optimization requests every 30 seconds, with the live locations now stored server-side after every update.

Optimization interval. Before testing, the route optimization occurred every 60 seconds. During testing it was observed that this frequency was too fast, resulting in different routes being assigned to the vehicles every time an optimization problem was solved. Furthermore it was observed that, given a new route A after the optimization, a *different new route* B would sometimes be suggested while navigating to route A. This would have made navigating buses with real passengers impractical. It was discovered that this phenomenon occurred due to two main reasons: firstly, quantum annealing and other such hybrid algorithms are metaheuristic optimization algorithms, meaning they are not guaranteed to return the same solution every time they are queried. In the event that multiple minimal solutions to the optimization problem have equal cost, the QPU may return any of the solutions [75]. Secondly, because of the frequent optimization requests, the vehicles' projected positions often stayed the same between optimization requests, causing the same QUBO problem to be formulated in successive requests, which is not particularly useful. After testing various optimization intervals (both faster and slower), 120 seconds between optimization requests resolved the issue, allowing for sufficient time to change the projected locations of the vehicles.

5.3.2 Final tests: Lisbon

The second testing phase occurred during September 2019, with a small number of buses and drivers in Lisbon. The final bus route start and end points as selected in Section 5.1 were tested using the Android navigation app and the quantum web service, including the changes implemented after the initial testing phase in Wolfsburg. The significantly different conditions in Lisbon allowed us to further tune our navigation system, explained below.

Street exclusions. The road network of Lisbon is significantly different from that of Wolfsburg. Apart from the major roads and highways in Lisbon, many of the local streets are narrow and one-way, making them not well suited for public transit. Additionally, many roads have steep inclines, which are difficult for buses to climb. Neither of these two conditions were present in Wolfsburg, which lead to multiple unacceptable scenarios when testing the buses in Lisbon. More than once, routes were suggested that utilized these small streets through which the buses could not fit, causing them to either turn back (adding delay to the travel time), or even worse, forcing them to stop completely. For example, between Saldanha roundabout and Alameda station there is a network of highly connected one-way streets. Suggested routes using these roads could be useful for cars, but were completely undesirable for the buses in the Quantum Shuttle fleet. Similarly, a network of narrow one-way streets exists near the Web Summit conference center, which also needed to be avoided. To accomplish this, whenever routes were suggested that utilized roads in these networks, the GPS locations were recorded, added to a list stored on the QWS server. This list (in the form of bounding boxes of areas to avoid on a map) was submitted as part of the HERE routing API request, which returned only routes that avoided those areas. This list of forbidden areas was actively updated throughout the successive tests in Lisbon based on feedback from the bus drivers, resulting in only valid routes being generated by the end of the final testing phase. This list of excluded regions was then used for the live run during the Web Summit.

Time filtering. The number of candidate routes that can be requested from the HERE routing API per vehicle is an unconstrained parameter client-side. By default, 3 candidate routes per vehicle were requested, to keep consistent with the work in [98]. However, it was observed that in some cases one or more of the routes suggested were significantly slower than the fastest route suggested, resulting in extremely slow routes sometimes being suggested. The reason for these slow routes being selected was due to the way the cost functions are formulated in the QUBO problem. The goal is to reduce the amount of congestion caused by the vehicles, defined by the number of streets/GPS points shared between candidate routes across all vehicles. The slower routes suggested by the HERE routing API were often significantly longer than the fastest suggested route, and thus had lower overlap with the faster routes, causing some of the vehicles to be assigned to the slower routes. To circumvent this, a time filter was implemented to assure only reasonably fast routes were considered as valid candidates. After testing various values for the time filter, a value of 2 minutes provided the best trade-off between the number of routes selected and the routes' relative expected travel times. By allowing the slowest suggested route to be *at most* two minutes slower than the fastest route suggested, we were able to maintain three valid candidates per vehicle for the majority of the trips.

5.3.3 Web Summit 2019: Live run

For the launch of the event at the Web Summit conference, this project was presented under the name "Quantum Shuttle". The service was active from November 4-7, 2019, and was operational for public use from November 5, 8:00 in the morning Lisbon time, to 18:00 in the evening on November 7, 2019. A total of 185 trips were recorded during the 4 day period with a total fleet size of 9 buses.

However, a small number of trips were erroneously recorded, due to manual driver cancellation or restart of the trip. Such trips were identified in two ways: either a small number of vehicle locations recorded in the database (fewer than 5), or one of the origin/destination points being far from the expected location (more than 1 km). Of the 185 trips, 162 (87.6%) were valid trips corresponding to the expected "Quantum Shuttle" service. The exact counts per day and line of the service are presented in Table 5.1. As per Section 5.1, two of the lines operated from the city center of Lisbon to the Web Summit: Alameda station to Web Summit (blue line) and Saldanha roundabout to the Web Summit (red line[1]). The third line (black) ran from the Web Summit to Alameda station.

	Black line	**Blue line**	**Red line**
Conference total	53	56	53
8:00-12:00	17	21	47
12:00-18:00	36	35	6

Table 5.1: The 162 trips taken by Carris buses operating the Quantum Shuttle, separated by time of day and line.

All three lines had roughly equal number of trips throughout the conference. The route frequency matches the expected demand in Lisbon– the red line from Saldanha was the popular choice during the morning, whereas the blue line from Alameda was used more in the afternoon. Likewise, the only line from the Web Summit back to Lisbon city center (black line) had double the trip frequency in the afternoon compared to the morning, again matching the demand of conference attendees returning to their accommodations after the conference ended each day. The duration of each trip was recorded together with the location history of each vehicle in the fleet throughout the conference. The corresponding average trip times are shown in Table 5.2. The trip times are recorded from the moment a driver presses the start button, until either the trip is manually ended or the bus is within 50 meters of its destination.

One of the key design goals in our traffic navigation system was making sure it could operate continually without manual intervention. As a consequence, there was significant variation in the complexity of the optimization problems being solved throughout the conference, depending on the number of active vehicles in

[1]Since the quantum navigation of the green and red lines have identical origin and destination, we combine them and refer to them together as the red line.

	Black line	**Blue line**	**Red line**
Conference total	23 min 41 s	23 min 18 s	26 min 34 s
8:00-12:00	25 min 36 s	22 min 43 s	27 min 36 s
12:00-18:00	22 min 46 s	23 min 38 s	18 min 19 s

Table 5.2: Average trip times for the Quantum Shuttle, separated by time of day and line.

the fleet. A total of 1275 optimization problems were solved by the QWS for the 162 trips, with an average response time of 4.69 seconds. Of those, 728 problems (57.1%) involved more than one route per vehicle in the system, with an average response time of 6.78 seconds. We consider the optimization problems for which there is more than one route per vehicle as the "harder" version of the traffic flow problem, since otherwise the route selection is trivial. It is important to note that the ability to navigate the fleet buses strongly depended on creating and solving the optimization problems in a timely manner. Since we cannot anticipate in advance whether the vehicles have one or more possible routes (depending on the traffic conditions), our system needed to operate uninterrupted in all cases. Additionally, while there was a fallback mechanism in place stored locally on each device, 100% of the calls to the D-Wave HSS completed successfully, thus maintaining our automated navigation system's integrity for the duration of the conference, regardless of the complexity of the problem being solved[1].

The goal of this project was to evaluate the use of solving the QUBO formulation of the traffic-flow optimization problem in a live setting to perform turn-by-turn navigation. Therefore, we now focus on interpreting the data and results of the project in this application context. The distribution of all the Quantum Shuttle trip data is shown on a map of Lisbon in Fig. 5.3. The trips are colored based on the lines to which they belong, as described above. The red, blue, and black circles correspond the the origins of their respective lines. The black circle is the Web Summit conference location, and is therefore also the destination for the red and blue lines. It is important to note that none of the three lines used the same route for all trips throughout the Web Summit, showing that our QWS navigation system provided flexible traffic-aware routing. The three highways that connect between

[1]The largest QUBO that was solved consisted of 12 variables, with 5 buses being navigated concurrently. This occurred on November 5, 9:11 Lisbon time, which was the busiest period during the conference.

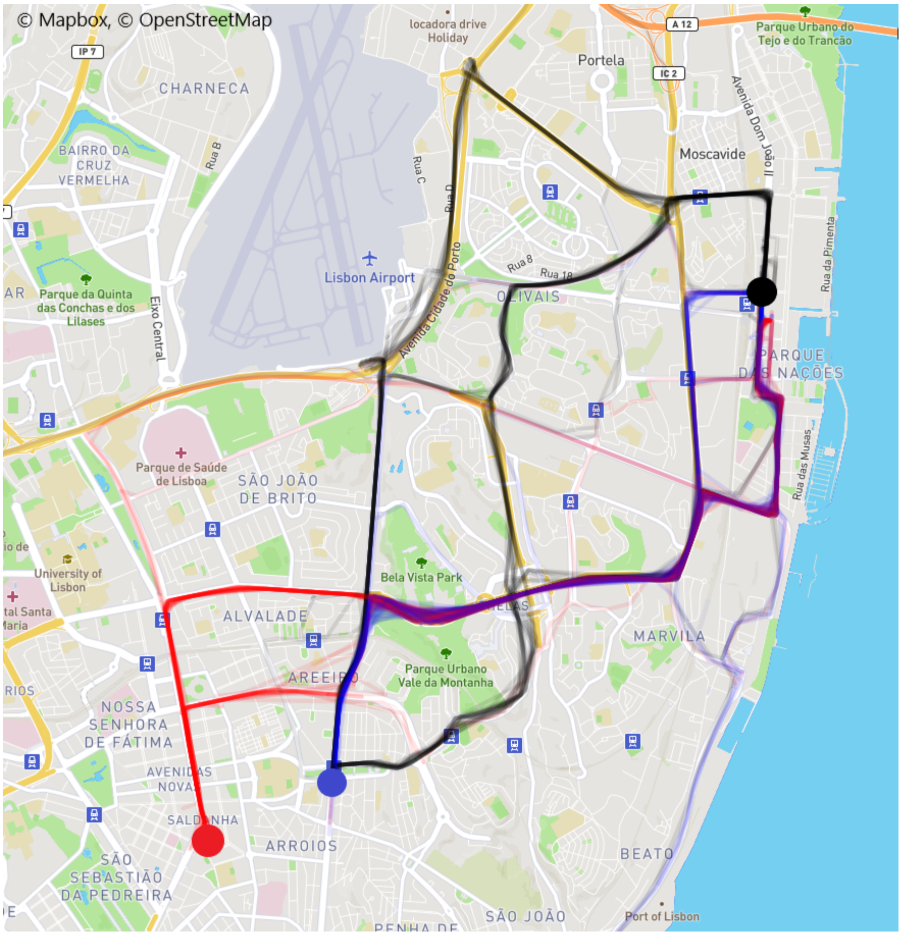


Figure 5.3: Distribution of all recorded Quantum Shuttle trips. The trips are color-coded based on the line they correspond to. The circles represent the origins and destinations of the respective lines [124].

the city center and the conference center were used extensively (although not exclusively), and at different times by the different lines. That these highways were prominent in the route selections is attributed to two design choices: time filtering and excluded streets. Highways are typically the fastest method of driving medium- and long-range distances, making them likely candidates for selection. Furthermore, the regions that were excluded from the route selection as per Section 5.3.2 removed fast route suggestions that avoided highways. It is reasonable to assume that in the case of navigating cars, as opposed to buses, Figure 5.3 would show increased distribution over the smaller city streets as well.

To quantify the customization of the routes used by the vehicles, we measure the dissimilarity between them. Specifically, we measure the overlap between the location histories of vehicles being navigated concurrently by our system. The overlap is defined as the fraction of GPS points in a vehicle's location history that coincided with another vehicle's route (that was being navigated at the same time), within a distance of 50 meters[1]. The overlap metric is therefore defined to lie between [0, 1]. Distance (d) between GPS points was calculated using the haversine formula:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right), \tag{5.3}$$

where ϕ_1, ϕ_2 are latitudes, λ_1, λ_2 are longitudes, and $r = 6,371,009$ meters is the Earth's radius.

Because the navigation system re-optimized the distribution of routes at most every 120 seconds, each vehicle's recorded trip is a sum of successive routes suggested by the optimization. It is important to note that the suggested routes, as obtained via the HERE API, are traffic-aware, and therefore already circumvent the existing traffic congestion in the city. Thus, by minimizing the overlap between the fleet vehicles, we minimized the additional traffic congestion caused by our fleet. In Figure 5.4 we show the overlap between pairs of buses, grouped by the lines the buses are following. There are six possible ways to compare the buses this way: three within the same line (red-red, blue-blue, black-black), and three between the lines (red-blue, red-black, and blue-black).

To contextualize the results in Figure 5.4, we also calculate the overlap between the most direct (i.e., fastest) routes suggested by the HERE API for each line without the traffic-aware component. This therefore simulates our navigation system in an "offline" mode– public buses typically have pre-defined routes that are not deviated from even in the presence of traffic congestion. Using these static routes, we obtain the following overlaps: 0.70 for red-blue, 0.01 for red-black, and 0.16 for blue-black (intra-line overlaps are trivially 1, since the same routes would be used for every vehicle in that line). The red-black and blue-black overlaps are similar in offline

[1]This definition differs from the one in Section 5.2.2, since the recorded location histories were irregular, as opposed to the points used to construct the traffic-flow optimization problem. The location histories were interpolated using the HERE API for consistency.

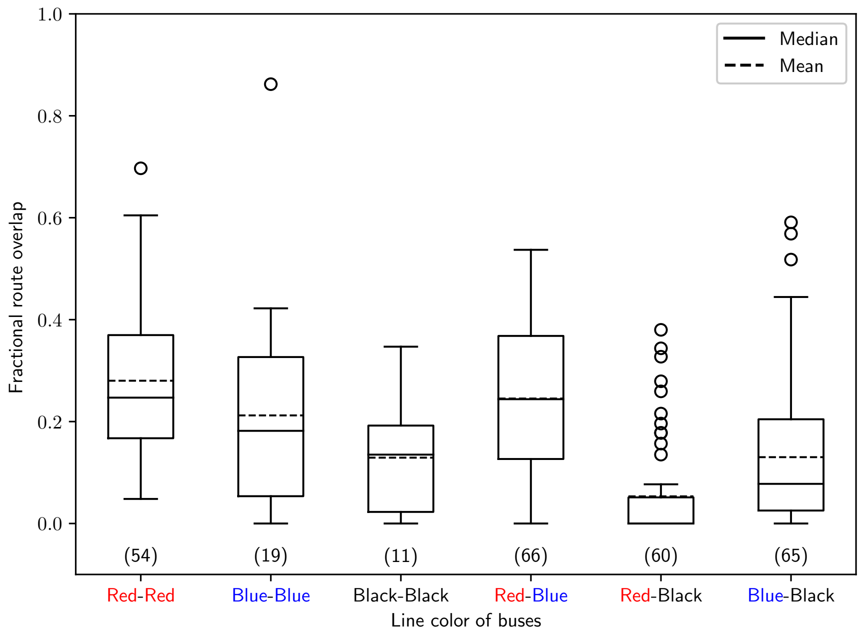


Figure 5.4: Box plot showing the fractional overlap of routes being navigated simultaneously in the Quantum Shuttle fleet. Boxes are grouped by the line colors. The number of observations are in parentheses below the boxes.

and online mode, however the expected time for the offline mode would be higher due to the lack of traffic-aware routing. For the red-blue overlap (as well as the intra-route overlaps), our online method significantly reduces the overlap between the routes. It is also worth noting that in all categories in Figure 5.4, the mean and median overlaps were below 0.5, meaning that the majority of every route was different from every other vehicle in the fleet. By using the QWS we were therefore able to both circumvent existing congestion as well as avoid creating new congestion.

From a technological perspective, the Quantum Web Service's modular implementation allowed us to communicate with a live QPU in a timely fashion, making it suitable for our traffic optimization use-case. The test runs in Wolfsburg and Lisbon were particularly instrumental, allowing us to fine-tune the connections between the components of the QWS given the constraints of the application. Due to this, the final implementation of the Quantum Web Service can handle live optimization of other, non-traffic related processes provided modifications to the QUBO construction and live data components. Many production processes have similar constraints to the those present in the Quantum Shuttle, making our work adaptable to other scenarios. However, we stress that the problem sizes investigated here were small (no more than 30 variables in all cases), well within the range of problem sizes addressable by classical algorithms. Therefore, while the application ran adequately in a live environment for a demonstration, further research is necessary to fully understand how well the problem can be solved by quantum annealing at scale.

5.4 Motivating a better real-world optimization use-case

We now explore another real-world optimization problem from the automotive industry. The *paint shop problem* refers to a set of combinatorial optimization problems where the objective is to color a (fixed) sequence of cars with a fixed number of colors such that the total number of color switches is minimized, given a set of customer orders. This simple problem poses interesting scientific questions, which in turn have real impact for solving such problems in practice. The paint shop problem was originally posed by Epping et. al [125] as a form of coloring problem. Some clarification on nomenclature: the given car sequence can also be referred to as a word, where each car is denoted by a character. In [125], it was shown that the paint shop problem is NP-complete in both the number of colors and cars in the sequence. Furthermore, results show that, for bounded numbers of colors and unique cars, there exists a polynomial-time dynamic programming solution to these instances. Subsequent work [126] extended these results, proving that even the simplest coloring version, with only two colors, is both NP-complete and APX-hard. Additional results show that a subset of problems meeting specific conditions can be solved in polynomial time. Therefore, this class of problems are good candidates for quantum optimization algorithms, and initial such studies have been conducted on a restricted definition of the problem [127] (different sub-classes are distinguished shortly).

5.4.1 The paint shop optimization problem

One of the steps in car production at Volkswagen is painting the car body before assembly. In general, this can be viewed as a queue of car bodies that enter the paint shop, undergo the painting procedure, and exit the paint shop. It is important to note that the area of the factory immediately proceeding the paint shop is typically assembly, where car components are assembled into the car bodies. Because of the many different models and configurations being produced, designing a sequence of cars to be assembled that is optimal (also known as the car sequencing problem) is a known NP-hard problem in itself [128]. In practice it is imperative to solve the sequencing problem because of worker safety and regulatory issues, and we therefore treat the sequence of cars entering the paint shop as a fixed queue. However, the *colors* assigned to the cars within a given sequence are still randomly distributed, and thus we can focus on optimizing them. An example with three different car groups is shown in Fig. 5.5 (top).

Each car body entering the paint shop is painted independently in two steps: the first layer is called the filler, which covers the car body with an initial coat of paint, and the final color layer is the base coat, which is painted on top of the filler. The base coat is the color that matches the final customer order: blue, green, etc. However, the filler has only two possible colors: white for the lighter base coats colors, and black for darker colors. We define a customer order as the number of cars of each configuration to be painted one of the color choices. We associate each coating step (filler or base coat) with a unique class of paint shop problems, each of which are NP-complete [125, 126]. A simple version of the filler optimization is when there are only two cars of each unique configuration in the sequence and each needs to be painted a different color; this is referred to as the *binary paint shop problem* (BPSP) [127]. More generally, the filler optimization is referred to as the *multi-car paint shop* problem, where the cardinality of each set of unique cars in the sequence is unconstrained, but the cardinality of the color set is restricted to two (e.g., black or white). The base coat optimization is therefore an extension of the MCPS problem, where the cardinality of the color set is also unconstrained– we call this version of the problem the *multi-car multi-color paint shop* problem. In our work we focus on the filler optimization, the MCPS problem, which can be formulated natively as a binary optimization problem. We formally define the problem as follows:

Given:	a word w defining the fixed sequence of N cars (w_i denotes the ith character in w),
	set of $C = \{C_1, \ldots, C_M\}$ unique configuration groups,
	binary choice of colors $\{W, B\}$,
	function $k(C_i)$ which defines the number of w_i to be painted B in w,
	function $f(w)$ to count the number of color switches in w,
such that:	$\#w_i\|_B = k(C_i),\ \forall C_i \in C$,
minimize:	$f(w)$.

In practice (i.e., in the real paint shop), the information required to formulate this optimization problem is always available, as it is a necessary part of fulfilling customer orders. Therefore, this MCPS problem representation above corresponds exactly to the industrial use-case of paint shop optimization on the filler line. This provides a more tangible use-case to solve using the hybrid quantum optimization methods developed in the previous sections. In Fig. 5.5 we show a simple example of the MCPS problem with three car groups.

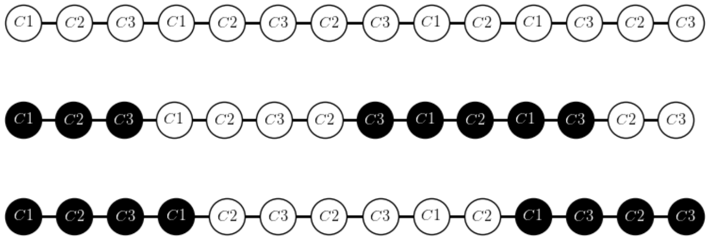


Figure 5.5: Simple example of a multi-car paint shop problem with three car groups (C_1, C_2, C_3). The three corresponding orders are $k(C_1) = 3, k(C_2) = 2$, and $k(C_3) = 3$. **Top:** The fixed sequence of cars in the paint shop queue. **Middle:** Sub-optimal solution to the problem with 3 color switches. **Bottom:** Optimal solution to the problem with 2 color switches.

Despite the focus of our work on the filler, we briefly address the optimization of the base coat. Although the filler and base coat are painted independently in separate locations, computationally the two problems are not separable. Abstractly, optimizing the base coat line (i.e., solving the multi-car multi-color paint shop problem) is a straightforward generalization of the MCPS problem: we can extend the binary color variables to discrete color variables, where $k(C_i)$ denotes the number of times each color appears in a car group. In practice, it is useful to consider solving the multi-color problem *after* solving the MCPS problem. Due to the aforementioned one-to-one mapping between base coat and filler color, it is still possible to permute the order of base coat colors within a contiguous sequence of filler colors to reduce the base coat color switches. Although discrete optimization problems can be represented as binary optimization problems, we do not solve the multi-color version of the problem and leave this work for future studies.

5.4.2 Ising model representation of MCPS

In principle, the formulation of the MCPS as an Ising model is straightforward: we start by representing every car in the sequence w (w_i) with a single spin variable (s_i). The spin up state denotes if the car is painted black, and the spin down state denotes if it is white. The Ising model which represents our problem can be divided into a hard constraint component and an optimization component. The optimization component is a simple Ising ferromagnet with $J = -1$ couplings between adjacent cars in the sequence:

$$H_A = -\sum_{i=0}^{N-2} s_i s_{i+1}. \tag{5.4}$$

This incentivizes adjacent cars to have the same color. The second component of our Ising model is the hard constraint, ensuring that the correct number of cars are colored white/black per customer orders. This is encoded in a second energy function, as a sum over independent k-hot constraint for each group of cars C_i:

$$H_B(C_i) = (\#C_i - 2k(C_i)) \sum_i s_i + \sum_{i<j} s_i s_j. \tag{5.5}$$

Therefore, the final Ising model is the sum of the two components:

$$\begin{aligned} H_{\text{MCPS}} = &- \sum_{i=0}^{N-2} s_i s_{i+1} \\ &+ \lambda \sum_{C_i \in C} \left[(\#C_i - 2k(C_i)) \sum_i s_i + \sum_{i<j} s_i s_j \right], \end{aligned} \quad (5.6)$$

with terms as previously defined in the MCPS problem statement. In order to ensure only valid configurations of spins are encoded in the ground state, it is necessary to scale H_B by the factor λ. The value of λ is chosen to be large enough such that it is never energetically favorable to violate a constraint to reduce the energy of the system. In our study we set $\lambda = N$, the total number of cars in our sequence, which guarantees this condition.

5.5 Creating Ising models from paint shop data

5.5.1 Data sources

The MCPS problem instances we used were generated from real data taken from a Volkswagen paint shop in Wolfsburg, Germany. The reason for this is two-fold: firstly, the main goal of this work is to test the viability of quantum annealing methods in solving industrial optimization problems. It is our goal to accurately capture the complexity of the industrial use-case without relying on simplifications or randomly-generated problem instances. Secondly, the paint shop currently operates on a first-come-first-served basis, where customer orders are entered into the queue as soon as they arrive. This guarantees a certain amount of randomness (although not uniformity) in the problem instances we solve: different car models, configurations, and base coat colors appear throughout the sequences in ways we do not control. Therefore, these conditions are suitable for our analysis.

A total of 104,334 cars were used in this study. To reproduce real-world conditions as faithfully as possible, the car sequences used are multiple independent sets of car sequences, each representing one week of continuous production, which are stitched together as one continuous block. The data is collected over a period of one year, roughly once every six weeks, to avoid seasonal biases in customer order preferences. There are 121 unique car configurations in the data set. Of those, 13 of the configurations appeared only once in the data set, and therefore do not need optimization at all. Typically this indicates that either a custom configuration was built that cannot normally be ordered, or a prototype assembled for testing purposes. Rather than exclude these from the study, we include them in the optimization, because fixing a color at one location in the sequence influences the adjacent cars (and consequently the total number of color switches) in a non-trivial way. We do, however, eliminate the spin variable from the Ising model by conditioning on the color.

5.5.2 MCPS problem sizes

To generate a variety of input sizes from the full data set, we partition the data into different sized sequences without permuting the car order. This is motivated by the amount of cars that need to be optimized for different purposes. For example, the paint shop used as a basis for this analysis has a queue capacity of roughly 300 cars. This is not the *total* capacity of the paint shop, but rather the maximum number of cars that can be physically inside the paint shop queue before they are painted. We consider this a rough lower bound on the problem size for industrially-relevant problem instances. An upper bound is more difficult to establish. From a theoretical point of view, there is merit in investigating the behavior of quantum systems in the infinite size limit, as in [127]. From an industrial perspective, car orders can be placed weeks to months in advance, which would yield problem sizes of $10^3 - 10^5$ variables. In reality, real-time and last-minute adjustments (due to manufacturing problems, supply chain issues, or imperfections in painting) can happen on a daily basis. We limit the analysis to problems of up to 3000 variables, which roughly corresponds to a few days worth of production. The data partitioning is performed by dividing the entire data set into equal chunks for each problem size N. Each partition is considered a candidate instance, yielding a total of $\lfloor \frac{104{,}344}{N_{\text{cars}}} \rfloor$ partitions per problem size. We then mine these partitions and select suitable partitions to generate problem instances from. This is due to the fact that there could be little frustration within some partitions. For example, it is possible that all cars of any one configuration (C_i) *all* need to be painted either black or white. While in general this is an accurate reflection of production, for experimental analyses this scenario is not useful. Therefore we deem a data partition to be a usable MCPS instance if the total number of non-fixed cars is at least 70% of the cars in the partition. Meaning that, in a 10-car data partition, at least 7 of the cars must have the freedom of being painted either color. We show the total number of partitions for the various problem sizes and how many partitions were valid problem instances in Table 5.3. For our experiments we randomly selected 50 valid instances to test at each N, except for the largest problem size of which we use all 34 valid instances.

Table 5.3: Problem sizes and number of problem instances generated from the data set partitions.

Problem size (cars)	**Num. partitions**	**Num. instances (% of partitions)**
10	10,433	172 (1.6%)
30	3,477	418 (12.0%)
100	1,043	756 (72.5%)
300	347	341 (98.3%)
1000	104	102 (98.1%)
3000	34	34 (100%)

5.5.3 Classical, quantum, and hybrid solvers

To evaluate the efficacy of quantum (and hybrid) algorithms in both the small-scale and industrially-relevant limit, we validate our methods using multiple algorithms. The goal of this analysis is to provide a fair but thorough comparison of results across different regimes of the MCPS problem, from small toy problems to large-scale instances, and represent real-world optimization conditions as closely as possible.

Random. Without optimization, we consider any assignment of colors to cars in a sequence where the orders are fulfilled to be a valid, but not necessarily optimal, solution to the problem. Thus, we can trivially generate random sets of valid solutions by uniformly assigning the color black to $k(C_i)$ cars for each car group C_i. While far from optimal, this solution to the MCPS problem may be preferable if other steps in the car manufacturing process are valued over the painting step. This was indeed the case for the data obtained from the paint shop in Wolfsburg, and thus random valid solutions serves as the baseline the competition algorithms are tasked with beating. For our study, we generate $2N_{\text{cars}}$ random valid solutions at each problem size to estimate the number of color switches that would occur naturally in the paint shop.

Black-first. This is the simplest algorithm that is used to solve the MCPS problem. Starting at the beginning of the sequence, we greedily assign the color black to every car until a white color *must* be assigned to the next car. In greedily obtained solutions the number of color switches grows linearly with the number of cars and is sub-optimal except for a minority of cases [126, 127]. Nonetheless, it serves as a good benchmark for more sophisticated optimization algorithms, as the improvement over random assignments grows with the problem size.
Simulated annealing. The simulated annealing metaheuristic used here is the same as those in the previous analyses in this thesis [99].
Tabu search. The Tabu metaheuristic here is also the same as in the previous chapters, and the same implementation is used [99].
D-Wave 2000Q. One of two different D-Wave QPUs used in this study, this QPU was the older-generation Chimera architecture with maximum degree 6. The processor used in this study had 2041 functional qubits.
D-Wave Advantage. The newest-generation QPU provided by D-Wave with a different topology and a significantly higher qubit count than its predecessor. The new topology, Pegasus, had a maximum degree of 15, and the QPU used in our study contained 5436 functional qubits. For further information regarding the D-Wave QPU topologies and the differences between them, we refer the reader to [64].
D-Wave Hybrid Solver. At large instance sizes (300 cars and higher) it was no longer possible to embed problems directly onto both D-Wave QPUs. Therefore we employed the hybrid quantum-classical algorithm used previously, the Hybrid Solver Service (HSS). As before, the QPU it uses cannot be programmed fully directly by the user, and thus we treat this algorithm as an optimization black-box with a single timeout parameter.

5.6 Benchmarking solvers in the industrial limit

We interpret our results relative to two different regimes: small-scale (10-100 cars) and industrial (300-3000 cars). Each solver used in these experiments was tuned in good-faith, but not necessarily optimally. Meaning, considerable effort was made to ensure solvers were being used to their strengths, but fully optimizing over all sets of hyperparameters for the solvers was deemed out of scope. We compared all solvers' performance in terms of their "improvement" over the random solver, defined as the difference in $f(w)$ between the best solution obtained by each solver and the random solver. This metric is representative of the real-world expected improvement using each of the solvers. In Table. 5.4 we report the median for each solver. We quote the median to be less susceptible to tails of the distribution. The results therefore represent the typical MCPS case at each problem size, rather than the expected value of each solver's overlap with the ground states. The solutions obtained from all solvers were post-processed (if needed) to ensure that the $k(C_i)$ constraint was satisfied in each problem. We show how often this occurred in Table 5.4 as well. We consider this necessary in order to interpret the solutions relative to the application, since it is trivial to reduce the number of color switches by ignoring the customer order constraint. In Fig. 5.6 we highlight the empirical scaling of our results in the industrial limit.

Solving problems directly with both QPUs was only possible for problem sizes 10-100. Embeddings were generated using the standard D-Wave embedding tool Python package, the same that used in previous experiments in this thesis [68]. The chain strengths required by each QPU was different depending on the length of the chains in the embeddings. We calculated the algebraic chain strength $\text{chain}_i = |h_i| + \sum_{j \in \text{adj}(i)} |J_{ij}|$ for every spin in the Ising model, and introduce an additional scaling parameter $s = [0.1, 0.2, \ldots, 1]$. Thus, the chain strength is defined as $s \cdot \max(\text{chain}_i)$, where s was optimized per problem size using a subset of the instances (10 per size), and $50 \cdot N$ samples per instance. Optimal s was defined as the value which yielded the highest frequency of valid solutions relative to the constraints of the MCPS problem in Eq. 5.6 (not chain breaks). We found that the D-Wave Advantage QPU had optimal $s = 0.3$, whereas the D-Wave 2000Q QPU had optimal $s = 0.45$. This is consistent with the fact that the Advantage QPU required shorter chains to embed the same Ising models as the 2000Q. Each QPU was then sampled for $500 \cdot N$ samples per problem size N, with annealing time $t_a = 1\mu s$. We used N spin-reversal transforms for each sample set, as it has been shown that there are diminishing returns between 100-1000 samples per transform [75].

We found that for the 10 car instances both QPUs matched the consensus best results between all solvers (median of $f(w) = 2$), and for the 30 car instances very near the best results ($f(w) = 5$ as opposed to SA's and HSS's 4). For the 100 car instances, with 50,000 samples per problem, the 2000Q QPU found valid solutions for 22/50 instances, as opposed to 37/50 for the Advantage QPU. From this we conclude that 50,000 samples is insufficient for the QPUs, but due to limited time availability we could not take more samples. We note that it was also possible to embed 47/50 of the 300 car instances onto the Advantage QPU. However, due to the poor performance on smaller sizes with limited resources, we did not evaluate those problems. Furthermore, due to the limited problem sizes that could be solved with the QPUs and the quality of results, we do not include these results in Fig. 5.6.

The Tabu solver was given $\lfloor N/3 \rfloor$ seconds per problem as its timeout parameter, and all other parameters were set to their default value. This solver struggled to find valid solutions past the 10 car instances. Due to the post-processing technique, which greedily corrected each sample to satisfy order $k(C_i)$, the Tabu results were essentially a worse version of the greedy algorithm at all problem sizes but the smallest.

Simulated annealing (SA) has many tunable hyperparameters: number of sweeps (N_{sweeps}), number of samples (N_{samples}), and (inverse) temperature (β) schedule. In our experiments we fixed the schedule to $\beta = [0.01, 10]$, interpolated geometrically using N_{sweeps}. We set $N_{\text{sweeps}} = 10 \cdot N$ and $N_{\text{samples}} = 20 \cdot N$, where N is the number of cars. The solutions obtained by SA (shown in Fig. 5.6) were consistently better than the greedy algorithm. Using the given parameters SA was able to provide valid solutions for 50/50, 49/50, and 44/50 for the 30, 50, and 100 car instances. However, the timescales necessary to obtain results were prohibitive from extending the experiments: 300 variable problems were terminated after running for 24 hours without returning a solution. We include the SA results in Fig. 5.6 due to their high quality at small sizes. Using a single-threaded SA implementation, run-time of the algorithm was on the order of seconds to minutes for the 10 and 30 car instances, and between 1-3 hours for the 100 car instances. We note that SA was the only solver which was allotted quadratically scaling computing resources: both sweeps and samples scaled with N. This is necessary for SA to be competitive, and exemplifies the trade-off between results quality and algorithmic run-time when using heuristics.

The D-Wave HSS was given equal time to the Tabu solver, given its only parameter is the timeout: $\lfloor N/3 \rfloor$ seconds per problem. The HSS was the only solver to consistently provide better solutions than the greedy algorithm for all problem sizes. The improvement continued to grow with increasing problem size, shown in Fig. 5.6. However, the gap between the HSS and the greedy algorithm shrank with increasing problem size, performing only slightly better than greedy for the 3000 car instances.

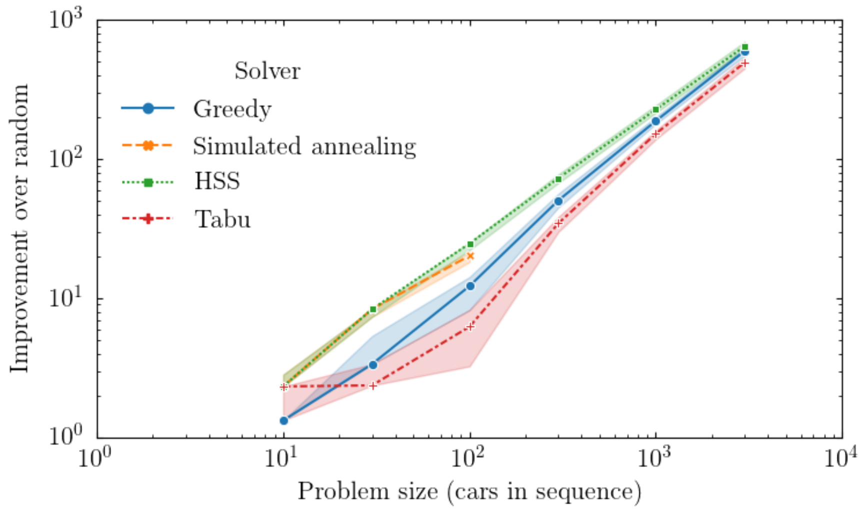


Figure 5.6: Number of color switches within the sequence shown as improvement over random configurations of orders.

Despite the simplicity of the Ising model representation of the MCPS problem, almost all solvers exhibited difficulty in finding valid solutions. This is particularly evident in the performance of the two different models of QPUs tested. The results degrading rapidly from 30 to 100 cars indicate that the problem became more difficult to solve disproportionately to the increase in system size. The SA and Tabu solvers exhibited similar trends, despite the increase in resources allotted to them. While the HSS was the best-performing algorithm, it also missed valid solutions for some problems of intermediate size (100 and 300 car instances). We identify two possible issues: first is the connectivity of the problem graph. Each order $k(C_i)$ requires a separate k-hot constraint which is represented using a fully-connected graph. This yields sub-cliques within each problem that increase with the problem size. In QPUs, denser problems create longer chains and higher chain strengths. For classical solvers, these sub-cliques create rugged landscapes which make single-flip optimization algorithms significantly less useful. Therefore, that the maximum sub-clique of the MCPS problem graph increases as a function of the number of cars is a bottleneck for performance. Secondly, the normalization terms necessary to encode the MCPS problem as an Ising model also scale with problem size. In Eq. 5.6, we set $\lambda = N$ to ensure the constraints are valid in the ground state of the Ising problem. Therefore, we observe that the numerical precision necessary to formulate the MCPS problem scales as $1/N$. This effect compresses the gaps between the local (and global) minima, making it harder to differentiate between them. Using a direct embedding approach for QPUs requires even higher precision due to chains: the chain strengths scale with N, and therefore the encoding precision as $1/N^2$. This may be prohibitively low in large-scale problems and analog devices due to finite control errors.

	Greedy		HSS		Tabu		SA		2000Q		Advantage		Random	
	$\%_{\text{valid}}$	$f(w)$	$\%_{\text{valid}}$	$f(w)$	$\%_{\text{valid}}$	$f(w)$	$\%_{\text{valid}}$	$f(w)$	$\%_{\text{valid}}$	$f(w)$	$\%_{\text{valid}}$	$f(w)$	$\%_{\text{valid}}$	$f(w)$
$N = 10$	100%	3	100%	2	86%	2	100%	2	100%	2	100%	2	100%	4.325
$N = 30$	100%	9	100%	4	26%	10	98%	4	100%	5	100%	5	100%	12.367
$N = 100$	100%	23	92%	10.5	0%	29	88%	15	44%	31	74%	28.5	100%	35.25
$N = 300$	100%	57	96%	34.5	0%	73	-	-	-	-	-	-	100%	107.58
$N = 1000$	100%	160	100%	121	0%	196	-	-	-	-	-	-	100%	348.35
$N = 3000$	100%	455	100%	406.5	0%	558.5	-	-	-	-	-	-	100%	1054.41

Table 5.4: Results for all solvers. We present the percentage of problems for which each solver found valid solutions ($\%_{\text{valid}}$) and the median number of color switches ($f(w)$) for each problem size N cars.

CHAPTER 6

Conclusions and outlook

The work presented in this thesis centered around the applicability of a quantum optimization algorithm, the quantum annealing (QA) algorithm, in a variety of settings, as motivated in **Ch. 1**. We started by motivating QA through the adiabatic theorem, and the approximations which result in this metaheuristic quantum optimization algorithm. We further presented the specific implementation of QA in quantum hardware, specifically the quantum processing units (QPUs) produced by D-Wave Systems. We showcased the special conditions required to interact with the quantum hardware in order to formulate meaningful optimization problems. We presented the various limitations and boundaries of these conditions, and investigated the work required to use such algorithms in practice. The impact of individual QPU parameters, problem formulation, tunability, and algorithmic execution were studied using both canonical NP-hard optimization problems and real-world problems derived from data. We introduced different methods to accomplish this, and demonstrated the efficacy of each. Comparisons were made to classical (meta)heuristics in order to identify both the potential and bottlenecks of QA. In this chapter we summarize the main contributions of the paper, and contextualize the results presented in two main categories, the first focused on algorithmic advancement of quantum annealing, and the second on the implementation of the methods developed in this paper in practice to solve real-world problems.

Combinatorial optimization with quantum annealing hardware. There has been significant development of quantum hardware since the field's inception, particularly in recent years. The rise of scalable quantum architectures have allowed the implementation of the quantum annealing algorithm in programmable hardware, and many studies in literature have performed initial investigations into the type of problems that can be addressed with this algorithm. In **Ch. 3**, we presented the various ways in which quantum annealing QPUs require special conditions in order to formulate optimization problems: the types of variables, objective functions, hardware topologies, and open quantum system conditions all must be taken into account and unified in order to understand the QPUs' performance. To this end, we motivated the use of a canonical NP-hard problem– the maximum independent set (MIS) problem– to test the affects of these considerations in hardware performance directly in **Ch. 3**. We were able to demonstrate that quantum annealing shows potential as a heuristic optimization algorithm, in the sense that the QPU is able to solve small MIS problems, even when compared to classical competition algorithms. However, we also demonstrated the ways in which the workflow of using a QPU may hinder its performance in ways which have no parallel in classical metaheuristic algorithms. The QPU's performance in these experiments was highly dependent on the minor-embedding procedure required to map arbitrary graph structures to the fixed topology of the qubits in the QPU. We found how larger embeddings required higher degrees of precision to encode the problem, which in turn led to higher levels of noise in the logical space of the MIS problems. Because the MIS instances are constraintless in their QUBO formulation, we found these to be a qualitative bound on performance of the QPU in solving combinatorial optimization problems.

To investigate the interplay between graph structure and embedded problems in the QPU, we introduced an evolutionary algorithm– the (1+1)-CMA-ES algorithm– to tune one of the physical control parameters of the QPU, the annealing offsets. We shifted in time (adding an advance or delay) the point at which each *logical* graph variable started its annealing procedure (represented as a set of connected qubits). By allocating a budget of resources as a pre-processing tuning step, we were able to improve the QPUs ability in finding optima of the MIS problem. Furthermore, we showed that the rate at which such MIS optima were found increased as well. The results in **Ch. 3** indicate that in order to solve combinatorial optimization problems in general with direct QPU embedding, it is vital to mitigate some of the open quantum system effects that arise when using quantum hardware. This supports other works in the field of QA where hardware parameters have been exploited to enhance QPU performance. The novel contribution of this thesis was the demonstration that a black-box evolutionary strategy can be used to enhance QPU performance on arbitrarily structured graph problems. In general, however, we found that this tuning strategy adds significant time and complexity when solving MIS problems in practice, and so other algorithmic solutions may be needed.

Real-world use-cases for quantum and hybrid algorithms. The link between canonical NP-hard problems and real-world combinatorial optimization is not always obvious. The necessity of real-world modeling inherently requires additional factors which are not present in most basic optimization problems, regardless of complexity. In the context of quantum annealing, this requires developing methods to map different variable types, constraints, and data pre-processing techniques to forms which are admissible to quantum hardware. In **Ch. 4** we develop two such paradigms to model real-world optimization problems. The first derives a complete QUBO formulation for a logistics optimization problem (the shipment rerouting problem, the middle step of the less-than-truckload problem) directly from data. The constraints required to encode this problem convolve multiple techniques, modeling capacity constraints as packing constraints (as in packing and knapsack problems), distance minimization objectives (as in traveling salesperson), and a data pre-processing technique of suggesting alternative routes as in traffic-flow optimization. This process is typical, and highlights the key differences between real-world and canonical combinatorial optimization. Interestingly, while the complexity of the problem class may not be affected, the ability of certain metaheuristics may suffer due to this transformation. This is indeed what was observed in **Ch. 4**: while the feasible search space of the shipment rerouting problem was modeled correctly using adaptations of known problems in NP, this step made it much harder for quantum annealing (and similar) algorithms to find solutions. Therefore, a second method was demonstrated in **Ch. 4** in which a known problem in NP (the set cover problem) was used as an oracle to solve the problem of classifying real-world time series data. We were able to demonstrate how discretizing continuous time series data was sufficient to encode this problem as a QUBO, which we were then able to solve to reconstruct and classify various kinds of open-source time series data.

When solving such combinatorial optimization problems in practice, it was imperative to consider the ability to use the techniques derived in the previous chapters to solve real-world problems in practice. To overcome some of the difficulties in using QA hardware directly, **Ch. 5** introduced the idea of *hybrid quantum-classical* algorithms as possible solutions. A traffic flow optimization use-case was explored in which a turn-by-turn navigation service was build for a small fleet of buses during the Web Summit 2019 conference. The custom navigation service was built from scratch in order to appropriately use quantum optimization algorithms in a live real-world setting. As with other examples, multiple techniques were used in order to construct the QUBO. However, due to the on-the-fly nature of navigation, external services were required to be integrated into the QUBO formulation (e.g., live traffic data) in order to solve the problem. Therefore, multiple hybrid quantum optimization algorithms were tested, including building a custom hybrid algorithm. In the end, it was evident that robust hybrid quantum algorithms need to be used in order to both mitigate the QPU deficiencies (in the direct embedding), as well as handle the changing conditions of the QUBO given the live traffic conditions.

The second half of **Ch. 5** motivated another use-case for quantum optimization in the realm of production optimization, and represents the aggregate of the various connected concepts developed in this thesis. We presented and solved the *multi-car paint shop* optimization problem (MCPS), in which car bodies are assigned colors with the goal of minimizing the number of color switches in the sequence. We derived a new Ising model to represent the MCPS problem by introducing a simple optimization objective (anti-ferromagnetic couplings between adjacent spins) and a well-known constraint (k-hot constraints). Thus, we were able to both use existing modeling techniques and new techniques to create the Ising model. Furthermore, by making the decision variables (individual spins) each car in the sequence, we were able to map real-world data from the paint shop in Wolfsburg, Germany, directly to the Ising model, and thus solve the real-world application directly without any further modification. We compared direct QPU embedding (using two different generations of D-Wave QPUs), a hybrid quantum-classical algorithm, a simple greedy algorithm, and classical metaheuristics in solving these Ising models derived from data, thus providing a thorough benchmarking analysis based on techniques from previous chapters. By adapting a simple Ising model to represent this real-world combinatorial optimization problem, we were able to evaluate the efficacy of quantum optimization in solving these problems in practice. Based on the results in **Ch. 5**, we concluded that hybrid algorithms were still necessary given the number of available qubits in existing QPUs. Furthermore, while the hybrid algorithm was the best algorithm of those we tested, the performance at the largest problem sizes approached that of a simple greedy algorithm, further illustrating that the difficulty of solving real-world problems may affect quantum algorithms differently than classical ones.

It is evident that quantum algorithms hold significant potential in solving combinatorial optimization problems. The implementation of scalable quantum annealing hardware in particular has enabled the testing of a wide variety of both academic and real-world problems. The thorough study of the conditions necessary to both operate (and tune) these annealing QPUs, as well as model problems in suitable forms for QA, as presented in the body of work in this thesis, has highlighted that specific paradigms must be adopted when developing applications around quantum optimization and quantum annealing in particular. The continued growth of quantum annealing processors in number of qubits and density of connectivity will surely aid the development of more real-world applications. To complement this quantum hardware, additional methods such as those presented in the chapters of this thesis must also be continuously developed. It can be expected that many of the core concepts derived in this thesis will need to be incorporated in order to truly utilize quantum algorithms in real-world settings in the future.

Bibliography

[1] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741.

[2] Carl Hierholzer and Chr Wiener. Ueber die möglichkeit, einen linienzug ohne wiederholung und ohne unterbrechung zu umfahren. *Mathematische Annalen*, 6(1):30–32, March 1873.

[3] Wikipedia. Königsberg — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=K%C3%B6nigsberg&oldid=1067675023`, 2022. [Online; accessed 29-January-2022].

[4] Yuri Manin. Computable and uncomputable. *Sovetskoye Radio, Moscow*, 128, 1980.

[5] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, 22(5):563–591, 1980.

[6] Paul Benioff. Quantum Mechanical Models of Turing Machines That Dissipate No Energy. *Physical Review Letters*, 48(23):1581–1585, 1982.

[7] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, 1982.

[8] David Deutsch. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, July 1985.

[9] David Deutsch. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, December 1992.

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

[11] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212âĂŞ219, New York, NY, USA, 1996. Association for Computing Machinery.

[12] J. A. Jones and M. Mosca. Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer. *The Journal of Chemical Physics*, 109(5):1648–1653, August 1998.

[13] Isaac L. Chuang, Lieven M. K. Vandersypen, Xinlan Zhou, Debbie W. Leung, and Seth Lloyd. Experimental realization of a quantum algorithm. *Nature*, 393(6681):143–146, May 1998.

[14] Isaac L. Chuang, Neil Gershenfeld, and Mark Kubinec. Experimental implementation of fast quantum searching. *Phys. Rev. Lett.*, 80:3408–3411, Apr 1998.

[15] Stephan Gulde, Mark Riebe, Gavin P. T. Lancaster, Christoph Becher, Jürgen Eschner, Hartmut Häffner, Ferdinand Schmidt-Kaler, Isaac L. Chuang, and Rainer Blatt. Implementation of the deutsch–jozsa algorithm on an ion-trap quantum computer. *Nature*, 421(6918):48–50, Jan 2003.

[16] J. H. Plantenberg, P. C. de Groot, C. J. P. M. Harmans, and J. E. Mooij. Demonstration of controlled-not quantum gates on a pair of superconducting quantum bits. *Nature*, 447(7146):836–839, Jun 2007.

[17] Florian Dolde, Ville Bergholm, Ya Wang, Ingmar Jakobi, Boris Naydenov, Sébastien Pezzagna, Jan Meijer, Fedor Jelezko, Philipp Neumann, Thomas Schulte-Herbrüggen, Jacob Biamonte, and Jörg Wrachtrup. High-fidelity spin entanglement using optimal control. *Nature Communications*, 5(1):3371, Feb 2014.

[18] R. C. C. Leon, C. H. Yang, J. C. C. Hwang, J. Camirand Lemyre, T. Tanttu, W. Huang, K. W. Chan, K. Y. Tan, F. E. Hudson, K. M. Itoh, A. Morello, A. Laucht, M. Pioro-Ladrière, A. Saraiva, and A. S. Dzurak. Coherent spin control of s-, p-, d- and f-electrons in a silicon quantum dot. *Nature Communications*, 11(1):797, Feb 2020.

[19] Yang Wang, Xianli Zhang, Theodore A. Corcovilos, Aishwarya Kumar, and David S. Weiss. Coherent addressing of individual neutral atoms in a 3d optical lattice. *Phys. Rev. Lett.*, 115:043003, Jul 2015.

[20] Valentin Kasper, Daniel González-Cuadra, Apoorva Hegde, Andy Xia, Alexandre Dauphin, Felix Huber, Eberhard Tiemann, Maciej Lewenstein, Fred Jendrzejewski, and Philipp Hauke. Universal quantum computation and quantum error correction with ultracold atomic mixtures. *Quantum Science and Technology*, 7(1):015008, Nov 2021.

[21] Pascal Scholl, Michael Schuler, Hannah J. Williams, Alexander A. Eberharter, Daniel Barredo, Kai-Niklas Schymik, Vincent Lienhard, Louis-Paul Henry, Thomas C. Lang, Thierry Lahaye, Andreas M. Läuchli, and Antoine Browaeys. Quantum simulation of 2d antiferromagnets with hundreds of rydberg atoms. *Nature*, 595(7866):233–238, Jul 2021.

[22] Z. Vernon, N. Quesada, M. Liscidini, B. Morrison, M. Menotti, K. Tan, and J.E. Sipe. Scalable squeezed-light source for continuous-variable quantum sampling. *Phys. Rev. Applied*, 12:064024, Dec 2019.

[23] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M.

Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, Oct 2019.

[24] Gary J. Mooney, Gregory A. L. White, Charles D. Hill, and Lloyd C. L. Hollenberg. Whole-device entanglement in a 65-qubit superconducting quantum computer. *Advanced Quantum Technologies*, 4(10):2100061, 2021.

[25] 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, 2011.

[26] Trevor Lanting, Andrew D. King, Bram Evert, and Emile Hoskinson. Experimental demonstration of perturbative anticrossing mitigation using nonuniform driver Hamiltonians. *Physical Review A*, 96(4):042322, October 2017.

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

[28] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing - STOC '71*. ACM Press, 1971.

[29] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[30] IBM. CPLEX optimizer. `https://www.ibm.com/analytics/cplex-optimizer`. [Online; accessed 29-January-2022].

[31] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021.

[32] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, February 1996.

[33] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[34] James Kennedy, Russell C. Eberhart, and Yuhui Shi. *Swarm Intelligence.* Elsevier, 2001.

[35] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Phys. Rev. E*, 58:5355–5363, Nov 1998.

[36] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028 [quant-ph]*, November 2014.

[37] Michael Booth, Steven P Reinhardt, and Aidan Roy. Partitioning optimization problems for hybrid classical/quantum execution. *Technical Report*, pages 01–09, 2017.

[38] James King, Masoud Mohseni, William Bernoudy, Alexandre Fréchette, Hossein Sadeghi, Sergei V. Isakov, Hartmut Neven, and Mohammad H. Amin. Quantum-assisted genetic algorithm. *arXiv:1907.00707*, 2019.

[39] Shuntaro Okada, Masayuki Ohzeki, Masayoshi Terabe, and Shinichiro Taguchi. Improving solutions by embedding larger subproblems in a d-wave quantum annealer. *Scientific Reports*, 9(1):2098, 2019.

[40] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. Technical Report MIT-CTP-2936, preprint available as arXiv:quant-ph/0001106, Center for Theoretical Physics, Massachusetts Institute of Technology, 2000.

[41] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Review*, 50(4):755–787, 2008.

[42] Philipp Hauke, Helmut G. Katzgraber, Wolfgang Lechner, Hidetoshi Nishimori, and William D. Oliver. Perspectives of quantum annealing: Methods and implementations. *Reports on Progress in Physics*, 83:054401, March 2020.

[43] Francisco Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.

[44] Yoshiki Matsuda, Hidetoshi Nishimori, and Helmut G. Katzgraber. Ground-state statistics from annealing algorithms: Quantum versus classical approaches. *New Journal of Physics*, 11(7):073021, July 2009.

[45] Neil G. Dickson and Mohammad H. Amin. Algorithmic approach to adiabatic quantum optimization. *Phys. Rev. A*, 85:032303, Mar 2012.

[46] N. G. Dickson, M. W. Johnson, M. H. Amin, R. Harris, F. Altomare, A. J. Berkley, P. Bunyk, J. Cai, E. M. Chapple, P. Chavez, F. Cioata, T. Cirip, P. deBuen, M. Drew-Brook, C. Enderud, S. Gildert, F. Hamze, J. P. Hilton, E. Hoskinson, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Lanting, T. Mahon, R. Neufeld, T. Oh, I. Perminov, C. Petroff, A. Przybysz, C. Rich, P. Spear, A. Tcaciuc, M. C. Thom, E. Tolkacheva, S. Uchaikin, J. Wang, A. B. Wilson, Z. Merali, and G. Rose. Thermally assisted quantum annealing of a 16-qubit problem. *Nature Communications*, 4(1):1903, May 2013.

[47] Sebastian Feld, Christoph Roch, Thomas Gabor, Christian Seidel, Florian Neukart, Isabella Galter, Wolfgang Mauerer, and Claudia Linnhoff-Popien. A Hybrid Solution Method for the Capacitated Vehicle Routing Problem Using a Quantum Annealer. *Frontiers in ICT*, 6:13, 2019.

[48] Adam Douglass, Andrew D. King, and Jack Raymond. Constructing SAT Filters with a Quantum Annealer. In Marijn Heule and Sean Weaver, editors, *Theory and Applications of Satisfiability Testing – SAT 2015*, Lecture Notes in Computer Science, pages 104–120, Cham, 2015. Springer International Publishing.

[49] Jie Chen, Tobias Stollenwerk, and Nicholas Chancellor. Performance of domain-wall encoding for quantum annealing. *IEEE Transactions on Quantum Engineering*, 2:1–14, 2021.

[50] R.M. Karp. *Reducibility among combinatorial problems*, volume 85. Complexity of Computer Computations, 1972.

[51] Davide Venturelli, Dominic J. J. Marchand, and Galo Rojo. Quantum Annealing Implementation of Job-Shop Scheduling. *arXiv:1506.08479 [quant-ph]*, 2016.

[52] Eleanor G. Rieffel, Davide Venturelli, Bryan O'Gorman, Minh B. Do, Elicia M. Prystay, and Vadim N. Smelyanskiy. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, 14(1):1–36, January 2015.

[53] Olawale Titiloye and Alan Crispin. Quantum annealing of the graph coloring problem. *Discrete Optimization*, 8(2):376–384, May 2011.

[54] Olawale Titiloye and Alan Crispin. Graph coloring with a distributed hybrid quantum annealing algorithm. In James O'Shea, Ngoc Thanh Nguyen,

Keeley Crockett, Robert J. Howlett, and Lakhmi C. Jain, editors, *Agent and Multi-Agent Systems: Technologies and Applications*, pages 553–562, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[55] Daniel Marx. Graph colouring problems and their applications in scheduling. *Period. Polytech. Electr. Eng.*, 48:11–16, 2004.

[56] Kensuke Tamura, Tatsuhiko Shirai, Hosho Katsura, Shu Tanaka, and Nozomu Togawa. Performance comparison of typical binary-integer encodings in an ising machine. *IEEE Access*, 9:81032–81039, 2021.

[57] S. Karimi and P. Ronagh. Practical integer-to-binary mapping for quantum annealers. *Quantum Inf Process*, 18(94), 2019. preprint arxiv:1706.01945.

[58] David Harris and Sarah Harris. *Digital Design and Computer Architecture.* Elsevier, 2012.

[59] Nicholas Chancellor. Domain wall encoding of discrete variables for quantum annealing and QAOA. *Quantum Science and Technology*, 4(4):045004, August 2019.

[60] Jesse Berwald, Nicholas Chancellor, and Raouf Dridi. Understanding domain-wall encoding theoretically and experimentally. *arXiv:2108.12004*, 2021.

[61] John Golden and Daniel O'Malley. Reverse annealing for nonnegative/binary matrix factorization. *PLOS ONE*, 16(1):e0244026, January 2021.

[62] R. Harris, M. W. Johnson, T. Lanting, A. J. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, F. Cioata, I. Perminov, P. Spear, C. Enderud, C. Rich, S. Uchaikin, M. C. Thom, E. M. Chapple, J. Wang, B. Wilson, M. H. S. Amin, N. Dickson, K. Karimi, B. Macready, C. J. S. Truncik, and G. Rose. Experimental Investigation of an Eight-Qubit Unit Cell in a Superconducting Optimization Processor. *Physical Review B*, 82(2):024511, 2010.

[63] D-Wave System Documentation: QPU-Specific Characteristics. See documents at `https://docs.dwavesys.com/docs/latest/doc_physical_properties.html`, last checked November 2021.

[64] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-generation topology of d-wave quantum processors, 2020.

[65] Reinhard Diestel. *Graph Theory.* Springer.

[66] Vicky Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing*, 7(5):193–209, 2008.

[67] Vicky Choi. Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design. *Quantum Information Processing*, 10(3):343–353, June 2011.

[68] Jun Cai, William G. Macready, and Aidan Roy. A practical heuristic for finding graph minors, 2014.

[69] Tomas Boothby, Andrew D. King, and Aidan Roy. Fast clique minor generation in chimera qubit connectivity graphs. *Quantum Information Processing*, 15(1):495–508, 2016.

[70] Davide Venturelli, Salvatore Mandrà, Sergey Knysh, Bryan O'Gorman, Rupak Biswas, and Vadim Smelyanskiy. Quantum optimization of fully connected spin glasses. *Phys. Rev. X*, 5:031040, Sep 2015.

[71] Alexandre M. Zagoskin, Evgeni Ilichev, Miroslav Grajcar, Joseph J. Betouras, and Franco Nori. How to test the "quantumness" of a quantum computer? *arXiv:1401.2870 [quant-ph]*, January 2014.

[72] Troels F. Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V. Isakov, David Wecker, John M. Martinis, Daniel A. Lidar, and Matthias Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, 2014.

[73] Helmut G. Katzgraber, Firas Hamze, Zheng Zhu, Andrew 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.

[74] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Phys. Rev. X*, 6:031015, Aug 2016.

[75] Jack Raymond, Sheir Yarkoni, and Evgeny Andriyash. Global warming: Temperature estimation in annealers. *Frontiers in ICT*, 3:23, 2016.

[76] James King, Sheir Yarkoni, Mayssam M. Nevisi, Jeremy P. Hilton, and Catherine C. McGeoch. Benchmarking a quantum annealing processor with the time-to-target metric. *arXiv:1508.05087 [quant-ph]*, August 2015.

[77] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014.

[78] T. Bäck and S. Khuri. An evolutionary heuristic for the maximum independent set problem. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 531–535 vol.2, 1994.

[79] Yan-Long Fang and P. A. Warburton. Minimizing minor embedding energy: an application in quantum annealing. *Quantum Information Processing*, 19, 2020.

[80] Ravi Boppana and Magnús M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32(2):180–196, June 1992.

[81] Jeffrey Marshall, Eleanor G. Rieffel, and Itay Hen. Thermalization, freeze-out, and noise: Deciphering experimental quantum annealers. *Phys. Rev. Applied*, 8:064025, Dec 2017.

[82] D-Wave Systems. Reverse quantum annealing for local refinement of solutions. *D-Wave Whitepaper Series*, 2017.

[83] Yu Yamashiro, Masaki Ohkuwa, Hidetoshi Nishimori, and Daniel A. Lidar. Dynamics of reverse annealing for the fully connected p-spin model. *Phys. Rev. A*, 100:052321, Nov 2019.

[84] Nicholas Chancellor. Modernizing quantum annealing using local searches. *New Journal of Physics*, 19(2):023024, February 2017.

[85] Andrew D. King, Juan Carrasquilla, Jack Raymond, Isil Ozfidan, Evgeny Andriyash, Andrew Berkley, Mauricio Reis, Trevor Lanting, Richard Harris, Fabio Altomare, Kelly Boothby, Paul I. Bunyk, Colin Enderud, Alexandre Fréchette, Emile Hoskinson, Nicolas Ladizinsky, Travis Oh, Gabriel Poulin-Lamarre, Christopher Rich, Yuki Sato, Anatoly Yu. Smirnov, Loren J. Swenson, Mark H. Volkmann, Jed Whittaker, Jason Yao, Eric Ladizinsky, Mark W. Johnson, Jeremy Hilton, and Mohammad H. Amin. Observation of topological phenomena in a programmable lattice of 1,800 qubits. *Nature*, 560(7719):456–460, 2018.

[86] Davide Venturelli and Alexei Kondratyev. Reverse quantum annealing approach to portfolio optimization problems. *Quantum Machine Intelligence*, 1(1-2):17–30, 2019.

[87] Erica Grant, Travis S. Humble, and Benjamin Stump. Benchmarking Quantum Annealing Controls with Portfolio Optimization. *Physical Review Applied*, 15(1):014012, January 2021.

[88] Kazuki Ikeda, Yuma Nakamura, and Travis S. Humble. Application of Quantum Annealing to Nurse Scheduling Problem. *Scientific Reports*, 9(1):12837, September 2019.

[89] Daniele Ottaviani and Alfonso Amendola. Low rank non-negative matrix factorization with d-wave 2000q, 2018.

[90] M. H. S. Amin, Peter J. Love, and C. J. S. Truncik. Thermally Assisted Adiabatic Quantum Computation. *Physical Review Letters*, 100(6):060503, February 2008.

[91] Nicholas Chancellor, Philip J. D. Crowley, Tanja Ðurić, Walter Vinci, Mohammad H. Amin, Andrew G. Green, Paul A. Warburton, and Gabriel Aeppli. Disorder-induced entropic potential in a flux qubit quantum annealer. *arXiv:2006.07685 [quant-ph]*, June 2020.

[92] Jeffrey Marshall, Davide Venturelli, Itay Hen, and Eleanor G. Rieffel. Power of pausing: Advancing understanding of thermalization in experimental quantum annealers. *Phys. Rev. Applied*, 11:044083, Apr 2019.

[93] Evgeny Andriyash, Zhengbing Bian, Fabian Chudak, Marshall Drew-Brook, Andrew D. King, William G. Macready, and Aidan Roy. Boosting integer factoring performance via quantum annealing offsets. `https://www.dwavesys.com/resources/publications`, 2016.

[94] Riccardo Mengoni, Daniele Ottaviani, and Paolino Iorio. Breaking rsa security with a low noise d-wave 2000q quantum annealer: Computational times, limitations and prospects. *arXiv:2005.02268*, 2020.

[95] Nikolaus Hansen. The CMA Evolution Strategy: A Comparing Review. In Jose A. Lozano, Pedro Larrañaga, Iñaki Inza, and Endika Bengoetxea, editors, *Towards a New Evolutionary Computation: Advances in the Estimation of*

Distribution Algorithms, pages 75–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[96] Christian Igel, Thorsten Suttorp, and Nikolaus Hansen. A Computational Efficient Covariance Matrix Update and a (1+1)-CMA for Evolution Strategies. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 453–460, New York, NY, USA, 2006. ACM.

[97] Anne Auger and Nikolaus Hansen. Benchmarking the (1+1)-CMA-ES on the BBOB-2009 Noisy Testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2467–2472, New York, NY, USA, 2009. ACM.

[98] Florian Neukart, Gabriele Compostella, Christian Seidel, David von Dollen, Sheir Yarkoni, and Bob Parney. Traffic flow optimization using a quantum annealer. *Frontiers in ICT*, 4:29, 2017.

[99] D-Wave Systems has produced an open-source software stack in Python (Ocean tools) for accessing its quantum hardware, formulating problems for execution, and classical QUBO/Ising solvers. More information can be found at `https://docs.ocean.dwavesys.com/en/stable/`, last checked november 2021.

[100] Fred W. Glover and Manuel Laguna. *Tabu Search*. Springer US, 1997.

[101] Clark Alexander, Luke Shi, and Sofya Akhmametyeva. Using quantum mechanics to cluster time series. *arXiv*, 2018.

[102] Esma Aimeur, Gilles Brassard, and SÃľbastien Gambs. Quantum speed-up for unsupervised learning. *Machine Learning*, 90:261–287, 02 2013.

[103] Nathan Wiebe, Ashish Kapoor, and Krysta Svore. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Information and Computation*, 15:318–358, 03 2015.

[104] David Horn and Assaf Gottlieb. The method of quantum clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS '01, page 769âĂŞ776, Cambridge, MA, USA, 2001. MIT Press.

[105] Vaibhaw Kumar, Gideon Bass, Casey Tomlin, and Joseph Dulny. Quantum annealing for combinatorial clustering. *Quantum Information Processing*, 17(2):39, 2018.

[106] Florian Neukart, David Von Dollen, and Christian Seidel. Quantum-assisted cluster analysis on a quantum annealing device. *Frontiers in Physics*, 6:55, 2018.

[107] Kazuo Iwama, Junichi Teruyama, and Shuntaro Tsuyama. Reconstructing strings from substrings: Optimal randomized and average-case algorithms. *ArXiv*, 1808.00674, 2018.

[108] Jayadev Acharya, Hirakendu Das, Olgica Milenkovic, Alon Orlitsky, and Shengjun Pan. On reconstructing a string from its substring compositions. pages 1238 – 1242, 07 2010.

[109] Alan M Frieze, Franco P Preparata, and Eli Upfal. Optimal reconstruction of a sequence from its probes. *Journal of Computational Biology*, 6(3-4):361–368, 1999.

[110] Steven S Skiena and Gopalakrishnan Sundaram. Reconstructing strings from substrings. *Journal of Computational Biology*, 2(2):333–353, 1995.

[111] Patrick Schäfer and Mikael Högqvist. Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 516–527. ACM, 2012.

[112] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.

[113] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, May 2017.

[114] Anthony Bagnall, Jason Lines, William Vickers, and Eamonn Keogh. The uea & ucr time series classification repository. `www.timeseriesclassification.com`. Accessed: 2020-02-01.

[115] Abdullah Mueen, Eamonn Keogh, and Neal Young. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1154–1162. ACM, 2011.

[116] Chotirat (Ann) Ratanamahatana and Eamonn J. Keogh. Three myths about dynamic time warping data mining. In Hillol Kargupta, Jaideep Srivastava, Chandrika Kamath, and Arnold Goodman, editors, *SDM*, pages 506–510. SIAM, 2005.

[117] Goldberger AL, Amaral LAN, Glass L, Ivanov JM, Hausdorff amd PCh, Mark RG, Mietus JE, GB Moody, Peng C-K, and HE Stanley. Mit-bih long-term ecg database. `physionet.org/content/ltdb/1.0.0/`. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals (2003). Circulation. 101(23):e215-e220.

[118] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I. Lunze, Wojciech Samek, and Tobias Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific Data*, 7(1):154, May 2020.

[119] Dau H.A. Chinatown. `http://www.pedestrian.melbourne.vic.gov.au`. Accessed: 2020-02-01.

[120] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[121] European Commission. Clean transport, urban transport. `https://ec.europa.eu/transport/themes/urban/urban_mobility_en`.

[122] Sasan Amini, Eftychios Papapanagiotou, and Fritz Busch. *Traffic Management for Major Events*, pages 187–197. 07 2016.

[123] Julia Lindner. Big-data-analysen für verkehr in münchen. https://www.telefonica.de/news/corporate/2017/08/kooperation-von-telefonica-next-und-intraplan-big-data-analysen-fuer-verkehr-in-muenchen.html.

[124] Mapbox. Image uses map data from mapbox and openstreetmap and their data sources. to learn more, visit https://www.mapbox.com/about/maps/ and http://www.openstreetmap.org/copyright.

[125] T. Epping, W. Hochstättler, and P. Oertel. Complexity results on a paint shop problem. *Discrete Applied Mathematics*, 136(2):217–226, 2004. The 1st Cologne-Twente Workshop on Graphs and Combinatorial Optimization.

[126] P. Bonsma, T. Epping, and W. Hochstättler. Complexity results on restricted instances of a paint shop problem for words. *Discrete Appl. Math.*, 154(9):1335–1343, June 2006.

[127] M. Streif, S. Yarkoni, A. Skolik, F. Neukart, and M. Leib. Beating classical heuristics for the binary paint shop problem with the quantum approximate optimization algorithm. *Phys. Rev. A*, 104:012403, Jul 2021.

[128] Tamás Kis. On the complexity of the car sequencing problem. *Operations Research Letters*, 32(4):331–335, 2004.

Summary

In this thesis we explore the use of quantum annealing in solving real-world combinatorial optimization in a variety of settings. With the continuous development of quantum hardware and software, the potential of quantum algorithms is slowly being unlocked. However, some of the peculiarities of quantum hardware have no exact parallel in classical computing, and so the work in this thesis develops new methods to address quantum annealing for combinatorial optimization. We start by describing the quantum annealing (QA) paradigm in Chapter 2. The underlying theory of the algorithm as a metaheuristic quantum optimization algorithm is discussed. We outlined the basics concerning the quantum processing units (QPUs) produced by D-Wave Systems which use the transverse-field Ising Hamiltonian to implement QA. The mathematical description of the objective functions and models admissible to QA QPUs is presented, which sets the context for general combinatorial optimization using these QPUs.

In Chapter 3 we take a deeper dive into solving problems directly using qubits in QA by studying the maximum independent set (MIS) problem, a well-known NP-hard problem. We find how minor-embedding arbitrarily-structured graphs to fixed QPU topologies affects performance, and what parameters are involved in doing so. We then further attempt to mitigate the performance issues encountered by embedding procedures by tuning a control parameter of the QPU, the *annealing offsets*. We use the (1+1)-CMA-ES algorithm to tune these offsets as a pre-processing step. We find that the tuning procedure improves the QPU performance on average, and the CMA-ES routine is more efficient in doing so compared to other tuning routines. The trade-off between using the tuning routine and not is explored. We then transition to solving real-world problems, as opposed to simple known models, in Chapter 4. We introduce two methods of solving combinatorial optimization problems: (i) deriving a new QUBO/Ising model from real-world data, and (ii) processing real-world data and optimizing it using a well-known QUBO formulation. The former strategy is used to solve a real-world logistics combinatorial optimization problem, the *shipment rerouting problem*, an intermediate task of the more general less-than-truckload problem. The introduction of a mix of hard constraints (truck capacity) and simple optimization (reducing truck distances) into a combined QUBO results in a overhead in transformation which is not mitigated by the QUBO solvers tested. In the latter case of (ii), we use known discretization techniques to transform continuous time series data to a string of characters, which are then used to perform data reconstruction (from a known database). This is accomplished by formulating the reconstruction task as a set cover problem (another well-known NP-hard problem) which has a simple known QUBO formulation. We show how this set cover QUBO correctly reconstructs the original strings from fragments within the database, thus solving the original time series problem. We further show how to extend this to perform classification, which results in a semi-supervised classification algorithm.

In order to use quantum annealing in practice, we must overcome the QUBO/Ising modeling difficulties as well as the hardware limitations. To this end, we explore the use of hybrid quantum-classical algorithms in Chapter 5. We start by motivating a real-world traffic optimization problem: navigation a fleet of buses between known distances. A fully-automated hybrid optimization service is built and launched as part of a pilot project with a fleet of nine buses for the Web Summit 2019 conference. The connections between live traffic data used to build the traffic flow QUBO and accessing hybrid algorithms are constructed and deployed in practice. We find that while the service built results in stable navigation, the system is only able to handle small fleets. Therefore, in the second half of Chapter 5, we motivate another use-case for real-world optimization, the *paint shop problem*: painting a sequence of car bodies in a factory to minimize the number of color switches. We show how to construct an Ising model that directly represents the paint shop data with minimal overhead. By doing so, we include multiple techniques presented in the previous chapters of this thesis, and solve real-world problems directly on two different generations of D-Wave QPUs, as well as classical and hybrid algorithms. We are able to explore these Ising models at relatively large scales, where the problem sizes approach those of industrial relevance. While the performance of the hybrid algorithm tested showed the most utility, at the largest problem sizes a simple greedy algorithm was nearly as effective. We conclude on the work of this thesis in Chapter 6. The efficacy of the various methods used in this thesis are discussed, and the future development work necessary to apply quantum annealing (and similar) algorithms in practice is explored.

Samenvatting

In dit proefschrift onderzoeken we het gebruik van quantum annealing bij het oplossen van praktische combinatorische optimalisatie in verschillende situaties. Met de continue ontwikkeling van quantum hardware en -software wordt het potentieel van quantum algoritmes langzaam werkelijkheid. Sommige eigenaardigheden van quantum hardware hebben echter geen exacte parallel in klassieke informatica, en daarom ontwikkelt het werk in dit proefschrift nieuwe methoden om quantum annealing voor combinatorische optimalisatie te gebruiken. We beginnen met het beschrijven van het quantum annealing (QA) paradigma in Hoofdstuk 2. De onderliggende theorie van het algoritme als een metaheuristisch quantum optimalisatie-algoritme wordt besproken. We schetsten de basis met betrekking tot de quantum processing units (QPU's) geproduceerd door D-Wave Systems die de transversal field Ising Hamiltonian gebruiken om QA te implementeren. De wiskundige beschrijving van de te optimaliseren functies en modellen die toelaatbaar zijn voor QA QPU's wordt ook gepresenteerd, hetgeen de context vormt voor algemene combinatorische optimalisatie met behulp van deze QPU's.

In Hoofdstuk 3 gaan we dieper in op het rechtstreeks oplossen van problemen met behulp van qubits in QA door het probleem van de "maximum independent set" (MIS) te bestuderen, een bekend NP-hard probleem. We ontdekken hoe het insluiten van willekeurig gestructureerde grafieken in vaste QPU-topologieën de prestaties beïnvloedt, en welke parameters daarbij betrokken zijn. Vervolgens proberen we de prestatieproblemen die we tegenkomen bij het inbedden van procedures te verminderen door een controleparameter van de QPU, de *annealing offsets*, af te stemmen. We gebruiken het (1+1)-CMA-ES-algoritme om deze offsets af te stemmen als een voorbewerkingsstap. We vinden dat de afstemmingsprocedure de QPU-prestaties gemiddeld verbetert, en de CMA-ES-routine is hierin efficiënter in vergelijking met andere afstemmingsroutines. De afweging tussen het wel en niet gebruiken van de afstemmingsroutine wordt onderzocht. We gaan hierna in Hoofdstuk 4 over op het oplossen van problemen uit de echte wereld, in tegenstelling tot eenvoudige bekende modellen. We introduceren twee methoden voor het oplossen van combinatorische optimalisatieproblemen: (i) het afleiden van een nieuw QUBO/Ising-model uit praktijk data, en (ii) het verwerken van praktijk data en het optimaliseren ervan met behulp van een bekende QUBO-formulering. De eerste strategie wordt gebruikt om een praktische logistiek combinatorisch optimalisatieprobleem op te lossen, het *shipment rerouting-probleem*, een tussenliggende taak van het meer welbekende “less-than-truckload” probleem. De introductie van een mix van harde beperkingen (vrachtwagencapaciteit) en eenvoudige optimalisatie (verkleining van vrachtwagenafstanden) in een gecombineerde QUBO resulteert in een overhead in transformatie die niet wordt overkomen door de geteste QUBO-solvers. In het laatste geval van (ii), gebruiken we bekende discretisatietechnieken om continue tijdreeksgegevens om te zetten in een reeks karakters, die vervolgens worden gebruikt om gegevensreconstructie uit te voeren (uit een bekende database). Dit wordt bereikt door de reconstructietaak te formuleren als een set-cover-probleem (een ander bekend NP-hard probleem) dat een eenvoudige bekende QUBO-formulering heeft. We laten zien hoe deze “set cover QUBO” de originele strings correct reconstrueert uit fragmenten in de database, en zo het originele tijdreeksprobleem oplost. We laten verder zien hoe dit kan worden uitgebreid om classificatie uit te voeren, wat resulteert in een semi-supervised classificatie-algoritme.

Om quantum annealing in de praktijk te gebruiken, moeten we de QUBO/Ising-modelleringsproblemen en de hardwarebeperkingen overwinnen. Hiertoe onderzoeken we het gebruik van hybride kwantum-klassieke algoritmen in Hoofdstuk 5. We beginnen met het motiveren van een reëel verkeersoptimalisatieprobleem: het navigeren van een vloot bussen tussen bekende afstanden. Een volledig geautomatiseerde hybride optimalisatieservice wordt gebouwd en gelanceerd als onderdeel van een proefproject met een vloot van negen bussen voor de Web Summit 2019-conferentie. De verbindingen tussen live verkeersgegevens die worden gebruikt om de verkeersstroom QUBO op te bouwen en toegang te krijgen tot hybride algoritmen, worden in de praktijk gebouwd en ingezet. We merken dat hoewel de ingebouwde service resulteert in een stabiele navigatie, het systeem alleen kleine vloten aankan. Daarom motiveren we in de tweede helft van Hoofdstuk 5 een andere use-case voor real-world optimalisatie, het *lakwinkelprobleem*: het schilderen van een reeks autocarrosserieën in een fabriek met zo min mogelijk kleur wisselingen.

We laten zien hoe u een Ising-model construeert dat direct de gegevens van de spuiterij weergeeft met minimale overhead. Door dit te doen, nemen we meerdere technieken op die in de vorige hoofdstukken van dit proefschrift zijn gepresenteerd, en lossen we problemen uit de echte wereld rechtstreeks op op twee verschillende generaties D-Wave QPU's, evenals klassieke en hybride algoritmen. We zijn in staat om deze Ising-modellen op relatief grote schaal te onderzoeken, waarbij we de probleemgrootte van industriële relevantie benaderen. Hoewel de prestaties van het geteste hybride algoritme het meeste nut vertoonden, was een eenvoudig hebzuchtig algoritme bijna net zo effectief bij de grootste probleemgroottes. We concluderen het een en ander over het werk van dit proefschrift in Hoofdstuk 6. De nuttigheid van de verschillende methodes uit deze thesis worden gediscussierd, en het toekomstige werk dat nodig is om quantum annealing (en vergelijkbare) algoritmes in praktijk toe te passen wordt uitgelicht.

About the Author

Sheir Yarkoni was born 1990 in Tampa, FL, USA. He received his Bachelors in Physics and Computer Science (2012) and Masters in Physics (2014) at McGill University in Montréal, Canada. After graduating, Sheir started working at the quantum computing company D-Wave Systems in Vancouver, Canada. In September 2017 he joined Leiden Institute of Advanced Computer Science (LIACS) as an external PhD student under the supervision of Prof. Thomas Bäck and Prof. Aske Plaat. His research focused on designing and improving algorithms that could solve optimization problems using existing quantum processors. In November 2018, Sheir joined the Volkswagen Data:Lab in Munich, Germany, to focus on industrial applications of his research in the field of hybrid quantum-classical algorithms and optimization. Other areas of interest include benchmarking and characterization of quantum computers, as well as quantum machine learning.