Constrained quantum optimisation for customer data science

Puya Mirkarimi

A thesis presented for the degree of

Doctor of Philosophy

Department of Physics

Durham University

United Kingdom

May 2025

Constrained quantum optimisation for customer data science

Puya Mirkarimi

Abstract

One of the potential use cases of a quantum computer is to solve combinatorial optimisation problems more efficiently than is possible with the most advanced classical computers. Combinatorial optimisation problems that appear in industry often involve a large number of constraints. This thesis investigates algorithmic techniques that improve the performance of quantum optimisation algorithms in solving problems with constraints. We test these techniques on customer data science problems.

We study an alternative penalty method for encoding constraints in Ising Hamiltonians, which can be applied to various quantum algorithms. This method only introduces linear terms to the Ising Hamiltonian, allowing for more efficient use of hardware resources than the standard quadratic penalty method. These efficiency improvements are particularly beneficial for near-term devices with a limited number of qubits that are sparsely connected. We analyse the impact of using this alternative encoding method on the performance of the quantum approximate optimisation algorithm and quantum annealing applied to example problems in customer data science. Our results are based on numerical simulations and experiments on quantum hardware.

We introduce a new variant of the quantum approximate optimisation algorithm that improves its ability to solve problems with constraints. In this variant of the algorithm, the strengths of penalty functions are associated with additional variational parameters, allowing them to take different values in each layer of the quantum circuit. We perform numerical simulations of multiple variants of the quantum approximate optimisation algorithm and compare their performance.

Acknowledgements

My supervisor, Nick Chancellor, has been incredibly supportive and encouraging throughout my project. Thank you for the excellent guidance you gave me and for the time you dedicated to answering my questions. I thank Viv Kendon for being a fantastic Master's project supervisor and inspiring me to pursue a career in quantum computing. I would also like to thank Stuart Adams for taking on the role of primary supervisor after Nick moved to Newcastle and ensuring that my PhD could be completed smoothly. I thank dunnhumby and EPSRC for funding this PhD. Special thanks to Ross Williams and David Hoyle for their valuable guidance and for giving me an insight into the interesting work that dunnhumby does.

I am grateful for all the friends and memories I have made during this time. Being a member of QLM has made my PhD experience much more enjoyable. Thank you Gianluca, Bethan, Mitch, Jack, Steph, Joe, Ben, Archie, and everyone else in QLM for the lively discussions at lunch time, cake club, and the Vic. Shout out to Matt, Fraser, Luke, Alex, and Albert for regularly attending QLM football despite the frequent injuries. Thank you to all the members of the Iranian Society for making our events so memorable, especially Mamad, Asal, Avesta, Reza, the other Reza, Ali, the other Ali, and Khashayar. I never expected to find people who could cook such exceptional ghormeh sabzi and kebab in Durham.

To my parents, thank you for the sacrifices that gave me so many opportunities in life, including this PhD. I will always be grateful for your love and support. Arman, I am lucky to have such a kind and supportive brother. I hope your journey into fatherhood is full of joy.

I dedicate this thesis to my homeland with the hope that it may heal its wounds. The following poem, written in the 13th century as advice for kings, continues to resonate today.

بنی آدم اعضای یکدیگرند
که در آفرینش ز یک گوهرند
چو عضوی به‌درد آورد روزگار
دگر عضوها را نماند قرار
تو کز محنت دیگران بی‌غمی
نشاید که نامت نهند آدمی

Human beings are members of a whole,
In creation of one essence and soul.
If one member is afflicted with pain,
Other members uneasy will remain.
If you've no sympathy for human pain,
The name of human you cannot retain!

— Sa'di Shirazi, *Gulistan*
(English translation by M. Aryanpoor)

Contents

Declaration

The work in this thesis is based on research carried out at the Quantum Light and Matter Group, Department of Physics, Durham University, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification, and it is the sole work of the author unless referenced to the contrary in the text.

Some of the work presented in this thesis has been published in journals. The relevant publications are listed below.

Publications

P. Mirkarimi, I. Shukla, D. C. Hoyle, R. Williams, and N. Chancellor. Quantum optimization with linear Ising penalty functions for customer data science. *Physical Review Research*, 6(4):043241, 2024. doi:10.1103/PhysRevResearch.6.043241 [1]

P. Mirkarimi, D. C. Hoyle, R. Williams, and N. Chancellor. Experimental demonstration of improved quantum optimization with linear Ising penalties. New Journal of Physics, 26(10):103005, 2024. doi:10.1088/1367-2630/ad7e4a [2]

Nomenclature

1Q-PCP single-quarter promotion cannibalisation problem

2A-QAOA two-angle quantum approximate optimisation algorithm

2Q-PCP two-quarter promotion cannibalisation problem

4Q-PCP four-quarter promotion cannibalisation problem

AQC adiabatic quantum computing

COBYLA constrained optimization by linear approximation

LQQL linear-quadratic-quadratic-linear

MA-QAOA multi-angle quantum approximate optimisation algorithm

QA quantum annealing

QAOA quantum approximate optimisation algorithm

QLLQ quadratic-linear-linear-quadratic

QPU quantum processing unit

QUBO quadratic unconstrained binary optimisation

SA simulated annealing

SPSA simultaneous perturbation stochastic approximation

VPS-QAOA variable penalty strength quantum approximate optimisation algorithm

Chapter 1

Introduction

A quantum computer is a device that can leverage the properties of quantum mechanics to solve certain problems in a fundamentally more efficient way than a computer that behaves according to classical physics. The idea of quantum computing was independently proposed by Paul Benioff [3; 4], Yuri Manin [5] (according to [6]), and Richard Feynmann [7] in the early 1980s. At the time, it appeared that classical computers could not simulate all quantum mechanical systems efficiently, and it was suggested that a machine that operates according to quantum mechanics would be able to do so. However, it was not clear whether quantum computers could be experimentally realised.

Since then, the theoretical potential of quantum computing has become more clear with the discovery of quantum algorithms that produce speedups relative to their classical counterparts. The classic example is Shor's algorithm [8], which can factor an integer in an amount of time that scales polynomially with the number of bits required to represent it. There is no known polynomial time classical algorithm for this problem. The proposal of Shor's algorithm in 1994 demonstrated that quantum computers could be used to solve useful problems that are thought to be classically intractable, and this sparked a great amount of interest in quantum computing. Soon after, Lov Grover proposed a quantum algorithm for searching an unstructured database that produces a quadratic speedup over the best classical

strategy, which is brute force search [9]. This quadratic speedup was later shown to be optimal [10]. Grover's algorithm can be applied to various problems, including combinatorial optimisation problems, in which the goal is to minimise an objective function that can take a finite but very large number of possible solutions [11].

Meanwhile, significant experimental advances have been made in the development of quantum computers. Various hardware platforms for quantum computers are being explored, including superconductors [12; 13], trapped ions [14], neutral atoms [15], and photons [16]. A range of quantum devices are now commercially available and can be accessed on cloud platforms [17]. Claims of a quantum advantage in solving certain artificial problems [12; 16] and in simulating quantum systems [18] have been made. Recently, experimental implementations of quantum error correction have demonstrated that the logical error rates of quantum computations can be reduced by distributing quantum information across a quantum computer in a way that produces redundancy [19; 20].

It is hoped that further scientific progress in quantum hardware and algorithms will lead to a demonstration of a quantum advantage for combinatorial optimisation. To date, there has been no experimental demonstration of a quantum computer solving a combinatorial optimisation problem that cannot be solved classically in a reasonable amount of time [21]. However, recent evidence of a scaling advantage in approximate optimisation has been presented [22]. Combinatorial optimisation problems are ubiquitous in industry, and applications of quantum optimisation have been explored in a variety of fields, including finance [23; 24], molecular biology [25], material design [26], and air traffic management [27]. In this thesis, we consider an application of quantum optimisation in customer data science.

A common feature of combinatorial optimisation problems found in industry is that they often involve a large number of constraints [28]. A constraint is a feature of a problem that makes certain solutions undesirable. In commerce and retail settings, the number of constraints can be in the tens or hundreds [29; 30]. These constraints arise from both strategic and operational considerations. Therefore, to produce

the most value from quantum optimisation in a commercial setting, we require quantum algorithms that can outperform classical alternatives for problems that are significantly affected by constraints. This motivates the aim of this thesis, which is to develop and analyse algorithmic techniques that allow quantum optimisers to more efficiently solve problems that include constraints.

This thesis is organised as follows. In Chapter 2, we provide a background on quantum optimisation and define the example customer data science problems that we consider in this thesis. In Chapter 3, we investigate how linear Ising penalty functions can be used to encode Hamming weight constraints in quantum optimisation algorithms. We discuss the benefits and shortcomings of this method compared to other approaches. An important practical consideration is the sensitivity of linear Ising penalties to their penalty strengths, which we examine for problems with a single constraint and problems with many constraints. In Chapter 4, we numerically simulate two different quantum optimisation algorithms and study the use of linear Ising penalties. This allows us to determine the impact that this penalty method has on the performance of algorithms and how this is affected by changes in the penalty strengths. In Chapter 5, we perform experiments with linear Ising penalties on a quantum annealer. Our results in this chapter demonstrate the effectiveness of the linear method in making efficient use of the limited physical resources that are available on current quantum devices.

In Chapter 6, we introduce a new variant of the quantum approximate optimisation algorithm that can perform better in solving problems with constraints. This variant introduces additional parameters to the quantum circuit that allow penalty strengths to be changed between layers. We evaluate the performance of this algorithm in numerical simulations and perform a comparison against other variants. Finally, we provide the conclusions and outlook of our work in Chapter 7.

The data and code that support the findings of Chapters 3–5 are available at [31; 32]. In Chapter 6, the problem instances of [31] are used.

Chapter 2

Background

In this chapter, we provide a summary of prior work that underpins this thesis and introduce the key concepts that we use. We begin by introducing the Ising model and relating it to the quadratic unconstrained binary optimisation (QUBO) problem. We then provide explanations of how computations are performed in gate model quantum computing, adiabatic quantum computing (AQC), and quantum annealing (QA). We also describe the quantum approximate optimisation algorithm (QAOA) and simulated annealing (SA), which are algorithms for combinatorial optimisation problems. After that, we introduce the promotion cannibalisation problems that we base our work on and discuss existing methods for encoding a problem's constraints in quantum optimisation. Finally, we define the performance metrics that are used in this thesis and outline the numerical and experimental methods that we use.

2.1 Ising model and quadratic unconstrained binary optimisation

The Ising model is a classical representation of n magnetic spins $s_i = \pm 1$ that have pairwise interactions and interactions with an external magnetic field [33]. The

energy of a configuration $\mathbf{s} \in \{-1, +1\}^n$ is given by the Hamiltonian function

$$\mathcal{H}_I(\mathbf{s}) = \sum_{i=1}^{n} h_i s_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} J_{i,j} s_i s_j, \tag{2.1}$$

where $\mathbf{J} \in \mathbb{R}^{n \times n}$ is a strictly upper-triangular matrix that represents couplings and $\mathbf{h} \in \mathbb{R}^n$ represents local fields. Throughout this thesis, n refers to the number of variables or qubits. Recall that the Pauli operators are represented in matrix form as

$$\sigma_x \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \ \sigma_y \equiv \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}, \ \sigma_z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.2}$$

A quantum version of the Ising model for n qubits can be defined by replacing the spin variable s_i in Eq. (2.1) with the operator

$$\sigma_i^z = \mathbb{1}_2^{\otimes i-1} \otimes \sigma_z \otimes \mathbb{1}_2^{\otimes n-i}, \tag{2.3}$$

which applies the Pauli-Z operator σ_z to the ith qubit and the identity operator $\mathbb{1}_2$ to all other qubits. This gives the Ising Hamiltonian

$$\mathcal{H}_I = \sum_{i=1}^{n} h_i \sigma_i^z + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} J_{i,j} \sigma_i^z \sigma_j^z. \tag{2.4}$$

Finding the ground state of an Ising Hamiltonian is NP-hard, and many combinatorial optimisation problems of interest can be efficiently mapped to this problem [34; 35]. The Ising Hamiltonian can be directly encoded on many types of quantum computers, and this gives rise to various quantum algorithms for minimising $\mathcal{H}_I$.

Often, it is more natural to formulate an optimisation problem in terms of binary variables $\mathbf{x} \in \{0, 1\}^n$ than spin variables. Through the mapping $\sigma_i^z \mapsto 1 - 2x_i$, the problem of finding the ground state of $\mathcal{H}_I$ is equivalent to the QUBO problem

$$\text{find :} \qquad \arg\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{i,j} x_i x_j. \tag{2.5}$$

That is, the bitstring $\mathbf{x}^*$ that minimises the QUBO objective function $f(\mathbf{x})$ is the same bitstring that appears in the ground state $|\mathbf{x}^*\rangle$ of $\mathcal{H}_I$. The linear and quadratic term coefficients $\mathbf{a}$ and $\mathbf{b}$ in $f(\mathbf{x})$ are related to $\mathbf{h}$ and $\mathbf{J}$ through the equations

$$J_{i,j} = \frac{b_{i,j}}{4} \tag{2.6}$$

and

$$h_i = -\frac{a_i}{2} - \frac{1}{4} \sum_{j=1, j\neq i}^{n} b_{i,j}. \tag{2.7}$$

The QUBO objective function is often expressed in terms of a single upper-triangular matrix $Q \in \mathbb{R}^{n\times n}$ as

$$f(\mathbf{x}) = \mathbf{x}^{\mathsf{T}} Q \mathbf{x} = \sum_{i=1}^{n} \sum_{j=i}^{n} Q_{i,j} x_i x_j. \tag{2.8}$$

where $Q_{i,j} = b_{i,j}\ \forall j \neq i$ and $Q_{i,i} = a_i$. We use the formulation of Eq. (2.5) in this thesis because it explicitly separates the linear and quadratic terms, which is convenient for our discussion.

2.2 Gate model quantum computing

Just as there are different mathematical models of classical computation, such as the Turing machine [36] and the circuit model [37], there are also different models of quantum computation. The most common model for quantum computing is the gate model, which is also known as the quantum circuit model [38; 39; 40]. In the gate model, a computation involves the initialisation of qubits to a predetermined state, the operation of quantum gates that represent unitary transformations, and the measurement of qubit states. This model has a strong analogy to the circuit model of classical computing and is universal for quantum computing [40]. The gate model uses discrete variable encodings and processes data in discrete steps, making it a form of digital quantum computing. Another example of digital quantum computation is measurement-based quantum computing [41; 42], in which qubits are prepared in a highly entangled state and the computation progresses by sequentially performing measurements on each qubit.

The general expression for a pure state of a single qubit is

$$|\psi\rangle = a_0 |0\rangle + a_1 |1\rangle , \tag{2.9}$$

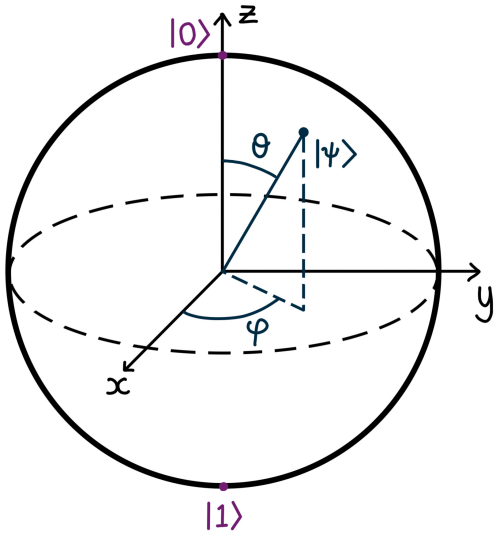


Figure 2.1: Diagram of the Bloch sphere. The pure state $|\psi\rangle$ of a single qubit is represented as a point on the surface of a unit sphere.

where the probability amplitudes $a_0, a_1 \in \mathbb{C}$ are normalised: $|a_0|^2 + |a_1|^2 = 1$. This can be rewritten as

$$|\psi\rangle = \mathrm{e}^{\mathrm{i}\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + \mathrm{e}^{\mathrm{i}\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right), \tag{2.10}$$

where $\gamma, \theta, \varphi \in \mathbb{R}$ are related to a_0 and a_1. The factor $\mathrm{e}^{\mathrm{i}\gamma}$ can be ignored as it is a global phase that does not have observable effects. Writing the state in this form allows it to be visually represented as a point on a unit sphere, where θ is the polar angle and φ is the azimuthal angle. This is known as the Bloch sphere, which is shown in Fig. 2.1.

Here, we will introduce the quantum gates that are used in this thesis. The Hadamard gate

$$H \equiv \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.11}$$

takes a qubit from a computational basis state to an equal superposition state. For example, $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. All quantum gates are unitary matrices and are therefore invertible. The inverse of the Hadamard gate is itself: $HH = \mathbb{1}$. We

consider the single-qubit rotation gates

$$R_X(\theta) \equiv \exp\left(-\mathrm{i}\frac{\theta}{2}\sigma_x\right) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\mathrm{i}\sin\left(\frac{\theta}{2}\right) \\ -\mathrm{i}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{2.12}$$

and

$$R_Z(\theta) \equiv \exp\left(-\mathrm{i}\frac{\theta}{2}\sigma_z\right) = \begin{pmatrix} \mathrm{e}^{-\mathrm{i}\frac{\theta}{2}} & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\frac{\theta}{2}} \end{pmatrix}, \tag{2.13}$$

which correspond to rotations by an angle θ about the x-axis and z-axis of the Bloch sphere respectively. The only two-qubit gate we consider is the rotation gate

$$R_{ZZ}(\theta) \equiv \exp\left(-\mathrm{i}\frac{\theta}{2}\sigma_z \otimes \sigma_z\right) = \begin{pmatrix} \mathrm{e}^{-\mathrm{i}\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & \mathrm{e}^{\mathrm{i}\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & \mathrm{e}^{-\mathrm{i}\frac{\theta}{2}} \end{pmatrix}. \tag{2.14}$$

This gate introduces a phase of $\mathrm{e}^{-\mathrm{i}\frac{\theta}{2}}$ if the two qubits are in the states $|00\rangle$ or $|11\rangle$ and a phase of $\mathrm{e}^{\mathrm{i}\frac{\theta}{2}}$ if they are in the states $|01\rangle$ or $|10\rangle$. Together, these quantum gates form a universal set [43], which means that any unitary operation on any number of qubits can be approximated to arbitrary precision by a circuit composed of these gates [40].

2.3 Adiabatic quantum computing

An interesting property of qubits is that they can be rotated from one state to another in a continuous fashion, whereas a classical bit can only be flipped between the states 0 and 1 by discrete operations. This allows for models of quantum computing in which qubit states evolve continuously in time, which are collectively referred to as continuous-time quantum computing [44]. Two closely related types of quantum computation that fit into this category are AQC [45] and QA [46; 47; 48; 49]. AQC and QA work by constructing a target Hamiltonian $\mathcal{H}_T$ that encodes the solution of the computational problem we wish to solve in its ground state. The task then becomes to find the ground state of this Hamiltonian.

In both AQC and QA, transitions between computational basis states are induced by a driver Hamiltonian $\mathcal{H}_D$. A common choice is the transverse-field driver Hamiltonian

$$\mathcal{H}_D = -\sum_{i=1}^{n} \sigma_i^x, \tag{2.15}$$

where $\sigma_i^x = \mathbb{1}_2^{\otimes i-1} \otimes \sigma_x \otimes \mathbb{1}_2^{\otimes n-i}$ applies the Pauli-X operator σ_x to the ith qubit. AQC and QA begin by preparing the system in the ground state of $\mathcal{H}_D$. For the transverse-field driver Hamiltonian, this is the state

$$|\psi(t=0)\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle = |+\rangle^{\otimes n}, \tag{2.16}$$

where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. This initial state gives an equal probability amplitude to all computational basis states. Alternative protocols exist that use other initial states. For example, in reverse annealing protocols, the system is initialised to a particular computational basis state, which may be an educated guess of the solution to the problem [50; 51; 52; 53].

In the AQC model, the system evolves according to a time-dependent Hamiltonian

$$\mathcal{H}(t) = A(t)\mathcal{H}_D + B(t)\mathcal{H}_T \tag{2.17}$$

from an initial time $t = 0$ to a final time t_f. $A(t)$ and $B(t)$ are control functions, which are time-dependent real numbers satisfying $A(0) \gg B(0)$ at $t = 0$ and $A(t_f) \ll B(t_f)$ at $t = t_f$. In this thesis, we consider the case where the target Hamiltonian $\mathcal{H}_T$ is the Ising Hamiltonian $\mathcal{H}_I$, which is given in Eq. (2.4). The total Hamiltonian is then

$$\mathcal{H}(t) = A(t)\mathcal{H}_D + B(t)\mathcal{H}_I. \tag{2.18}$$

A particular choice of $A(t)$ and $B(t)$ is called an annealing schedule. A common choice in numerical work is the linear annealing schedule

$$A(t) = 1 - \frac{t}{t_f}, \; B(t) = \frac{t}{t_f}. \tag{2.19}$$

We note that AQC with a linear annealing schedule does not reproduce the quadratic speedup of Grover's algorithm for unstructured search but using a more carefully chosen schedule does [54]. AQC operates in a closed system, and the dynamics are therefore described by the Schrödinger equation

$$\frac{\partial}{\partial t}\left|\psi(t)\right\rangle = -\mathrm{i}\mathcal{H}(t)\left|\psi(t)\right\rangle . \tag{2.20}$$

Note that in this equation and throughout this thesis, we set $\hbar = 1$ for convenience.

AQC relies on the adiabatic theorem, which was developed for discrete spectra in [55], according to [56]. The theorem ensures that a system that is prepared in the ground state of the initial Hamiltonian $\mathcal{H}(0) = \mathcal{H}_D$ will remain in the ground state of the instantaneous Hamiltonian $\mathcal{H}(t)$ throughout the duration of the evolution with arbitrarily high probability, provided that certain conditions are met. This means that AQC prepares the ground state of $\mathcal{H}_I$ with high fidelity, thus producing the optimal solution to the problem. There are many versions of the adiabatic theorem that each impose different conditions on the form of $\mathcal{H}(t)$. A commonly used approximate version of the adiabatic theorem that applies to Hamiltonians defined by Eq. (2.18) and Eq. (2.19) requires that the total anneal time t_f is large on the time scale set by $\frac{1}{g_{\min}^2}$, where $g_{\min}$ is the minimum spectral gap between the ground state and the first excited state of $\mathcal{H}(t)$ [57]. Rigorous versions of the theorem give different scalings of t_f with $g_{\min}$. A summary of different adiabatic theorems that can be applied to AQC can be found in [56].

In Fig. 2.2, we visually depict the distinction between gate-based quantum computing and AQC. In both models, the computation begins with the initialisation of qubits to some state, such as $|00\ldots0\rangle$, and ends with the measurement of some or all of the qubits. In the gate model, the time-evolution is determined by a sequence of unitary transformations that are performed on the qubits. In AQC, the time-evolution is instead determined by a time-dependent Hamiltonian and obeys the Schrödinger equation. It has been shown that any quantum circuit can be simulated by a time-dependent Hamiltonian in physically realistic settings with at

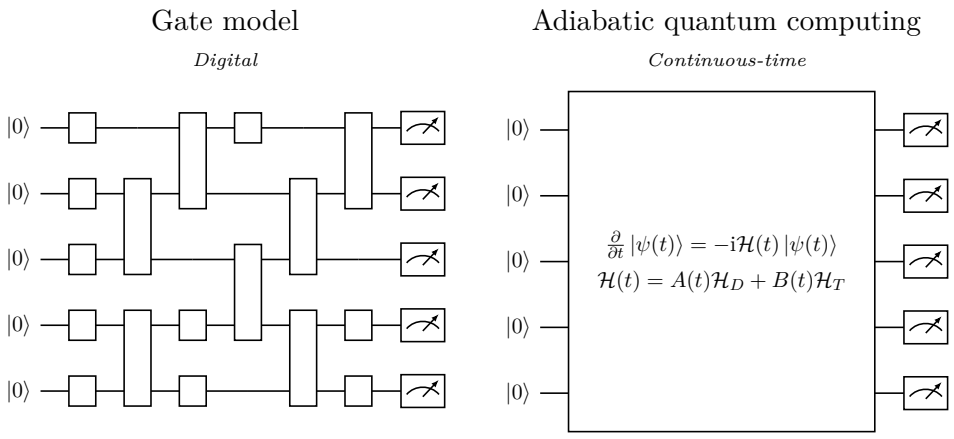


Figure 2.2: Diagrams illustrating how a computation is performed in the gate (left) and adiabatic (right) models of quantum computing.

most a polynomial overhead [58; 59], making AQC a universal model for quantum computing.

While AQC is a useful model for theoretical and numerical work, its experimental implementation is more challenging than for the gate model. This is because AQC requires the system to remain coherent for a period of time that is long enough to ensure adiabaticity. This is impractical on real-world devices because interactions with the environment produce decoherence, and its effect becomes more prominent as the anneal time is increased. In gate-based quantum computing, quantum error correction codes can be used to correct errors introduced by unwanted interactions with the environment as well as other types of errors [60]. If certain physical requirements are met, quantum error correction can be used to suppress the logical error rate to arbitrarily low levels [61; 6; 62]. See [63] for an introduction to quantum error correction in the gate model. Quantum error correction codes for AQC have been proposed, but they require high-order interactions that are experimentally impractical [64].

2.4 Quantum annealing

QA is a protocol that generalises AQC by removing the requirements of adiabaticity and closed system dynamics[1], which makes its experimental implementation more feasible. Devices called "quantum annealers" that physically implement quantum annealing have been created and are commercially available [13]. In the regime where the conditions for adiabaticity are not met, the theoretical guarantees of the adiabatic theorem no longer apply. However, other theoretical tools have been developed for the diabatic regime. It has been shown that coherent annealing with a schedule that monotonically increases the strength of $\mathcal{H}_I$ relative to $\mathcal{H}_D$ can only decrease the expectation value of $\mathcal{H}_I$ [65]. Hence, while a measurement of the final state may not produce the optimal solution, it produces better solutions than random guessing on average. In some situations, it has been found that annealing faster can benefit performance by taking the system through a path of diabatic transitions ending in a state that has a large overlap with the ground state [66; 67]. Various works have explored ways in which the QA Hamiltonian can be altered to produce an energy spectrum that allows diabatic transitions to be used advantageously [68; 69; 70].

QA is often treated as a metaheuristic because of the fact that it does not have the guarantee of producing the optimal solution. When the anneal time is short enough that multiple anneals can be performed, a common strategy is to generate many sample solutions with the quantum annealer and pick the best found solution. Note that the energy of a solution to an Ising problem can be evaluated in quadratic time, making it efficient to compare the quality of solutions and identify the best one.

While the AQC model allows for different types of driver and target Hamiltonians in Eq. (2.17), current quantum annealers are limited in the types of Hamiltonians they can implement. The flux qubit devices manufactured by D-Wave Quantum

[1]Note that AQC and QA are sometimes defined differently elsewhere in the literature.

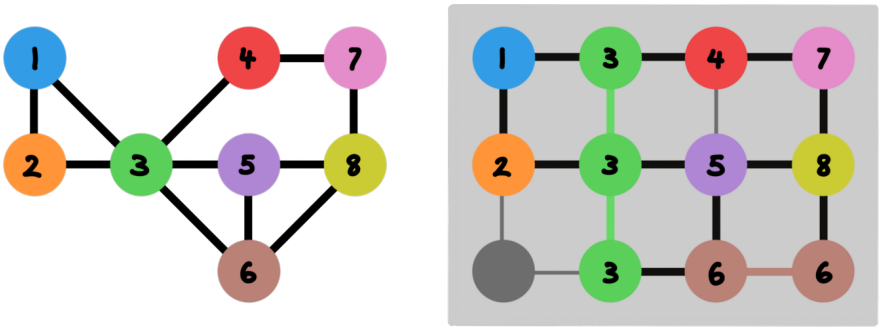
Inc. are the most advanced quantum annealers that are currently available [71]. D-Wave quantum annealers implement an approximation of the Hamiltonian given in Eq. (2.18) with a transverse-field driver Hamiltonian (Eq. (2.15)) and an Ising Hamiltonian given by

$$\mathcal{H}_I = \sum_{i=1}^{n} h_i \sigma_i^z + \sum_{i,j \in \chi} J_{i,j} \sigma_i^z \sigma_j^z, \tag{2.21}$$

where χ is the set of qubit pairs (i, j) that are physically coupled on the quantum processing unit (QPU) [72]. The values of the local fields h_i and couplings $J_{i,j}$ can be programmed, allowing for different problems to be expressed. Since D-Wave quantum annealers are restricted to implementing the transverse-field Ising model Hamiltonian, in practice, they are used for specific use cases. This includes solving combinatorial optimisation problems and performing quantum simulations of Ising-like systems [73]. In contrast, gate-based quantum computers are suitable for solving a larger variety of problems.

D-Wave quantum annealers and other superconducting qubit quantum devices that currently exist do not support couplings between arbitrary qubits. For example, the qubits in the D-Wave Advantage quantum annealers are coupled according to the "Pegasus" topology [72]. The Pegasus topology couples each qubit to a maximum of 15 other qubits. Often, the connectivity graph of the problem of interest (referred to as the "logical graph") cannot be directly mapped onto the graph representing the connectivity of the physical qubits (referred to as the "working graph"). In other words, the set χ may not contain the necessary pairs of qubits to allow a particular Ising Hamiltonian to be directly encoded.

To overcome the issue of limited qubit connectivity in QA, a common approach is to map each logical qubit to a chain of ferromagnetically coupled physical qubits such that the chains of qubits support the necessary couplings [74; 28]. This mapping is called a minor embedding because the graph representing the Ising Hamiltonian of the original problem is a minor of the graph representing the physical qubits. By picking the strength of the ferromagnetic couplings within the chains to be strong



(a) Logical graph (b) Minor embedding

Figure 2.3: (a) Logical graph of an example Ising problem, where nodes represent variables and edges represent couplings. (b) Minor embedding of the logical graph in (a) onto a square lattice working graph, where nodes and edges represent physical qubits and couplers respectively. Chains of physical qubits are labelled and coloured the same way as the logical variable they represent. Grey nodes and edges in (b) represent unused qubits and couplers respectively.

enough, the qubits in each chain will take the same value in the ground state. To demonstrate this technique, we show an example of a logical graph in Fig. 2.3(a) and a minor embedding of it onto an example working graph in Fig. 2.3(b). Note that in this example, the physical qubits in the working graph have at most four couplers available, whereas the variable labelled 3 in the logical problem is coupled to five other variables. Therefore, it would not be possible to directly map the logical graph onto the working graph in this example.

Despite being ferromagnetically coupled, the physical qubits in a chain can sometimes produce different measurement outcomes, which is called a chain break. In these cases, a post-processing algorithm can be applied to interpret the values of the affected logical variables. An example is to take a majority vote of the values of the qubits in the chain [75].

2.5 Quantum approximate optimisation algorithm

2.5.1 Standard QAOA

The QAOA is a hybrid quantum-classical optimisation algorithm based on the gate model [76]. The quantum circuit used in the QAOA produces a very similar time evolution as QA. In this circuit, the qubits are initially transformed from the state $|0\rangle^{\otimes n}$ to the state

$$|\psi_0\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}|j\rangle \tag{2.22}$$

by applying a Hadamard gate to every qubit. This is the same as the initial state in QA, given in Eq. (2.16). Then, p layers of gates are applied to the qubits and the final state is measured. In the lth layer of gates, the system evolves according to the Ising Hamiltonian $\mathcal{H}_I$ for a duration specified by an angle γ_l and then according to a mixer Hamiltonian $\mathcal{H}_M$ for a duration specified by an angle β_l. This is done by applying the phase separator unitary

$$U_I(\gamma_l) = \mathrm{e}^{-\mathrm{i}\gamma_l\mathcal{H}_I} \tag{2.23}$$

and then the mixer unitary

$$U_M(\beta_l) = \mathrm{e}^{-\mathrm{i}\beta_l\mathcal{H}_M}. \tag{2.24}$$

In the standard formulation of the QAOA, the mixer Hamiltonian is

$$\mathcal{H}_M = -\sum_{i=1}^{n}\sigma_i^x, \tag{2.25}$$

which is the same as the QA driver Hamiltonian in Eq. (2.15). The unitaries $U_I(\gamma_l)$ and $U_M(\beta_l)$ can be constructed with quantum gates using the relations

$$U_M(\beta_l) = \bigotimes_{i=1}^{n} R_X(-2\beta_l) \tag{2.26}$$

and

$$U_I(\gamma_l) = \left(\prod_{i=1,j>i}^{n} R_{Z_iZ_j}(2\gamma_l J_{i,j})\right)\left(\bigotimes_{i=1}^{n} R_Z(2\gamma_l h_i)\right), \tag{2.27}$$

where

$$R_{Z_iZ_j}(\theta) \equiv \exp\left(-\mathrm{i}\frac{\theta}{2}\sigma_i^z\sigma_j^z\right) \tag{2.28}$$

is the rotation gate $R_{ZZ}(\theta)$ applied to qubits i and j.

After applying p layers of phase separator and mixer unitaries, the system is measured in the computational basis, returning some classical solution. The state of the system immediately before measurement is

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = \mathrm{e}^{-\mathrm{i}\beta_p\mathcal{H}_M}\mathrm{e}^{-\mathrm{i}\gamma_p\mathcal{H}_I}\ldots\mathrm{e}^{-\mathrm{i}\beta_1\mathcal{H}_M}\mathrm{e}^{-\mathrm{i}\gamma_1\mathcal{H}_I}|\psi_0\rangle\,, \tag{2.29}$$

which is parameterised by $2p$ angles $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. The probability of a solution $\mathbf{x}$ being measured is $|\langle\mathbf{x}|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle|^2$. To relate the QAOA circuit to QA, we can set $\gamma_l = A\left(\frac{lt_f}{p}\right)$ and $\beta_l = B\left(\frac{lt_f}{p}\right)$. Then, the QAOA circuit becomes a discretisation of the QA time evolution. Hence, the adiabatic theorem implies that $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ can be chosen such that a measurement of $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ yields an optimal solution with arbitrarily high probability, provided that p is large enough to maintain adiabaticity.

Running QAOA circuits with enough layers that the adiabatic theorem can be applied does not typically produce desirable results on current quantum hardware. This is because decoherence and other sources of noise become dominant processes at the large circuit depths that are required. As a consequence, experiments on current hardware are limited to small values of p. Since performance is highly dependent on the choices of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, these angles are typically treated as variational parameters to be optimised classically, making the QAOA a variational quantum algorithm [77]. In this context, the structure of the QAOA circuit described by Eq. (2.29) is referred to as an ansatz. At each iteration of the variational algorithm, the QAOA circuit is run many times and the average objective value $\langle f(\mathbf{x})\rangle$ of the sampled solutions is calculated. The classical optimiser then updates the values of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ and repeats the process to minimise the cost function $\langle f(\mathbf{x})\rangle$. To avoid confusion, throughout this thesis, we refer to the function that is minimised by the classical optimiser in the QAOA as the cost function and the function corresponding to the original optimisation problem as the objective function $f(\mathbf{x})$.

A performance guarantee that applies for any number of layers p is that adding another layer to the QAOA circuit cannot increase the expectation value $\langle \boldsymbol{\gamma}, \boldsymbol{\beta} | \mathcal{H}_I | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle$ if $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ have optimal values [76]. This is because the unitary transformation of a QAOA circuit with $p+1$ layers can be made to match that of a circuit with p layers by setting all of the angles in the extra layer to zero and copying all other angles. While this suggests that a good method for improving the performance of the algorithm is to increase the number of layers, experiments on noisy quantum hardware show a drop in performance beyond a certain number of layers due to the effects of noise becoming more prominent as more layers are added [78]. Even in the ideal case where there is no noise, adding more layers can make it harder for a classical optimiser to find the optimal values of the circuit parameters within a limited number of circuit evaluations, which can make performance at higher values of p worse in practice.

A major challenge for the practicality of variational quantum algorithms is the barren plateau phenomenon, where the gradients in the cost landscape vanish exponentially with the problem size [79; 80]. In the presence of a barren plateau, an exponentially large number of measurements of the quantum circuit would be required to identify a direction in the parameter space that minimises the cost function. Barren plateaus have been observed for the QAOA applied to certain problems [81; 82].

2.5.2 Multi-angle QAOA

After the introduction of the QAOA, a large number of variants have been proposed to improve the original algorithm. These are reviewed in [83]. Some QAOA variants change the structure of the algorithm's ansatz. An example of this is the multi-angle quantum approximate optimisation algorithm (MA-QAOA), which was briefly introduced in [84] and was further investigated and given its name in [85]. This variant of the QAOA changes the ansatz by assigning a different variational parameter, or angle, to each rotation gate in the phase separator and mixer unit-

aries. As far as we are aware, all prior works on the MA-QAOA have been based on the MaxCut and Weighted MaxCut problems, which can be formulated as Ising Hamiltonians with only quadratic terms. For these problems, the phase separator unitary only needs to implement couplings, not local fields. However, since we are interested in Ising Hamiltonians that contain both quadratic and linear terms, we provide a more general definition of the MA-QAOA ansatz that also allows for local fields in the phase separator unitary.

The MA-QAOA ansatz can be defined by replacing the phase separator unitary in Eq. (2.23) with

$$U_I(\boldsymbol{\gamma}_l) = \prod_{i,j,J_{i,j}\neq 0} \mathrm{e}^{-\mathrm{i}\gamma_{l,i,j}J_{i,j}\sigma_i^z\sigma_j^z} \prod_{i,h_i\neq 0} \mathrm{e}^{-\mathrm{i}\gamma_{l,i,i}h_i\sigma_i^z} \tag{2.30}$$

and replacing the mixer unitary in Eq. (2.24) with

$$U_M(\boldsymbol{\beta}_l) = \prod_{i=1}^{n} \mathrm{e}^{\mathrm{i}\beta_{l,i}\sigma_i^x}. \tag{2.31}$$

With this ansatz, $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_p)$ contains a variational parameter for each nonzero coupling and each nonzero local field per layer, and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_p)$ contains a variational parameter for each qubit per layer. Hence, for an Ising Hamiltonian where every possible coupling and local field is nonzero, the total number of parameters is $\frac{pn(n+3)}{2}$, compared to $2p$ parameters in the standard QAOA ansatz.

The motivation behind the MA-QAOA is to improve the performance of the algorithm at a constant circuit depth by increasing the number of variational parameters. This is particularly useful for experiments on near-term devices, where good performance can only be attained at low circuit depths due to the presence of noise in the devices and the inability to circumvent this with quantum error correction. Indeed, numerical and experimental studies have demonstrated improved performance of the MA-QAOA compared to the QAOA at constant circuit depths when solving weighted and unweighted MaxCut problem instances [86; 87]. Although introducing more variational parameters gives the classical optimiser a more fine-grained ability to control the cost function, it can also make the classical

optimisation task more challenging [85]. In fact, the MA-QAOA has been found to be extremely prone to the barren plateaus phenomenon, even when the ansatz has a single layer [88]. For problem graphs that have symmetries, some of the parameters in the MA-QAOA can be removed without significantly affecting performance, indicating that there can sometimes be undesirable redundancy in the parameters [89].

2.6 Simulated annealing

QA is often viewed as a quantum analogue of SA, which is a classical stochastic algorithm for combinatorial optimisation [90]. SA is based on the physical process of annealing in metallurgy, where a metal is quickly heated to a high temperature and then slowly cooled to a new state that has fewer defects than the initial state and therefore has a lower energy. Following this analogy, the objective function of the optimisation problem is often referred to as the energy $E(\mathbf{x})$ instead of $f(\mathbf{x})$, and the aim is to minimise $E(\mathbf{x})$.

The SA algorithm starts in some arbitrary initial classical state $\mathbf{x}$. Similarly to QA, transitions between different classical states are induced with a strength that is swept from an initially large value down towards zero. SA differs to QA in that the transitions are induced by thermal fluctuations rather than quantum fluctuations. Specifically, each iteration of the algorithm considers a transition from the current state $\mathbf{x}_{\text{current}}$ to a random neighbouring state $\mathbf{x}_{\text{new}}$. If, for example, the neighbouring states are those with Hamming distance one, $\mathbf{x}_{\text{new}}$ can be selected by flipping a random bit in $\mathbf{x}_{\text{current}}$. The probability of performing the transition is given by the acceptance probability $P_{\text{accept}}(E_{\text{current}}, E_{\text{new}}, T)$, where $E_{\text{current}} = E(\mathbf{x}_{\text{current}})$, $E_{\text{new}} = E(\mathbf{x}_{\text{new}})$, and T is a parameter called the temperature. Transitions to states with lower (higher) energies are said to be downhill (uphill).

The temperature parameter controls the probability of accepting uphill transitions. For $T > 0$ and $E_{\text{new}} > E_{\text{current}}$, we must have some nonzero acceptance probability

$P_{\text{accept}} > 0$ to allow the algorithm to escape local minima by exploring higher-energy states. As T tends to zero, P_{accept} should tend to zero if $E_{\text{new}} > E_{\text{current}}$, and P_{accept} should tend to some positive value otherwise. This makes it less likely to accept uphill transitions as the temperature drops. A common choice of P_{accept} is

$$P_{\text{accept}}\left(E_{\text{current}}, E_{\text{new}}, T\right) = \min\left\{1, \mathrm{e}^{-\frac{E_{\text{new}}-E_{\text{current}}}{T}}\right\}, \tag{2.32}$$

where min selects the element with the minimum value. This choice satisfies the requirements above and has two additional properties of interest. One of these is that $P_{\text{accept}} = 1$ when $E_{\text{new}} \leq E_{\text{current}}$. That is, downhill transitions are always accepted. The other property is that uphill transitions are less likely to be accepted when $E_{\text{new}} - E_{\text{current}}$ is increased. These two properties are not essential for the algorithm to work, but are common in implementations of SA.

At each iteration of the SA algorithm, the temperature parameter T is updated according to an annealing schedule. The schedule is chosen such that T starts at some large value in the first iteration and decreases until reaching $T = 0$ by the final iteration. Therefore, the algorithm becomes increasingly more selective in accepting uphill transitions, and the extent of the search becomes more localised. When $T = 0$, SA reduces to a local search algorithm and only performs downhill transitions [91]. It has been proven that the final state converges to the globally optimal state as the total duration of the annealing schedule is increased [92]. However, the time required to guarantee a large probability of reaching the optimal state is usually longer than the time required to perform an exhaustive search of all states [93]. In practice, SA typically finds optimal or low-energy states in shorter timescales, where there is no theoretical guarantee of convergence to the optimal state. Thus, SA is usually used as a heuristic algorithm that samples good but not necessarily optimal solutions, much like QA.

2.7 Promotion cannibalisation problems

The optimisation problems considered in this thesis are simplified forms of a customer data science problem faced by retailers when planning to promote products with temporary price reductions. The goal of a product promotion is to generate additional revenue from sales of the product; however, it can also have the undesired effect of reducing revenue for other similar products. For example, a promotion of one brand of toothpaste may generate new sales for that brand at the partial expense of other brands' sales. This phenomenon is called cannibalisation [94; 95], and we will refer to cannibalisation arising from product promotions as *promotion cannibalisation*. When two similar products are promoted concurrently, the overall bilateral cannibalisation can result in minimal new sales and possibly even a net reduction in revenue. Therefore, one goal of promotion planning for retailers is to minimise the total revenue loss due to cannibalisation between concurrent promotions.

One way to model promotion cannibalisation is to consider the average amount of cannibalisation between pairs of products that are promoted at the same time and express this as a matrix C. In this model, the matrix element $C_{i,j}$ represents the average loss of revenue from sales of a promoted product i due to a simultaneous promotion of product j. The phenomenon of promotion cannibalisation occurs predominantly between substitutable products [D. C. Hoyle and R. Williams, personal communication, 14 April 2025], which are products that consumers usually use for the same purpose. Hence, the amount of promotion cannibalisation between two products is typically nonnegative and is usually zero if the products are in different categories. Based on this, we make the assumption that C is nonnegative, that is, $C_{i,j} \geq 0 \ \forall i, j$. This assumption is applied throughout this thesis.

One of the example problems that we consider is to find a promotion plan for one fiscal year that minimises the total amount of cannibalisation between pairs of products that are promoted in the same fiscal quarter. The promotion plan,

which selects the products that are promoted in each quarter, must satisfy various constraints that are imposed by the retailer. In this example, we consider three sets of constraints:

C1. Each quarter must have A products promoted.

C2. Each product must be promoted between $B_{\min}$ and $B_{\max}$ times by the end of the year.

C3. The same product cannot be promoted in two consecutive quarters.

In practice, there may be a larger number of constraints that a retailer would want to implement. We refer to this example problem as the four-quarter promotion cannibalisation problem (4Q-PCP).

Optimisation problems are typically mathematically formulated as an objective function followed by a set of constraints. The task is to find a solution that minimises the objective function while satisfying the constraints. Common examples are linear programming, where the objective function and constraints are linear in the variables, and quadratic programming, where the objective function is quadratic and the constraints are linear [96; 97]. The 4Q-PCP can be expressed as the binary quadratic programming problem

$$\text{find :} \qquad \arg\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{q=1}^{4} \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} \lambda_q C_{i,j} x_{i,q} x_{j,q} \tag{2.33}$$

$$\text{subject to :} \qquad \sum_{i=1}^{n_p} x_{i,q} = A \quad \forall q, \tag{2.34}$$

$$B_{\min} \leq \sum_{q=1}^{4} x_{i,q} \leq B_{\max} \quad \forall i, \tag{2.35}$$

$$x_{i,q} + x_{i,q+1} \leq 1 \quad \forall q \leq 3 \quad \forall i. \tag{2.36}$$

Here, a solution $\mathbf{x}$ specifies a promotion plan, where $x_{i,q} = 1$ $(= 0)$ if product i is (not) promoted during fiscal quarter q. There are $n = 4n_p$ variables, where $n_p \in \mathbb{Z}$ is the total number of products. The seasonal scale factor $\lambda_q \in \mathbb{R}$ quantifies the

expected volume of total sales in a particular quarter. This problem has $4n_p + 4$ constraints to satisfy in total. Eqs. (2.34), (2.35), and (2.36) represent the sets of constraints C1, C2, and C3 respectively. By comparing Eq. (2.33) with Eq. (2.5), we note that the quadratic coefficients in the objective function are equal to $\lambda_q(C_{i,j} + C_{j,i})$. Hence, while C is often asymmetric, there is no loss of generality if we assume it to be symmetric.

Another example problem we consider is the two-quarter promotion cannibalisation problem (2Q-PCP), which is concerned with finding the optimal promotion plan for two consecutive quarters. This problem can be expressed as

$$\text{find :} \qquad \arg\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{q=1}^{2}\sum_{i=1}^{n_p}\sum_{j=1}^{n_p} \lambda_q C_{i,j} x_{i,q} x_{j,q}, \tag{2.37}$$

$$\text{subject to :} \qquad \sum_{i=1}^{n_p} x_{i,q} = A \quad \forall q \tag{2.38}$$

$$x_{i,1} + x_{i,2} \leq 1 \quad \forall i. \tag{2.39}$$

Here, we do not have the constraints C2 as, apart from the trivial cases where all or no products are promoted, they are effectively already implemented by the constraints C3 when the problem has two quarters. There are $n = 2n_p$ variables and $n_p + 2$ constraints in this version of the problem.

Finally, the simplest formulation of the problem that we consider is the single-quarter promotion cannibalisation problem (1Q-PCP). In this problem, we only consider a single fiscal quarter, and there is a single constraint of the type C1 on the desired number of promotions. The corresponding quadratic programming problem is

$$\text{find :} \qquad \arg\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^{n_p}\sum_{j=1}^{n_p} C_{i,j} x_i x_j \tag{2.40}$$

$$\text{subject to :} \qquad \sum_{i=1}^{n_p} x_i = A. \tag{2.41}$$

There are $n = n_p$ variables in this formulation.

The objective functions of these problems are in QUBO form and can be converted to Ising form using Eq. (2.6) and Eq. (2.7). Taking the 1Q-PCP as an example, a

C matrix with a single pair of nonzero elements $C_{1,2} = C_{2,1} = 1$ substituted into Eq. (2.40) would produce a QUBO objective function containing a single quadratic term with coefficient $b_{1,2} = C_{1,2} + C_{2,1} = 2$. Hence, the objective function would be $f(\mathbf{x}) = 2x_1x_2$. In Ising form, this would correspond to $J_{1,2} = \frac{1}{2}$ and $h_1 = h_2 = -\frac{1}{2}$, which gives the Hamiltonian $\mathcal{H}_I = -\frac{1}{2}\sigma_1^z - \frac{1}{2}\sigma_2^z + \frac{1}{2}\sigma_1^z\sigma_2^z$.

2.8 Quadratic penalty method

As discussed earlier, quantum optimisation algorithms are particularly suitable for solving QUBO problems. The objective functions of the promotion cannibalisation problems we have described are already in QUBO form. However, this leaves out the problems' constraints, which are not considered in QUBO. Therefore, some technique must be used to ensure that the quantum algorithm provides solutions that satisfy the constraints.

The promotion cannibalisation problems that we have defined feature two different types of constraints. The constraints C1 are examples of linear equality constraints, which take the general form

$$\sum_{i=1}^{n} \mu_i x_i = k, \tag{2.42}$$

for some coefficients $\boldsymbol{\mu} \in \mathbb{R}^n$ and constraint value $k \in \mathbb{R}$. The constraints C2 and C3 are linear *inequality* constraints. These take the general form

$$l_{\min} \leq \sum_{i=1}^{n} \nu_i x_i \leq l_{\max}, \tag{2.43}$$

for some coefficients $\boldsymbol{\nu} \in \mathbb{R}^n$ and constraint values $l_{\min}, l_{\max} \in \mathbb{R}$. All of the constraints in our example problems are on the Hamming weights of the bitstrings because the coefficients μ_i and ν_i are either one or zero. The term *Hamming weight* refers to the number of bits that are equal to one. Hamming weight equality constraints such as the constraints C1 are also referred to as k-hot constraints as they only allow k binary variables to be equal to one.

The standard method for encoding constraints in quantum optimisation is the penalty method [34; 35]. In this approach, constraints are incorporated into the objective function through the addition of penalty functions that sufficiently raise the objective value of solutions that are infeasible (i.e. do not satisfy every constraint) so that the solution with the lowest objective value is feasible. This method is also used in classical quadratic programming [96]. For the general linear equality constraint given in Eq. (2.42), the most common penalty function used in quantum optimisation is the quadratic penalty function

$$P(\mathbf{x}) = \alpha_2 \left(\sum_{i=1}^{n} \mu_i x_i - k \right)^2, \tag{2.44}$$

where $\alpha_2 \in \mathbb{R}$ is the quadratic penalty strength. By squaring the brackets, we ensure that $P(\mathbf{x})$ contributes some positive value when the constraint is not satisfied. Therefore, as long as α_2 is chosen to be large enough, adding $P(\mathbf{x})$ to the objective function implements the constraint. Aside from the ability to rescale the function by changing the penalty strength parameter α_2, this penalty function has two desirable properties that are satisfied for all $\alpha_2 > 0$:

P1. $P(\mathbf{x}) = 0$ if $\mathbf{x}$ is feasible.

P2. $P(\mathbf{x}) > 0$ if $\mathbf{x}$ is infeasible.

The quadratic penalty method has some drawbacks that can severely inhibit the performance of a quantum optimiser. After expanding out the brackets in Eq. (2.44) for a quadratic penalty, we find quadratic terms with nonzero coefficients for all pairs of variables involved with the constraint. Therefore, encoding this as a Hamiltonian on a quantum computer requires all-to-all couplings between the associated qubits. Most quantum devices that are currently available to use do not support all-to-all couplings between the physical qubits, meaning that many Ising Hamiltonians of interest cannot be directly mapped to the hardware.

Various methods have been developed to resolve the issue of not being able to directly map an Ising Hamiltonian to the physical qubits. In the gate-based setting

of the QAOA, the quantum states of qubits can be swapped with a quantum gate called the SWAP gate. This allows for any two logical qubit states to be routed through the physical qubits so that they can be coupled [98; 99; 100]. The downside of the SWAP network scheme is that it increases the circuit depth. Previous work has produced optimised SWAP networks for the QAOA, taking into account the physical capabilities and limitations of current QPUs [99; 101]. In QA, SWAP gates are not available, so minor embedding is used instead. See Sec. 2.4 for a description of minor embedding. Typically, $\mathcal{O}(n^2)$ physical qubits are required to minor embed a graph onto the hardware graph of a D-Wave annealer [28].

In addition to requiring all-to-all couplings between the relevant qubits, another drawback of the quadratic penalty method is that it will typically reduce the effective dynamic range of qubit interactions. The quadratic penalty for the constraints C1 can be written as

$$P(\mathbf{x}) = \alpha_2 \left(\sum_{i=1}^{n_p} x_{i,q} - A \right)^2 . \tag{2.45}$$

for a given quarter q. Expanding out the brackets gives

$$P(\mathbf{x}) = \alpha_2 \left(\sum_{i=1}^{n_p} (1 - 2A) x_i + \sum_{i=1}^{n_p - 1} \sum_{j=i+1}^{n_p} 2 x_i x_j + A^2 \right) . \tag{2.46}$$

After mapping this to an Ising Hamiltonion with $x_i \mapsto (1 - \sigma_i^z)/2$, we get

$$P = \alpha_2 \left(\sum_{i=1}^{n_p - 1} \sum_{j=i+1}^{n_p} \frac{\sigma_i^z \sigma_j^z}{2} + \sum_{i=1}^{n_p} \left(\frac{n_p}{2} - A \right) \sigma_i^z + \frac{n_p(n_p+1)}{4} - n_p A + A^2 \right) . \tag{2.47}$$

Now, the couplings and local fields that are contributed to $\mathcal{H}_I$ can be read off as $J_{i,j} = \alpha_2/2$ and $h_i = \alpha_2(n_p/2 - A)$. As one might expect, the magnitudes of $\mathbf{J}$ and $\mathbf{h}$ increase with the magnitude of α_2. Additionally, the magnitude of $\mathbf{h}$ is proportional to the absolute difference $|n_p/2 - A|$. In other words, the quadratic penalty introduces strong local fields if the desired constraint value (number of promotions A) is far from half of the number of variables in the constraint (number of products n_p).

In QA, there are physical limitations on the range of values in $\mathbf{J}$ and $\mathbf{h}$ that can be implemented. Because of this, the device implements a normalised Ising Hamilto-

nian

$$\widetilde{\mathcal{H}}_I = \frac{1}{\mathcal{N}}\mathcal{H}_I, \tag{2.48}$$

where $\mathcal{N}$ is a normalisation factor that is usually chosen to be the minimum value that satisfies all physical constraints on $\mathbf{J}$ and $\mathbf{h}$. A penalty that introduces couplings and local fields with large magnitudes is therefore undesirable, as it will often result in a larger normalisation factor $\mathcal{N}$, reducing the effective dynamic range of the qubit interactions representing the unconstrained part of the problem.

Large couplings and local fields associated with a penalty function can also be detrimental in the QAOA, even though there is no direct physical limitation on size of the rotation angles that can be implemented in the unitary $\mathrm{e}^{-\mathrm{i}\gamma_k\mathcal{H}_I}$. The reasoning is that large interactions would risk rotating the phases by angles greater than 2π, which are effectively the same as rotations by less than 2π. Hence, despite it being possible to rotate phases by arbitrarily large angles, there is a limit beyond which no benefit is gained from attempting larger angles. This sets limits on the range of values of γ_l that are useful to consider, which is similar to the effect of having a physical dynamic range.

To encode inequality constraints with quadratic penalties, slack variables are generally required. Slack variables are auxiliary variables that are used to turn a penalty function for an equality constraint into a penalty function for an inequality constraint. They do not represent variables in the original problem. In the 4Q-PCP, $4\lceil\log_2(\Delta+1)\rceil$ binary encoded slack variables $\mathbf{s} \in \{0,1\}^{4\lceil\log_2(\Delta+1)\rceil}$ are required to encode the four constraints C2, where $\Delta = B_{\max} - B_{\min}$ is the range of the constraints. For cases where $\Delta + 1$ is a power of two, the penalty function takes the form

$$P(\mathbf{x},\mathbf{s}) = \alpha_2\left(\sum_{i=1}^{n_p} x_{i,q} + \sum_{j=1}^{\log_2(\Delta+1)} 2^{j-1}s_{j,q} - B_{\max}\right)^2. \tag{2.49}$$

The term $\sum_{j=1}^{\log_2(\Delta+1)} 2^{j-1}s_{j,q}$ gives the problem variables $x_{i,q}$ some "slack" by allowing the terms inside the brackets to sum to zero for any range of values of $\sum_{i=1}^{n_p} x_{i,q}$ that satisfies the inequality constraint. For the constraints C3, slack variables are

not needed as there are only two variables involved in each constraint. Quadratic penalties of the form

$$P(\mathbf{x}) = \alpha_2 x_{i,q} x_{i,q+1} \tag{2.50}$$

can be used to encode these constraints.

2.9 Alternative methods for encoding constraints in quantum optimisation

Various other approaches to encoding constraints in quantum optimisation algorithms have been proposed. These alternative methods intend to avoid some or all of the drawbacks of the quadratic penalty method and therefore produce better performance. In Chapter 3, we introduce an alternative penalty method involving only linear Ising terms. In this section, we provide a summary of some alternative methods for encoding constraints that are not considered in the rest of this thesis.

In QA, one approach is to engineer a constraint-preserving driver Hamiltonian $\mathcal{H}_D$ that drives transitions between feasible states, but never from a feasible state to an infeasible state [102; 103]. The constraint-preserving driver Hamiltonian should be designed so that its ground state in the feasible subspace is a superposition of all feasible states. Then, after initialising the system to this superposition state, QA can be performed without ever leaving the feasible subspace. This method does not require the addition of penalty functions or any other terms to $\mathcal{H}_I$, thereby avoiding all of the difficulties of the penalty method. Furthermore, it offers a natural way of restricting the search space to the feasible region, which, depending on the problem, can significantly reduce the number of states that are considered.

Although this approach is theoretically appealing, it is difficult to physically realise constraint-preserving driver Hamiltonians in QA. This is because they require multi-body interaction terms, which are more challenging to experimentally im-

plement than the bit flip terms σ_i^x in the standard driver Hamiltonian given in Eq. (2.15). It has been suggested that a more experimentally feasible route towards implementing this approach may be to combine it with a parity-based encoding scheme that requires hardware that supports 3- and 4-body interactions [104; 105]. For some constraints, mathematically formulating a suitable constraint-preserving driver Hamiltonian is a challenging task in itself. Another challenge is that in an open system, errors can occur, and there is no longer a guarantee that the system will remain in the feasible subspace [106]. So far, there has been no experimental demonstration of constraint implementation with a suitably engineered driver Hamiltonian on a quantum annealing device.

Building on the idea of encoding constraints in the driver Hamiltonian, Hadfield et al. proposed a generalization of the QAOA called the quantum alternating operator ansatz, which can be used to similarly restrict the search space to the feasible region [107]. In this extension of the QAOA, the phase separator and mixer unitaries do not necessarily need to correspond to time evolution under a fixed local Hamiltonian. They are therefore written as $U_I(\gamma_l)$ and $U_M(\beta_l)$ instead of $\mathrm{e}^{-\mathrm{i}\gamma_l\mathcal{H}_I}$ and $\mathrm{e}^{-\mathrm{i}\beta_l\mathcal{H}_M}$. This generalisation allows for more efficient implementations of mixers that preserve constraints. Numerical results show that using the XY-model ring or clique mixers to implement k-hot constraints outperforms the quadratic penalty method [108; 109]. These mixers can currently be implemented on gate model quantum computers, but they come at the cost of a larger circuit depth than the standard QAOA. An advantage of the gate model setting is that error correction can be used to prevent the system from leaving the feasible subspace due to noise [106].

Another approach to encoding constraints in QA uses sets of Ising terms called "gadgets" that are designed to interface with each other and have ground states that satisfy certain properties [110; 111; 112]. Some of the qubits in the gadgets represent variables in the original optimisation problem, while other qubits represent ancillary variables that are used to implement the desired properties. The gadgets

are combined to form a larger Ising problem that encodes the original optimisation problem, and the properties of each gadget ensure that the constraints are satisfied in the ground state. Previous work on this approach has been specifically tailored to the Chimera graph, which is the hardware graph of old generations of D-Wave devices. By tailoring the design of the gadgets to the hardware, it is possible to avoid the need for minor embedding. The use of ancillary qubits means that there are still more physical qubits required than variables in the original problem, just like with minor embedding. For certain problems, the gadget-based approach can be more efficient in terms of the number of physical qubits and dynamic range requirements when compared to the quadratic penalty method with minor embedding. To apply this approach to different hardware topologies or different types of constraints, new gadget designs are needed.

For inequality constraints, there are various methods that make more efficient use of resources than the standard method of using quadratic penalties with slack variables, which is used in Eq. (2.49). One alternative method, referred to as unbalanced penalisation [113], uses an expansion of a decaying exponential as the penalty. This is a heuristic method in the sense that it does not always satisfy property P1 of a penalty function. The benefit is that it avoids the use of slack variables, which reduces the total number of variables required to encode the problem compared to approaches using slack variables.

Another way to encode inequality constraints is the iterative quadratic polynomial method [114]. In this approach, a linear system of equations is formulated and classically solved in order to determine whether a penalty function with no slack variables can encode the desired constraint. If such a penalty function is found, it can be used. If it is not found, a new linear system is solved to determine whether a penalty function with a single slack variable exists for the constraint. Slack variables are repeatedly added until a suitable penalty function is found by solving the linear system. This process can find penalty functions using fewer slack variables than the standard approach with a quadratic penalty. When there

are multiple constraints defined over shared variables, the master-satellite method can be used to complement this approach [114]. This reformulates some of the linear systems to take advantage of the fact that infeasible states that are already penalised by one penalty function do not need to be penalised again by other penalty functions. This relaxation in the requirements of the linear systems can produce penalty functions with even fewer slack variables.

Some methods implement constraints without encoding them in a Hamiltonian or a set of quantum gates. An approach that can be used in gate-based settings is to make use of quantum Zeno dynamics to restrict the evolution of the system to the feasible subspace through repeated projective measurements [115]. This method extends the circuit depth and uses auxiliary qubits. It likely requires quantum error correction to be effective. Another approach is to enforce constraints in the classical component of an iterative quantum algorithm such as the recursive QAOA [116]. In these algorithms, variables are iteratively fixed to reduce the size of the problem. The rules for fixing variables can be constructed in a way that satisfies the constraints.

2.10 Performance metrics

We use various metrics to measure the performance of algorithms, which we define here. For a quantum algorithm that produces the state $|\psi\rangle$ before measurement, we define the success probability

$$P_S = \sum_{\mathbf{x}^* \in X^*} |\langle \mathbf{x}^* | \psi \rangle|^2 , \tag{2.51}$$

which indicates the probability of measuring an optimal solution. X^* denotes the set of optimal solutions. We similarly define the feasible probability

$$P_F = \sum_{\tilde{\mathbf{x}} \in \tilde{X}} |\langle \tilde{\mathbf{x}} | \psi \rangle|^2 , \tag{2.52}$$

which is the probability of measuring a solution from the set $\tilde{X}$ of solutions that satisfy all of the problem's constraints. While P_S and P_F can be calculated in sim-

ulations, these quantities can only be approximated when sampling an algorithm a finite number of times. For experimental runs of quantum and classical algorithms, we refer to the fraction of sampled solutions that are optimal as S and the fraction of samples that are feasible as F.

A useful metric to assess the quality of a particular solution $\mathbf{x}$ is the approximation ratio

$$R = 1 - \frac{f(\mathbf{x}) - f_{\min}}{f_{\max} - f_{\min}}, \tag{2.53}$$

where $f_{\min}$ and $f_{\max}$ are the minimum and maximum objective values of the constrained optimisation problem respectively. When $\mathbf{x}$ is feasible, i.e. satisfies all constraints in the problem, R ranges from 0 in the worst case ($f(\mathbf{x}) = f_{\max}$) to 1 for optimal solutions ($f(\mathbf{x}) = f_{\min}$). We refer to the numerator of the fraction in Eq. (2.53) as the residual energy E_{res}:

$$E_{\text{res}} = f(\mathbf{x}) - f_{\min}. \tag{2.54}$$

2.11 Numerical and experimental methods

This work made extensive use of the Python programming language [117]. We used the Python libraries NumPy [118] and SciPy [119] for computationally intensive calculations and Matplotlib [120] to produce plots. All linear fits were obtained using the implementation of the weighted least-squares method in `scipy.optimize.curve_fit`. PyQUBO [121] was used for formulating QUBO and Ising problem instances.

Gurobi Optimizer [122] was accessed through the GurobiPy Python interface and used to find optimal solutions of problem instances as well as their minimum and maximum objective values. We note that Gurobi operates at an adjustable numerical precision, which can lead to minor differences in results depending on the software version and solver parameters that are used. Throughout this thesis, we

used Gurobi version 10.0.2 with a single thread and the default values of all other solver parameters.

The C matrices used in our numerical analysis of the 1Q-PCP and the 2Q-PCP in Chapters 3, 4, and 6 were generated by selecting symmetric off-diagonal matrix elements $C_{i,j} = C_{j,i}$ uniformly at random from the interval $[0.1, 1.0)$. We set all main diagonal matrix elements $C_{i,i}$ to 0. According to this method, we generated 10,000 C matrices corresponding to different problem instances for each number of products between 6 and 18. Each instance was assigned a unique ID, which we sometimes use to refer to individual instances. Each ID is specified by the number of products in the instance, followed by an underscore, followed by a zero-based index number. For example, the ID `9_12` refers to the thirteenth instance with nine products. We have used the same C matrices for the analysis of the 1Q-PCP and the 2Q-PCP.

The C matrices used in the experimental tests on the D-Wave annealer in Chapter 5 were generated in a similar way by randomly selecting symmetric off-diagonal from the interval $[0.1, 1.0)$ and setting the main diagonal elements to zero. These matrices were also made sparse by setting some of the off-diagonal elements equal to zero. To do this, we chose a minimum number of nonzero elements per product, and matrix elements $C_{i,j}$ were randomly selected and set to zero if both products i and j had more than the minimum number of nonzero elements. This was repeated until every matrix element had been considered. In the context of C matrices, we use the term "connectivity" to refer to the number of nonzero cannibalisation interactions a particular product has with other products. For the D-Wave runs on the 1Q-PCP, we generated 10,000 C matrices with 100 products and the minimum connectivity for each product set to 3. Since some products end up with more than the minimum number of nonzero elements, the average connectivity is ≈ 3.4 for these instances. We generated another 10,000 C matrices for the D-Wave runs on the 4Q-PCP with 10 products. For these matrices, the minimum connectivity was set to 5 and the average connectivity is 5.1.

The majority of the simulations and intensive computations used for this thesis were performed on the Hamilton high performance computing cluster at Durham University. The SciPy function `expm_multiply` was used to simulate QA by discretising the time evolution in the manner that is described in [65]. For simulations of the QAOA, the `qasm_simulator` backend in the Qiskit SDK [123] was used to simulate quantum circuits without noise. We used Qiskit version 0.25.2. The implementation of the QAOA in Chapter 4 differs to the implementation of the QAOA and its variants in Chapter 6. For the simulations in Chapter 4, the constrained optimization by linear approximation (COBYLA) method [124] provided in `scipy.optimize.minimize` was used as the classical optimiser for the QAOA, whereas the simultaneous perturbation stochastic approximation (SPSA) method [125] provided in `qiskit.algorithms.optimizers.SPSA` was used for the simulations in Chapter 6. Further details of the two different implementations of the QAOA are specified in Sec. 4.6 and Sec. 6.4.

The QA experiments in Chapter 5 were conducted on the Advantage_system6.3 QPU [72], which is a D-Wave Advantage quantum annealer. The D-Wave Ocean SDK [126] was used to interface with the annealer. In each run, 1,000 solutions were sampled by the annealer. We used the `find_embedding` function in Ocean to calculate minor embeddings before problems were submitted to the annealer. The `find_embedding` function takes a seed parameter as input, which determines the resulting embedding. We set this parameter to a unique value for each problem instance for all experiments other than those for the 1Q-PCP using the quadratic penalty method, where the first 100 instances' minor embeddings were reused for the rest of the problem instances to save computation time. Chain strengths were calculated using the `uniform_torque_compensation` function in D-Wave Ocean with a prefactor of 1.414, which is the default behaviour in Ocean. Chain breaks were resolved by a majority vote strategy [75], which is the default behaviour in Ocean.

For QA experiments on the 1Q-PCP using the quadratic penalty method,

where the objective function is fully connected, we considered using the `find_clique_embedding` function instead of `find_embedding`. However, we found that although this heuristic provided more efficient minor embeddings, using `find_embedding` resulted in better performance of the annealer on average. We explain why we believe this happens in Sec. 5.2.2.

In Chapter 5, simulated annealing runs were performed using the sampler `SimulatedAnnealingSampler`, which is provided in Ocean. The solver was set to sample 1,000 solutions with a seed of 0. All other solver parameters were set to their default values.

Chapter 3

Encoding constraints with linear Ising penalties

The majority of the results in this chapter have been published in [1] and some of the results have been published in [2].

3.1 Introduction

In QA and the QAOA, the standard approach to encoding constraints is the quadratic penalty method [34; 35]. Quadratic penalty functions can make the interaction graph of the Ising Hamiltonian much more dense and introduce large energy scales to its energy landscape. This can negatively impact the performance of a quantum optimiser, especially on near-term devices that have many physical limitations [127]. Overcoming these physical limitations requires new strategies that address issues that are not typically a concern in classical optimisation. A particularly important issue to overcome in QA is that problems often need to be mapped to quasi-planar hardware graphs using strategies such as minor embedding [74] or parity encoding [104; 128], which introduce large overheads. Some progress has been made with new encoding strategies, such as the domain-wall encoding [129; 130; 131], which can reduce connectivity when used for one-hot encodings. Other efficiency improvements have been made with the development of

new methods for encoding constraints, such as those mentioned in Sec. 2.9.

In this chapter, we consider the use of a penalty method that involves only linear Ising terms and study how it can be used to encode linear equality Hamming weight constraints of the form given in Eq. (2.34). Our work is based on the promotion cannibalisation problems introduced in Sec. 2.7, but it generalises to many other problems. Previous works have suggested this penalty method in various different contexts [24; 132; 133]. Unlike the quadratic penalty method, the linear Ising penalty method does not change the Ising Hamiltonian's connectivity. Furthermore, the energy scales introduced by the linear method are often smaller than those introduced by the quadratic method. Because of these reductions in resource costs, theoretical arguments can be made for a better performance of quantum optimisation with the linear method than with the quadratic method. However, while the quadratic penalty method can always exactly implement a given constraint, there is no such guarantee for the linear Ising penalty method. Thus, the linear Ising penalty method is more applicable to certain problem types than others. We find that the customer data science problems that we consider are well suited for the linear method.

3.2 Linear Ising penalty method

In classical computing, penalty functions that are linear in $\mathbf{x}$ are sometimes used [134]. To satisfy property P2, non-Ising operations, i.e. mathematical operations that do not appear in Eq. (2.4), are required. For example, the linear equality constraint given in Eq. (2.42) can be implemented with the linear penalty $\alpha_1 \left|\sum_{i=1}^{n} \mu_i x_i - k\right|$, which makes use of the non-Ising operator $|\cdot|$. Computing using non-Ising operations in quantum optimisation introduces the challenge of encoding the operation as a circuit or Hamiltonian. de la Grand'rive and Hullo [135] showed that the QAOA can be extended to implement non-Ising linear penalties for inequality constraints by computing $\max(\cdot)$, which is another non-Ising operation. However,

their approach comes at the cost of requiring ancillary qubits and a more complex circuit.

We consider the use of linear Ising penalty functions, which are linear penalty functions that do not contain non-Ising operations. Such a penalty function would not contribute any couplings and may require weaker local fields than the quadratic penalty method. This is particularly desirable in quantum optimisation as it avoids increasing the number of couplings in the objective function and can be more efficient with the hardware's dynamic range [132; 136], which are the two main drawbacks of the quadratic penalty method. By not using any non-Ising operations, we ensure that the penalty function can be implemented on quantum hardware without requiring an encoding that introduces new couplings and ancillary qubits. However, the removal of non-Ising operations requires giving up on property P2 of a conventional penalty function, which means this penalty method does not produce the desired ground state in $\mathcal{H}_I$ in all cases. In other words, this penalty method is generally inexact in the sense that for some problems, there is no value of the penalty strength that ensures that the penalised objective function has a feasible optimal solution. We argue that despite this, it is worth pursuing this penalty method for problems that it is well-suited to as it allows problems to be encoded on current quantum hardware with a smaller physical overhead than other methods. This type of linear penalty method has been previously suggested for QA in the contexts of portfolio optimisation [24] and quantum machine learning [133], but neither of these two studies analysed this approach beyond making the observation that it can be used to encode the problems considered.

In [132], Ohzeki developed a method for implementing constraints with linear terms in quantum optimisation by taking the partition function of a QUBO objective function and applying a Hubbard-Stratonovich transformation [137; 138]. Although the mathematical motivation in Ohzeki's work is different to what we present here, the resulting algorithm is effectively the same as applying linear Ising penalties. Later works have applied the method introduced by Ohzeki to other problems,

including problems with inequality constraints [139; 136; 140]. Much of our work builds on Ohzeki's work and subsequent studies using Ohzeki's method. It has been observed that the method described by Ohzeki does not have the theoretical guarantee of being able to exactly implement all hard constraints [139]. We quantify the prevalence of problem instances that cannot be exactly constrained with this method in Chapters 4 and 5 and identify a problem structure that reduces the likelihood of this occurring in Sec. 3.3. In Chapter 4, we analyse the performance of the linear Ising penalty method in closed systems and study how the dynamics are influenced by the choice of penalty strength parameters, which has not been done in previous work as far as we are aware. When sampled solutions are infeasible, a post-processing step has been proposed that produces feasible solutions from infeasible solutions using the fewest number of bit flips possible [139]. In Sec. 3.5, we propose a different strategy of selectively switching some linear Ising penalties to quadratic penalties until feasible solutions are sampled.

For the general linear equality constraint given in Eq. (2.42), the corresponding linear Ising penalty is

$$P(\mathbf{x}) = \alpha_1 \left(\sum_{i=1}^{n} \mu_i x_i - k \right), \tag{3.1}$$

where the penalty strength α_1 can be positive or negative. The equality constraints that we consider, such as those in Eq. (2.34), are Hamming weight constraints. The linear Ising penalty for one of the constraints in Eq. (2.34) is

$$P(\mathbf{x}) = \alpha_1 \left(\sum_{i=1}^{n_p} x_{i,q} - A \right). \tag{3.2}$$

In the Ising formulation, this corresponds to local fields of $-\alpha_1/2$ on each variable, up to an unimportant constant offset of $\alpha_1 A$. In the rest of this thesis, when using the term *linear penalty*, we refer to linear Ising penalties of the form in Eq. (3.2) rather than linear penalties with non-Ising operations, which are used in classical computing.

3.3 Problems that are suitable for the linear Ising penalty method

As mentioned in Sec. 3.2, some problems cannot be exactly constrained with the linear penalty method with any value of the penalty strength. Therefore, it is important to identify problems and problem structures for which there is a high likelihood that a given instance is amenable to the linear penalty method. In this section, we identify a property of the promotion cannibalisation problems we have defined that we claim makes them able to be constrained with linear penalties more often than for random QUBO problem instances. We further offer some guidance as to how to identify whether the linear penalty method is suitable for a given problem or not.

We only consider constraints on the Hamming weight of solutions, for which the corresponding linear Ising penalty is also a function of Hamming weight. A useful visualisation is a plot of the objective function $f(\mathbf{x})$ or penalty term $P(\mathbf{x})$ against the Hamming weight $w(\mathbf{x})$ of solutions $\mathbf{x}$. We plot $P(\mathbf{x})$ against $w(\mathbf{x})$ for the quadratic and linear penalty functions for the constraint $\sum_{i=1}^{6} x_i = 2$ in Fig. 3.1(b) and Fig. 3.1(e) respectively. Due to the squaring of the term in the quadratic penalty function $P(\mathbf{x}) = \alpha_2 \left(\sum_{i=1}^{6} x_i - 2\right)^2$, the function is always positive when $w(\mathbf{x}) \neq 2$, which makes it clear that this penalty function can always create a feasible optimal solution for a large enough value of α_2. However, the linear penalty function $P(\mathbf{x}) = \alpha_1 \left(\sum_{i=1}^{6} x_i - 2\right)$ is a line with a gradient α_1 on this plot and is negative for all infeasible solutions with Hamming weights $w(\mathbf{x}) > 2$ when α_1 is negative. It therefore has the undesired effect of lowering the objective value of some infeasible solutions.

Since we assume that the matrix C is nonnegative, the QUBO objective function of the 1Q-PCP only has quadratic terms with nonnegative coefficients. This produces a structure in the problem that can be seen in Fig. 3.1(d). The lower envelope of

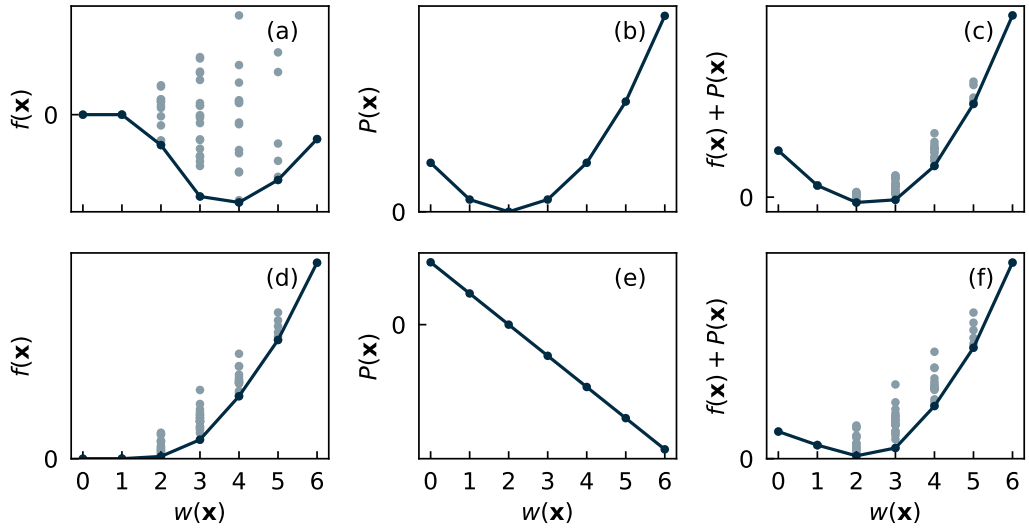


Figure 3.1: Objective values $f(\mathbf{x})$ of every possible solution $\mathbf{x}$ plotted against the Hamming weight $w(\mathbf{x})$ for (a) an example QUBO instance with both positive and negative quadratic term coefficients and (d) an instance of the 1Q-PCP. The minimum value of $f(\mathbf{x})$ for each Hamming weight is plotted in a darker colour and lines connecting these points are shown to guide the eye. (b) Quadratic and (e) linear penalty functions $P(\mathbf{x})$ for the equality constraint $\sum_{i=1}^{6} x_i = 2$ are similarly plotted. (c) The constrained objective function after adding a quadratic penalty to the objective function of the QUBO instance shown in (a). (f) The constrained objective function after adding a linear penalty to the objective function of promotion cannibalisation problem instance shown in (d). Figure adapted from [1].

the plot of $f(\mathbf{x})$ against $w(\mathbf{x})$, i.e. the minimum objective values at each Hamming weight, increases monotonically with $w(\mathbf{x})$. In other words, for every solution with some Hamming weight w, there exists another solution with Hamming weight $w-1$ that has the same or lower objective value. This is true not only for this particular instance, but for all instances of the 1Q-PCP. To prove this, assume there exists a solution $\mathbf{x}_a$ with objective value $f(\mathbf{x}_a)$ and Hamming weight w, where $f(\mathbf{x}_a)$ is strictly less than the objective values of all solutions with Hamming weight $w-1$. We can flip one of the variables of $\mathbf{x}_a$ from 1 to 0 and get another solution $\mathbf{x}_b$ with a Hamming weight $w-1$. Since the 1Q-PCP QUBO objective function does not have any terms with negative coefficients, flipping a variable from 1 to 0 can never increase the objective value. Hence, it must be that $f(\mathbf{x}_b) \leq f(\mathbf{x}_a)$, which contradicts the proposition that $f(\mathbf{x}_a)$ is strictly less than the objective value of all solutions with Hamming weight $w-1$. Therefore, solutions with a Hamming weight

w cannot have an objective value that is strictly less than the objective value of all solutions with Hamming weight $w-1$. The same is true for the 2Q-PCP and 4Q-PCP when considering the Hamming weight of the variables for a particular quarter while the other quarters' variables have fixed values because the objective functions have no cross terms between different quarters.

The monotonic relationship between the Hamming weight and the minimum objective value makes this problem particularly amenable to the linear Ising penalty method. We plot $f(\mathbf{x})$ against the Hamming weight of $\mathbf{x}$ for a 1Q-PCP instance with six products in Fig. 3.1(d) and for another six-variable QUBO instance with both positive and negative quadratic term coefficients in Fig. 3.1(a). If we want to implement the constraint $\sum_{i=1}^{6} x_i = 2$ for both of these problems, the penalised objective functions must have minimum values at $w(\mathbf{x}) = 2$. For the promotion cannibalization problem, we only need to penalise the solutions with Hamming weights less than two because of the fact that the minimum value of $f(\mathbf{x})$ is monotonically increasing with $w(\mathbf{x})$. A linear penalty (shown in Fig. 3.1(d)) achieves this, but it also has the undesired effect of lowering the objective value of solutions with Hamming weights greater than two. However, the monotonic structure means that these higher Hamming weight solutions have large existing objective values, which is often enough to compensate for this. Therefore, we claim that the linear penalty method is more likely to be able to produce the correct optimal solution for random instances of the 1Q-PCP than for a random QUBO instance without this structure. We find that for this instance of the promotion cannibalization problem, the linear penalty method is successful in producing a constrained objective function, which is shown in Fig. 3.1(f). For the other QUBO problem instance, there does not exist any value of α_1 that produces an optimal solution with a Hamming weight of two, so the quadratic penalty method is used instead (Figs. 3.1(a)–3.1(c)).

Indeed, the claim that the nonnegativity of the quadratic term coefficients in the objective function makes it more amenable to the linear penalty method can be

verified numerically. We have generated 100 QUBO instances with 20 variables that only have quadratic terms with coefficients selected uniformly at random from the interval $[0, 1)$, and we have generated another set of 100 instances where the coefficients are selected from the interval $[-1, 1)$. We have performed a search strategy on all these instances to find the value of α_1 up to a precision of 10^{-5} for a linear penalty that implements the constraint $\sum_i x_i = 5$. For the instance set with nonnegative quadratic term coefficients, we find that a value of α_1 that implements this constraint can be found for all 100 instances. However, for the instance set involving both negative and positive quadratic term coefficients, the search could only find a value of α_1 that implements the constraint for 10 out of 100 instances, up to the specified precision. We also performed this analysis on another set of instances that have quadratic term coefficients selected randomly from the interval $[-1, 0)$. For this set, a value of α_1 implementing the constraint could only be found for 5 out of 100 instances. This is because nonpositive quadratic term coefficients do not produce the monotonic structure seen in Fig. 3.1(d).

The monotonic structure in Fig. 3.1(d) does not guarantee that the linear penalty method is successful for all instances of the 1Q-PCP. Adding a linear function to another function can be thought of as applying a vertical shear. Therefore, a linear penalty function can only produce the desired optimal solution for a given objective function if the lower envelope in a plot of $f(\mathbf{x})$ against $w(\mathbf{x})$ can be vertically sheared such that the minimum value is at the desired Hamming weight. This is the case for the lower envelope in Fig. 3.1(d) because both its value and its gradient are monotonically increasing with $w(\mathbf{x})$. However, in general, the gradient of this line is not necessarily monotonically increasing for all 1Q-PCP instances. There exist some 1Q-PCP instances for which a given Hamming weight equality constraint cannot be exactly implemented with linear penalties.

Looking beyond the example of the promotion cannibalisation problem, we can identify some features that make a given problem more suitable for the linear penalty method. One desirable feature, as we have identified above, is that the

variables involved in the constraint do not appear in quadratic terms with negative coefficients in the objective function. This ensures that the minimum objective value is monotonically increasing with the Hamming weight of the variables in the constraint, which makes it more likely that a linear penalty can implement the constraint exactly. We note that this is based on the assumption that the constraints are functions of Hamming weight. For constraints that are not functions of Hamming weight, e.g. general equality constraints given by Eq. (2.42) with coefficients μ_i that can take values other than 0 and 1, our reasoning would not apply. Whether linear penalties are suitable for constraints that are not Hamming weight constraints is a question that could be answered in future work.

Another desirable feature would be that the constraints do not need to be implemented exactly. I.e. if they are soft constraints instead of hard constraints. Then, the fact that the linear penalty cannot produce the exact constraint value would not be important as long as a close enough constraint value can be produced. In this case, the structure of the objective function may not matter as much. For the same reason, the linear penalty method may be particularly useful for inequality constraints, but we have not studied this. A drawback to using linear penalties for inequality constraints is that without introducing quadratic terms, slack variables cannot be used to ensure that all feasible states have the same amount of penalty. This means that a linear penalty would change the objective value of some feasible states with respect to the objective value of other feasible states that have different Hamming weights, which changes the optimisation problem being solved.

Another aspect to consider is that the linear penalty method is most useful when applied to a few constraints acting on many variables, rather than many constraints acting on a few variables. This is because the main benefits of the linear penalty method, which are reducing the connectivity of the logical graph and making more efficient use of a limited dynamic range, are more prominent when applied to a constraint involving a large number of variables. Furthermore, if there are fewer constraints encoded with linear penalties, it is easier to tune the penalty strengths

because there are fewer of them in total. As an example, the linear penalty method could theoretically be applied to the constraints C1 and the constraints C2 in the 4Q-PCP. There are four constraints C1 acting on n_p variables each and n_p constraints C2 acting on four variables each. In a real-world scenario where n_p is significantly larger than four, it makes more sense to use linear penalties for the constraints C1 than the constraints C2 for the reasons we have just mentioned. This is why we only consider applying the linear penalty method to the constraints C1 in this thesis. To make the most of the linear penalty method's ability to reduce the number of required couplings, it should be applied to constraints acting on variables that are sparsely connected in the logical graph of the objective function.

An example problem where we do not expect the linear penalty method to always be effective is the knapsack problem, which has a linear objective function and a single linear inequality constraint. This constraint differs from the constraints C1 in that it is not a function of the Hamming weight and is not an equality. If a linear penalty could implement this constraint, the penalised objective function would be trivial to solve as it would be entirely linear. Assuming that the linear penalty strength could be tuned in polynomial time, this would render the NP-hard problem trivial and thus show that P = NP. This result would be unexpected, so it is more likely that the linear penalty method is not always effective for this problem or the penalty strength cannot always be tuned in polynomial time. We note that typical knapsack problem instances are known to be easy [141], so it is possible that the linear penalty method works well for some instances, and potentially most instances, without requiring that P = NP. In fact, creating hard knapsack problem instances is an active area of research [142].

3.4 Constraint dependence on linear penalty strength

While both the quadratic and linear penalty methods introduce a single penalty strength parameter per constraint, the two methods exhibit different behaviours

as their penalty strengths are changed. In this section, we study their behaviours in the context of the 1Q-PCP, which has a single constraint (Eq. (2.41)). For the quadratic method, if the penalty strength α_2 is too small, it will not be successful in producing a feasible ground state in $\mathcal{H}_I$. This is because infeasible states that have low objective values are not given a large enough energy penalty to leave the ground state manifold. On the other hand, if α_2 is much larger than necessary, $\mathcal{H}_I$ will have a feasible ground state, but the performance of some quantum algorithms in finding the ground state will be hindered. A physical reason for this is that the energy scales associated with the penalty function increase with α_2, and this effectively reduces all other energy scales in $\mathcal{H}_I$ after normalisation. Fig. 3.2 shows an example of this behaviour in simulations of closed-system QA. The feasible probability P_F increases with α_2 and remains elevated at large values, whereas the success probability P_S rises, peaks, and then falls. For a given problem instance, there is an optimal value of α_2 that produces the maximum success probability $P_S^{\max}$ of measuring an optimal solution. There is typically a broad range of values of α_2 that produce good performance, which can be seen in Fig. 3.2(a).

For the linear penalty method, the penalty strength α_1 directly relates to the value of the constraint that is implemented, i.e. the value of A in Eq. (2.41). As shown in Fig. 3.3(a), decreasing α_1 increases the Hamming weight w of the ground state of $\mathcal{H}_I$. All instances of Ising problems that have a linear penalty of the form in Eq. (3.2) applied exhibit this monotonic relation between α_1 and w. This is because negative local fields reduce the energy of states in proportion to their Hamming weights. For a desired value of A, the linear penalty strength α_1 should be tuned such that $w = A$. For each problem instance, there will be different upper and lower limits on the range of values of α_1 that accomplish this. Sometimes, this range is small. This can be seen in the short lengths of some of the bars in Fig. 3.3(a). In comparison, when using the quadratic penalty method, there is only a lower limit on α_2 values that produce a feasible ground state in $\mathcal{H}_I$. For some problem instances, there does not exist any value of α_1 that produces the desired ground

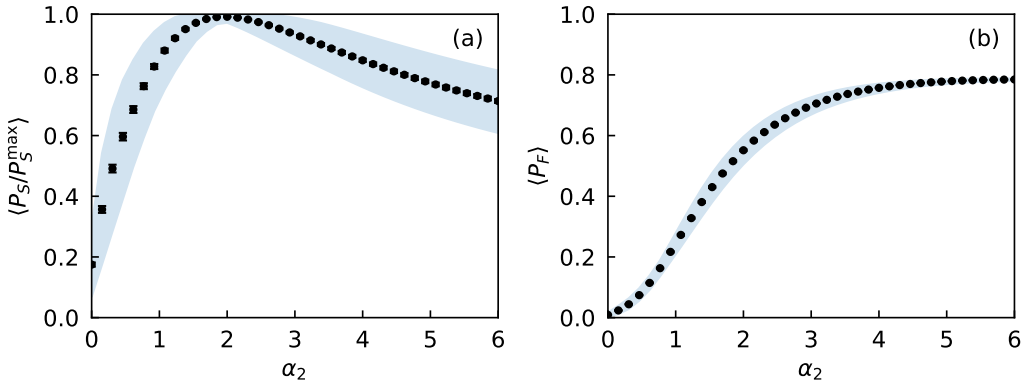


Figure 3.2: (a) QA success probability P_S after an anneal with duration $t_f = 10$ normalised by an estimate of the maximum success probability $P_S^{\max}$ as a function of the penalty parameter α_2. Results come from simulations of closed-system quantum dynamics. Each point is an average over 100 instances of the 1Q-PCP with $n_p = 12$ products. Error bars represent the standard error in the mean, and the blue shaded region contains the 5th to 95th percentile values. Note that $P_S^{\max}$ is estimated for each instance separately by taking the maximum P_S value over the plotted values of α_2. Hence, $P_S^{\max}$ is different for each instance. (b) Same as (a) with the y-axis showing the probability P_F of measuring a feasible state instead. Figure adapted from [1].

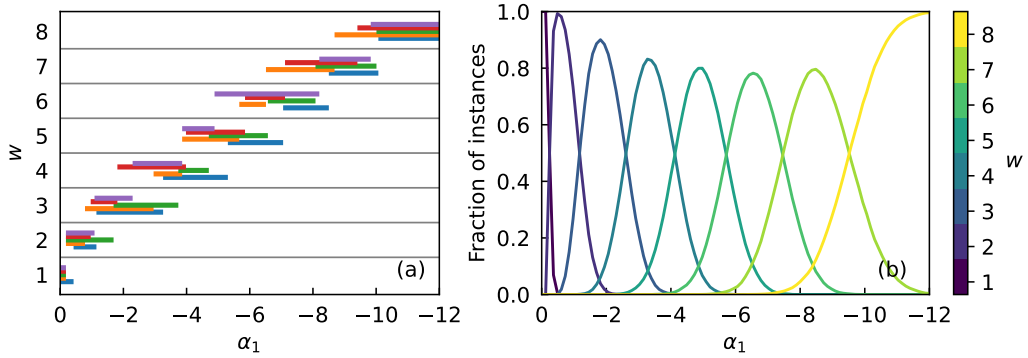


Figure 3.3: (a) A plot of the Hamming weight w of the ground state of $\mathcal{H}_I$ against the linear penalty strength α_1 for five random instances of the 1Q-PCP with eight products. Each instance is shown in a different colour. (b) The fraction of 10,000 1Q-PCP instances that have a ground state Hamming weight w plotted against α_1 for different values of w shown in different colours. Note that α_1 is decreasing along the x-axes. Figure adapted from [1].

state in $\mathcal{H}_I$ with the linear penalty method, which is not the case for the quadratic method.

When choosing α_1, a particular value might be able to implement the desired constraint in the ground state of $\mathcal{H}_I$ for many instances of a given problem, but

it is typically not possible to select a single value that works for all instances, as can be seen in Fig. 3.3 for the 1Q-PCP. Therefore, a good value of α_1 needs to be found for each instance individually. The monotonic relationship between w and α_1 means that a good value of α_1 can be found efficiently by iteratively decreasing (increasing) α_1 when sampled solutions have too few (many) ones. This strategy has previously been suggested in [24; 132].

For the 1Q-PCP, we know that α_1 must be negative to have any products on promotion as the unconstrained objective function only contains positive terms. A good search strategy is to start with some initial guess, such as $\alpha_1 = -1$, run the optimiser, and double $\alpha_1 = -1$ until the returned solution has the correct Hamming weight $w = A$ or one that is too large. If w is larger than A, a binary search can be performed to find a value of α_1 that produces the desired Hamming weight. For an optimiser that always returns the optimal solution, this method finds α_1 in a number of calls to the optimiser that scales logarithmically with the value of α_1 that correctly implements the constraint and is closest to the initial guess. The condition that makes this true is the monotonic relationship between α_1 and w. When only a finite number of optimiser calls are allowed, there is a possibility that the search terminates before it finds a correct value of α_1 if the interval of correct values of α_1 is smaller than the precision of the search. The search can be made exponentially more precise by increasing the maximum number of iterations. We give an example of an algorithm that iteratively searches for good values of α_1 for problems with one or more linear penalties in Sec. 3.6. The iterative nature of the application of linear penalties makes the method a hybrid quantum-classical technique [143].

3.5 Applying linear Ising penalties to multiple constraints

It is common for applied optimisation problems to have many constraints. Therefore, it is important to consider the application of multiple linear penalties and understand how penalty functions interact with each other. Here, we study this using the 2Q-PCP. We set the seasonal scale factors in Eq. (2.37) to $\boldsymbol{\lambda} = (1.5, 1.0)^{\mathsf{T}}$ for the problem instances we consider. We use w_1 and w_2 to denote the Hamming weights of the variables associated with the first and second fiscal quarters, respectively, in the ground state of $\mathcal{H}_I$. These correspond to the number of promotions in each quarter. The constraints in Eq. (2.38) require that $w_1 = w_2 = A$.

We first consider the case where linear penalties are used for both constraints in Eq. (2.38), where the penalty strengths for the constraints on the first and second quarters are denoted as $\alpha_1^{(1)}$ and $\alpha_1^{(2)}$, respectively. Fig. 3.4 shows how w_1 and w_2 change with $\alpha_1^{(1)}$ and $\alpha_1^{(2)}$ for two different instances of the two-quarter problem. For every possible value of A that allows Eq. (2.39) to be satisfied, there is a region in Fig. 3.4(a) where the constraint $w_1 = w_2 = A$ is satisfied. However, these regions cannot always be found by applying the search strategy outlined in Sec. 3.4 for the single-quarter problem to $\alpha_1^{(1)}$ and $\alpha_1^{(2)}$ independently. This is because changes made to $\alpha_1^{(1)}$ impact both w_1 and w_2, and likewise for $\alpha_1^{(2)}$. Therefore, $\alpha_1^{(1)}$ and $\alpha_1^{(2)}$ cannot be found independently of each other. The monotonic relationship between $\alpha_1^{(i)}$ and w_i can still be used to develop a search strategy that finds all penalty strengths at the same time, such as the one outlined in Sec. 3.6.

In some cases, it is not feasible to simultaneously implement multiple constraints with linear penalties. This could be either because it is not possible or because the region of good penalty strength values is too small. It is often possible to resolve this issue by changing some of the linear penalty functions to quadratic penalties. Depending on which penalties are made quadratic, some of the advantages of using

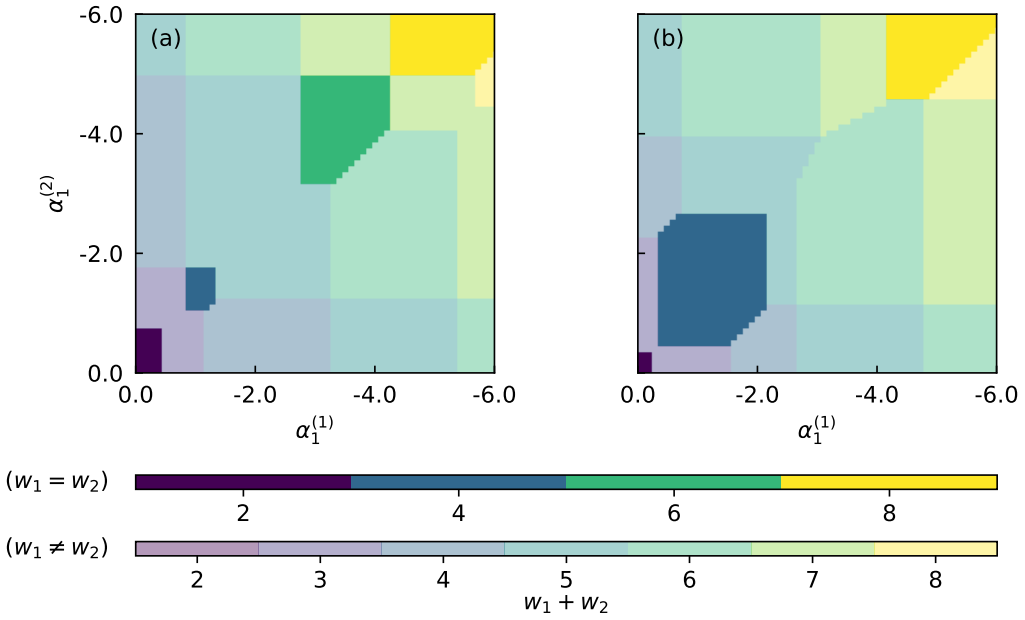


Figure 3.4: Heat maps showing the sum of the first-quarter Hamming weight w_1 and second-quarter Hamming weight w_2 for instances of the 2Q-PCP with eight products. Linear penalties are applied to the first and second quarters with penalty strengths $\alpha_1^{(1)}$ and $\alpha_1^{(2)}$ respectively. The penalty strengths are swept along the x and y axes. This is shown for (a) the instance with ID `8_0` and (b) the instance with ID `8_19`. A colour bar with more saturated colours is used where $w_1 = w_2$ and a less saturated colour bar is used where $w_1 \neq w_2$. Note that $\alpha_1^{(1)}$ and $\alpha_1^{(2)}$ decrease along the axes of both plots. Figure adapted from [1].

the linear penalty method can be maintained with this approach. An example where this can be used is the instance shown in Fig. 3.4(b), for which we did not find any combination of $\alpha_1^{(1)}$ and $\alpha_1^{(2)}$ values that satisfies both constraints in Eq. (2.38) with $A = 3$. Since our search was conducted at a finite precision, this implies that the region in which the constraints are satisfied either does not exist or is very small.

In Fig. 3.5, we consider the same problem instance as in Fig. 3.4(b) and instead apply a linear penalty to the first quarter only and a quadratic penalty to the second quarter. With this scheme, we find that there exists an interval of values of $\alpha_1^{(1)}$ for which all of the problem's constraints are satisfied. In Sec. 5.7.2, there is a more in-depth analysis of combining linear and quadratic penalties in the context of the 4Q-PCP. An alternative approach for cases where linear penalties are not

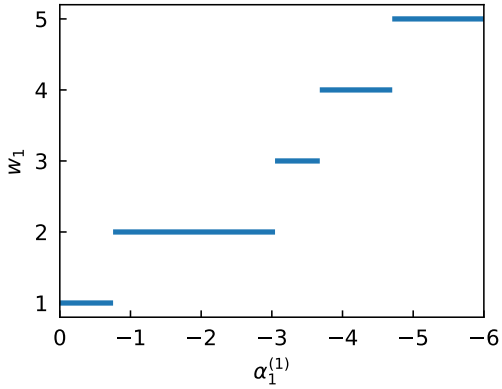


Figure 3.5: Ground state Hamming weight w_1 of the first quarter of a 2Q-PCP instance with ID `8_19` plotted against the linear penalty strength $\alpha_1^{(1)}$ applied to the first quarter. The second quarter's Hamming weight is fixed to $w_2 = 2$ with a quadratic penalty. For this problem instance, we did not find any combination of linear penalty strengths for which applying linear penalties to both quarters is successful in creating a ground state in $\mathcal{H}_I$ that satisfies $w_1 = w_2 = 3$. Note that $\alpha_1^{(1)}$ decreases along the x-axis. Figure adapted from [1].

successful in implementing all constraints is proposed in [139], where infeasible solutions are post-processed to obtain feasible solutions.

3.6 Penalty strength search strategy for multiple linear Ising penalties

When there are multiple constraints that are being implemented with linear penalties, the variables in the different constraint can interact with each other, making the search for α_1 values more difficult. This is because changing the penalty strength for a constraint does not just change the assignment of the variables within the particular constraint, but it can also change the assignment of other variables through their couplings. Therefore, the linear penalty strengths cannot be tuned individually and a strategy to search for all penalty strengths at the same time is required. We are not sure whether a search strategy that can efficiently find good linear penalty strengths up to a desired precision exists for the 4Q-PCP. However, we suspect that the fact that the problem has nonnegative couplings makes it likely

that such a strategy exists, as in the case of the 1Q-PCP.

As an example of a linear penalty strength search strategy for problems with many constraints, we outline one such algorithm for the 4Q-PCP in Algorithm 1. An implementation of this algorithm in the Python programming language is used in some of the analysis in Chapter 5. The function `linear_penalty_strength_search` attempts to find values of α_1 such that Eq. (2.34) is satisfied for each fiscal quarter for some constraint value A. The algorithm makes use of an exact solver that is called through the `optimiser` function, which returns an optimal solution of a problem with the provided linear penalty strengths. If `linear_penalty_strength_search` is successful in finding good values of α_1 within the maximum number of iterations specified by the parameters $max\_iterations\_1$ and $max\_iterations\_2$, it returns these values. Otherwise, it returns $Null$. In principle, one can use a search strategy that has a similar structure with solvers that are not exact, such as quantum annealing and simulated annealing.

The function `linear_penalty_strength_search` performs two searches. The first search, which is performed in lines 10–34 in Algorithm 1, attempts to satisfy the four constraints using the same value of α_1 for each constraint and tuning it such that the average Hamming weight of the variables of the four quarters is equal to A. The first search performs a maximum of $max\_num\_iterations\_1$ iterations. In each iteration, α_1 is increased if the average Hamming weight is larger than A or decreased if it is less than A. Initially, the amount by which α_1 is increased or decreased by is given by a constant step size *step* until an upper and lower limit on the value of α_1 is established. After that, a binary search is used to converge on a value of α_1 for which the average constraint value is equal to A in the solution returned by the solver. Sometimes, this first search finds a value of α_1 that not only produces an average constraint value of A, but also satisfies the individual constraint of each quarter. In these cases, the search is complete.

In cases where the first search does not produce a solution that satisfies all four constraints, a second search is performed. This corresponds to lines 38–60 of Al-

Algorithm 1 Four-quarter linear penalty strength search.

```
function LINEAR_PENALTY_STRENGTH_SEARCH(A, max_iterations_1, max_iterations_2)
    target_num_ones ← [A, A, A, A]                ▷ Target Hamming weight for each fiscal quarter
    pen_strengths ← [−1, −1, −1, −1]              ▷ Initial linear penalty strengths for each quarter
    prev_pen_strengths ← [0, 0, 0, 0]             ▷ Array to store previous penalty strengths
    step ← 0.5                                    ▷ Penalty strength step size
    steps ← [step, step, step, step]
    num_iterations ← 0
    average_converged ← False                     ▷ Flag for convergence of the average Hamming weight
    high, low ← Null, Null                        ▷ Variables to store penalty strengths that are too large or small
    while (not average_converged) and num_iterations < max_iterations_1 do
        num_iterations ← num_iterations + 1
        solution ← optimiser(pen_strengths)       ▷ Call solver
        quarters_num_ones ← get_num_ones(solution)   ▷ Get an array of Hamming weights
        av_num_ones ← mean(quarters_num_ones)     ▷ Average over the quarters' Hamming weights
        if av_num_ones < A then                   ▷ If the average Hamming weight is too small
            high ← pen_strengths[0]               ▷ Penalty strength was too high
            if low = Null then                    ▷ If a penalty strength that is too low hasn't been found yet
                prev_pen_strengths ← pen_strengths
                pen_strengths ← pen_strengths − steps   ▷ Reduce the penalty strength by step
            else                                  ▷ Otherwise, perform a binary search
                prev_pen_strengths ← pen_strengths
                pen_strengths ← 0.5 × (low + high)
        else if av_num_ones > A then              ▷ If the average Hamming weight is too large
            low ← pen_strengths[0]                ▷ Penalty strength was too low
            if high = Null then                   ▷ If a penalty strength that is too high hasn't been found yet
                prev_pen_strengths ← pen_strengths
                pen_strengths ← pen_strengths + steps   ▷ Increase the penalty strength by step
            else                                  ▷ Otherwise, perform a binary search
                prev_pen_strengths ← pen_strengths
                pen_strengths ← 0.5 × (low + high)
        else
            average_converged ← True
    if quarters_num_ones = target_num_ones then
        return pen_strengths                      ▷ If all quarters' Hamming weights converged to A, we are done
    num_iterations ← 0
    converged_all = False                         ▷ Flag for convergence of all quarters' Hamming weights
    step ← 0.1 × step                             ▷ Reduce the step size before the second search
    while (not converged_all) and num_iterations < max_iterations_2 do
        num_iterations ← num_iterations + 1
        solution ← optimiser(pen_strengths)       ▷ Call solver
        quarters_num_ones ← get_num_ones(solution)   ▷ Get an array of Hamming weights
        converged_all = True
        for q ← from 0 to 3 do                    ▷ For each fiscal quarter
            if quarters_num_ones[q] < A then      ▷ If the quarter's Hamming weight is too low
                converged_all ← False
                if pen_strengths[q] > prev_pen_strengths[q] then   ▷ If we stepped up
                    new_guess ← 0.5 × (prev_pen_strengths[q] + pen_strengths[q])   ▷ Take midpoint
                else                              ▷ If we stepped down
                    new_guess ← pen_strengths[q] − step   ▷ Take another step down
                prev_pen_strengths[q] ← pen_strengths[q]
                pen_strengths[q] ← new_guess
            else if quarters_num_ones[q] > A then   ▷ If the quarter's Hamming weight is too high
                converged_all ← False
                if pen_strengths[q] < prev_pen_strengths[q] then   ▷ If we stepped down
                    new_guess ← 0.5 × (prev_pen_strengths[q] + pen_strengths[q])   ▷ Take midpoint
                else                              ▷ If we stepped up
                    new_guess ← pen_strengths[q] + step   ▷ Take another step up
                prev_pen_strengths[q] ← pen_strengths[q]
                pen_strengths[q] ← new_guess
        step = step × 0.93                        ▷ Reduce the step size in each iteration
    if not converged_all then
        return Null                               ▷ Return null if the search failed
    return pen_strengths                          ▷ Return the penalty strengths if the search was successful
```

gorithm 1. The second search performs a maximum of $max\_iterations\_2$ iterations with a step size that is initialised to a smaller value than the step size for the first search. The main difference is that in the second search, each quarter has its own value of α_1 that is tuned individually. These values are increased or decreased based on the Hamming weight of the variables associated with each particular quarter. If the direction in which α_1 is being changed is opposite to the direction of the previous iteration, then α_1 is set to the midpoint of the previous and current values. Otherwise, α_1 is changed by an amount equal to the step size *step*. At the end of each iteration, the value of *step* is reduced such that the search becomes increasingly more precise.

In our runs of this search strategy for the results presented in Chapter 5, we used the parameter values $max\_num\_iterations\_1 = 20$ and $max\_num\_iterations\_2 = 100$. However, the average number of total iterations performed by the search strategy was ≈ 13 for the problem instances that we were able to find good α_1 values for. This implies that much smaller values of $max\_num\_iterations\_1$ and $max\_num\_iterations\_2$ could be used for the 4Q-PCP considered in Chapter 5. Furthermore, the number of iterations required could be reduced by optimising the values of *step* on line 5 of Algorithm 1 and the multiplicative factors on lines 37 and 60. We did not spend much time experimenting with different values.

In parts of Chapter 5, we consider penalty schemes where some of the constraints C1 are implemented with linear penalties and others with quadratic penalties. The α_1 search strategy in Algorithm 1 was used in these cases by modifying the function `optimiser` such that it always satisfies the constraints that have quadratic penalties applied[1]. This is because we know that the quarters with quadratic penalties applied always have the correct Hamming weight in the optimal solution as along as α_2 is chosen to be large enough. The function `linear_penalty_strength_search`

[1]This can be done by either formulating the problem with quadratic penalties that have large penalty strengths or by directly specifying these constraints to the classical solver if it is capable of constrained optimisation. In our case, we used Gurobi Optimizer as the classical solver, and we directly specified the constraints. This approach results in much faster solution times than using the quadratic penalty method to encode constraints into the objective function.

returns linear penalty strengths even for quarters that have quadratic penalties applied, which can be ignored.

Fig. 3.6 shows a demonstration of Algorithm 1 applied to a random instance of the 4Q-PCP with $n_p = 10$ products. We set the seasonal scale factors in Eq. (2.33) to $\boldsymbol{\lambda} = (1.5, 1.0, 1.0, 1.5)^\mathsf{T}$. In this example, the constraints C1 have a target Hamming weight of $A = 4$ for each fiscal quarter. We use Gurobi to find the optimal solution in each iteration and ensure that the constraints C2 and C3 are met. In a search strategy using QA to sample solutions, the constraints C2 and C3 would be implemented with quadratic penalties. We set $max\_num\_iterations\_1 = 20$ and $max\_num\_iterations\_2 = 100$, though neither of these limits were reached in this run. The first stage of the search took 6 iterations, which produced an average Hamming weight of A without satisfying the constraints for the first and fourth quarters. The second stage of the search took 11 more iterations for all of the quarters' Hamming weights to converge to A. We can see that in the final iteration, the values of α_1 for the first and fourth quarters are the same and similarly for the second and third quarters. This is because of the symmetry in the problem due to those pairs of quarters having the same seasonal scale factors.

3.7 Chapter conclusions

In this chapter, we have investigated the linear Ising penalty method for encoding Hamming weight equality constraints and made two theoretical arguments for why the method can lead to better performance in quantum optimisation than the quadratic penalty method. The first argument is based on the fact that the linear penalty method does not introduce any additional quadratic terms to the objective function, and the second is based on considerations of the different energy scales associated with the two types of penalty functions. The linear penalty method is not always successful in exactly implementing a desired constraint in the ground state of the Ising Hamiltonian. However, we have identified a structure in the

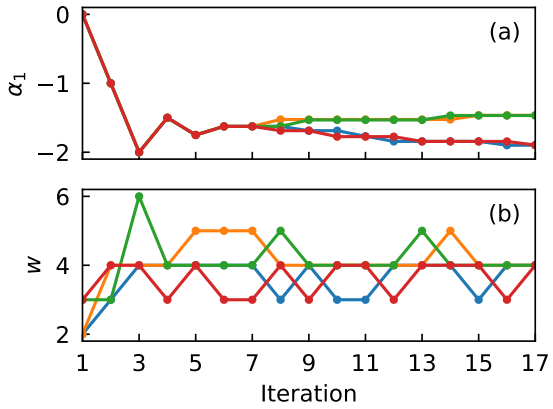


Figure 3.6: Demonstration of the linear penalty strength search strategy in Algorithm 1 on a random instance of the 4Q-PCP with $n_p = 10$ products and seasonal scale factors $\boldsymbol{\lambda} = (1.5, 1.0, 1.0, 1.5)^\mathsf{T}$. The target Hamming weight for each fiscal quarter is $A = 4$. We plot (a) the linear penalty strength α_1 for each quarter and (b) the Hamming weight w of each quarter's variables in the optimal solution against the iteration of the search strategy. The points for the first (blue), second (orange), third (green), and fourth (red) quarters each have lines joining them to guide the eye. We use Gurobi to get the optimal solution $\mathbf{x}$ with a particular set of penalty strengths and specify the constraints C2 and C3 so they are always satisfied. We use the constraint values $B_{\text{min}} = 1$ and $B_{\text{max}} = 2$ for the constraints C2.

promotion cannibalisation problems we have considered that results in the linear penalty method being effective more often. Furthermore, we provided some guidance for determining whether the linear penalty method is appropriate for a given problem. There may be other structures that make problems more amenable to the linear penalty method, which would be interesting to explore in future work.

We have studied how changes in the penalty strength of a linear penalty change the constraint value that is implemented. We compared this with the behaviour of the quadratic penalty method, where all penalty strengths above a certain value implement the desired constraint. Our results show that the penalty strengths of linear penalties typically need to be tuned for different problem instances individually. For problems with a single linear penalty applied, we have outlined a strategy that can efficiently find good penalty strengths by exploiting their monotonic relationship with Hamming weight. We presented another search strategy for problems that have multiple linear penalties applied, in which case the different linear pen-

alties can influence each other, making the search for good penalty strength values more difficult. For cases where one or more constraints cannot be successfully implemented with linear penalties, we have shown that it is sometimes possible to fix this by using linear penalties for some constraints and quadratic penalties for others.

The work presented in this chapter forms a basis for Chapter 4, where the quadratic and linear penalty methods are compared in simulations of quantum algorithms, and Chapter 5, where quantum annealing experiments are performed with the two penalty methods. We also consider how the linear penalty method can be combined with another algorithmic technique for solving constrained optimisation problems on quantum computers in Chapter 6.

Chapter 4

Simulating quantum optimisation with linear Ising penalties

The majority of the results in this chapter have been published in [1].

4.1 Introduction

In this chapter, we perform a numerical analysis of QA and the QAOA solving the 1Q-PCP, comparing the use of quadratic and linear penalty functions. This analysis pursues two objectives: first, to understand the behaviour of linear Ising penalties, and second, to assess their impact on the performance of a quantum optimiser. All simulations in this thesis are of error-free closed-system dynamics and assume all-to-all connectivity of the physical qubits.

Real-world promotion cannibalisation problems involve sparse C matrices, which produce Ising Hamiltonians $\mathcal{H}_I$ with fewer nonzero couplings when using the linear penalty method than when using the quadratic penalty method, as discussed in Chapter 3. For example, a real-world problem may be concerned with 1,000 products that each have nonzero cannibalisation interactions with five other products on average. Switching from using the quadratic penalty method to the linear penalty method would then result in a $\approx 99.5\%$ reduction in the number of nonzero couplings in $\mathcal{H}_I$. This is because the linear penalty method maintains the sparsity

of the matrix C in the objective function, whereas the quadratic penalty method introduces nonzero couplings between all pairs of variables in the 1Q-PCP. Due to limitations on the sizes of problems we can simulate, all C matrices used in this chapter are fully connected. Therefore, our numerical results do not reflect any potential performance benefits for the linear penalty method that would come from the sparsity of C matrices at large problem sizes. The findings in this chapter are complementary to the experimental results in Chapter 5, where more substantial improvements are observed when solving larger promotion cannibalisation problems on a real quantum device.

4.2 Tuning penalty strengths

In this chapter's simulations comparing the quadratic and linear penalty methods, we have taken different approaches to tuning the penalty strengths α_1 and α_2. This is to reflect the different strategies that would likely be used in practice. For the quadratic penalty method, we have used the same penalty strengths for all problem instances. We have tuned this value to produce high success and feasible probabilities on average. The linear penalty method is more sensitive to its penalty strength, and using the same value of α_1 for all problem instances results in some problem Hamiltonians that have infeasible ground states, as discussed in Chapter 3. Therefore, we select the linear penalty strength for each problem instance individually. This additional tuning means that there is an overhead when using the linear penalty method that is associated with the α_1 search strategy that is used, such as the one outlined in Sec. 3.6. This overhead should be taken into consideration when interpreting the results of this chapter.

In Fig. 4.1, we plot the average normalised success probability and average feasible probability of simulations of QA as a function of penalty strength for three different sizes of the 1Q-PCP. Note that Fig. 4.1(b) and Fig. 4.1(e) show the same plots as in Fig. 3.2. As the number of products n_p is increased, we find that P_S peaks at

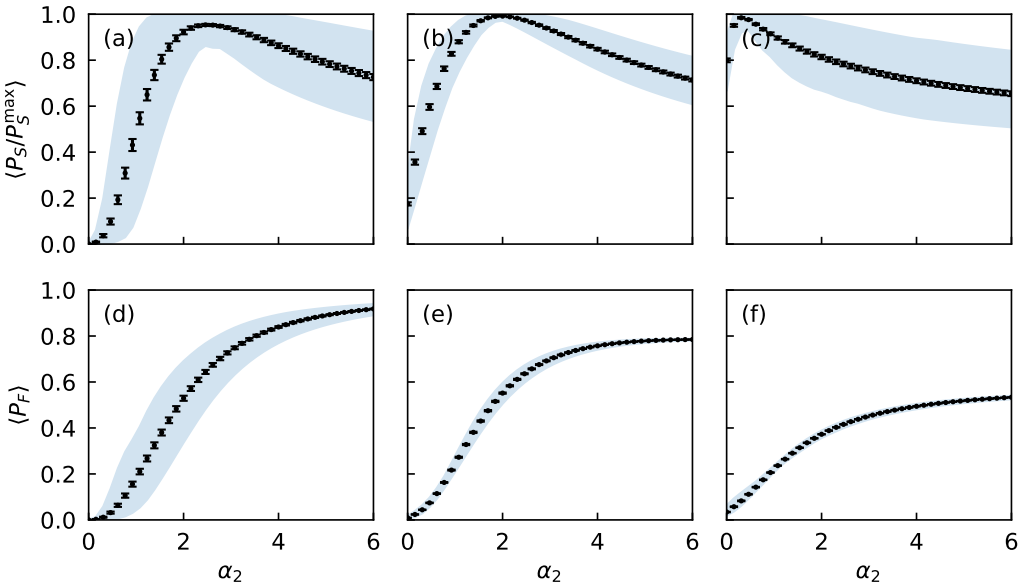


Figure 4.1: (a-c) Plots of the QA success probability P_S after an anneal with duration $t_f = 10$ normalised by an estimate of the maximum success probability $P_S^{\max}$ and averaged over 100 instances of the 1Q-PCP. The value of the penalty strength of the quadratic penalty that is applied is changed along the x-axis. We consider instances with (a) 6, (b) 12, and (c) 18 products. The value of $P_S^{\max}$ is estimated for each instance separately and is calculated by taking the maximum of P_S over the plotted values of α_2. Hence, $P_S^{\max}$ is different for each instance. Error bars represent the standard error in the mean, and the blue shaded regions contain the 5th to 95th percentile values of $P_S/P_S^{\max}$. (d-f) Similar plots of the probability P_F of measuring a feasible state averaged over the same 100 instances with (d) 6, (e) 12, and (f) 18 products. Figure adapted from [1].

smaller values of α_2 on average. The values of α_2 at which P_F becomes elevated are not as significantly affected by changing n_p. There is a gradual drop in P_S as α_2 is increased beyond the value at which P_S is maximised. Based on these results, we have chosen a value of $\alpha_2 = 2$ for our performance analysis because it results in a good compromise between maximising P_S and maximising P_F for most problem instances.

Fig. 3.2 and Fig. 4.1 plot average *normalised* success probabilities. The normalisation was performed to avoid skewing the averages in favour of instances for which the maximum success probability is large. For transparency, we also plot the unnormalised success probabilities for problem instances with $n_p = 12$ products and their average values in Fig. 4.2. Comparing this to Fig. 4.1(b), we find that the

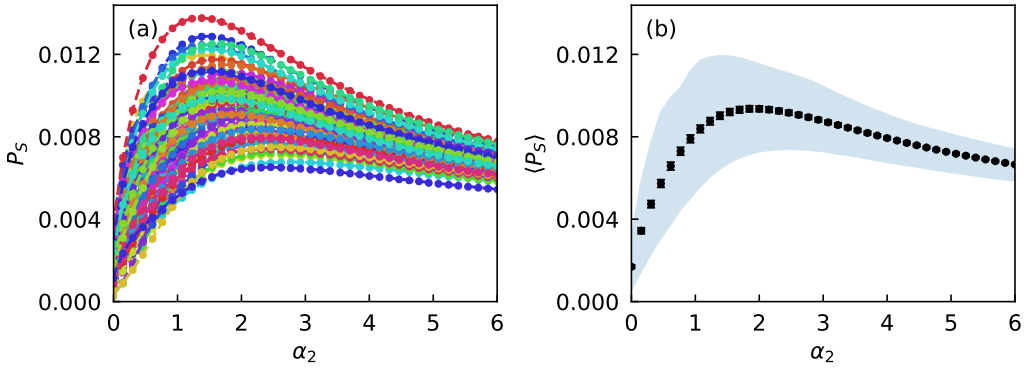


Figure 4.2: (a) Success probability P_S of QA with a duration $t_f = 10$ plotted against the quadratic penalty strength α_2 for the 1Q-PCP with $n_p = 12$ products. This is plotted for 100 instances of the problem that are each represented by different colours, with dashed lines connecting each instance's points to guide the eye. (b) The probability P_S is averaged over the different problem instances. Error bars represent the standard error in the mean, and the blue shaded region contains the 5th to 95th percentile values of P_S. Figure adapted from [1].

average success probability peaks at approximately the same value of α_2 regardless of whether normalisation is performed.

For the linear penalty method, even if α_1 is chosen from the finite interval that implements the desired constraint, the specific choice of α_1 within that interval can have a significant impact on the performance of a quantum algorithm. Fig. 4.3 demonstrates this for an example 1Q-PCP instance. While all choices of α_1 in the yellow and green shaded regions of the plot produce the correct ground state in $\mathcal{H}_I$, some choices result in significantly better success probabilities than others. It is possible for QA to have a significant probability of measuring the optimal feasible solution even when the choice of α_1 does not make the ground state feasible. This is demonstrated by elevated values of P_S outside the shaded regions in Fig. 4.3. This probability will tend to zero as the anneal time is increased to the adiabatic limit and the probability of measuring the infeasible ground state of $\mathcal{H}_I$ approaches unity.

For our performance analysis of the linear penalty method, we use α_1 values that have been selected uniformly at random for each instance from the interval that produces the correct ground state in $\mathcal{H}_I$. This corresponds to taking a random

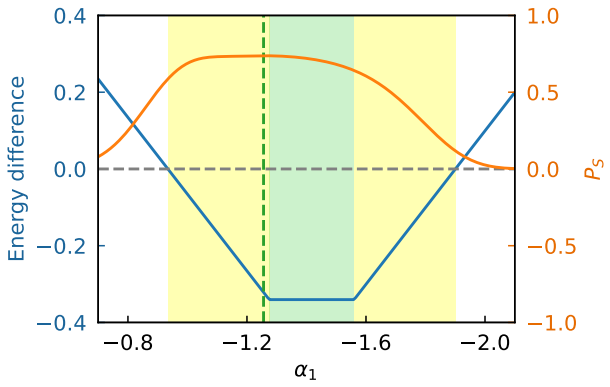


Figure 4.3: A linear penalty is applied to a 1Q-PCP instance with ten products and a constraint of $A = 3$ promotions. In orange, we plot the success probability P_S of measuring the optimal feasible solution $\mathbf{x}^*$ for QA after an anneal of duration $t_f = 200$. The penalty strength α_1 decreases along the x-axis. In blue, we plot the energy difference between the desired state $|\mathbf{x}^*\rangle$ and the minimum energy eigenstate of $\mathcal{H}_I$ that isn't $|\mathbf{x}^*\rangle$. A negative energy difference, highlighted by the yellow and green shaded regions, indicates that $|\mathbf{x}^*\rangle$ is the non-degenerate ground state of $\mathcal{H}_I$. In the green shaded region, $|\mathbf{x}^*\rangle$ is the ground state and the energy separation to the first excited state is maximised. The value of α_1 that maximises P_S is indicated by the green dashed line. Figure adapted from [1].

point along the x-axis in the yellow and green shaded regions in Fig. 4.3. Since in practice the optimal choice of α_1 is not known in advance, we have not performed any further fine-tuning within this region. Therefore, our choices of α_1 roughly correspond to values that would be found by a search strategy such as the one outlined in Sec. 3.6, and they do not represent the theoretically optimal values. For some problem instances, no value of α_1 could be found that produced a feasible ground state of $\mathcal{H}_I$. This is either because the interval of good values is smaller than the precision of our search, which was 10^{-5}, or because the interval does not exist. We did not perform simulations using the linear penalty method for these instances.

4.3 Analysis of the spectral gap in quantum annealing

One useful measure of the difficulty of solving a problem is the minimum spectral gap $g_{\min}$ between the ground state and the first excited stated of the QA Hamilto-

nian $\mathcal{H}$. At small problem sizes, the instantaneous energy spectrum of $\mathcal{H}$ can be calculated for different times during the anneal. As discussed in Sec. 2.3, the total anneal time t_f required to guarantee a large success probability increases as $g_{\min}$ shrinks. For two-level systems, this transition probability was shown to decay exponentially with $g_{\min}^2$ by Landau [144] (according to [56]) and Zener [145]. Hence, larger values of $g_{\min}$ are more desirable in AQC.

Fig. 4.4(a-b) plots the spectral gap g between the ground state and the first excited stated of $\mathcal{H}$ at different times of an anneal for the instance of the 1Q-PCP with ID `12_0`. We compare the use of the linear and quadratic penalty methods with different values of their penalty strengths. In all cases, $\mathcal{H}_I$ is normalised using Eq. (2.48) such that $\max_{i,j}(|J_{i,j}|) = \max_i(|h_i|) = 10$ after normalisation. The value of 10 was chosen so that the size of the spectral gap at the beginning and end of the anneal are of similar magnitude. For both penalty methods, we can see that the size of the minimum spectral gap $g_{\min}$ and its location along the anneal changes as the penalty strength α_2 or α_1 is changed. For good choices of α_2 and α_1, the energy gap is generally larger when using the linear penalty method than using the quadratic penalty method in this example. This indicates that AQC would perform better with the linear penalty method for this problem instance.

The location of $g_{\min}$ is closer to the beginning of the anneal for the linear penalty method than the quadratic penalty method. One way to improve the performance of QA is to use an annealing schedule that slows down when g is small [54]. Hence, the optimal annealing schedule for the two penalty methods would be different in this case. In our simulations of QA in this chapter, we use a linear annealing schedule, which has a constant annealing speed throughout the anneal. This serves as a fair comparison between the two penalty methods as it does not slow down at any particular point in the anneal. For the QA experiments in Chapter 5, we used the default annealing schedule of the D-Wave quantum annealer we used. An analysis of optimised annealing schedules for different penalty methods is a topic that could be explored in future work.

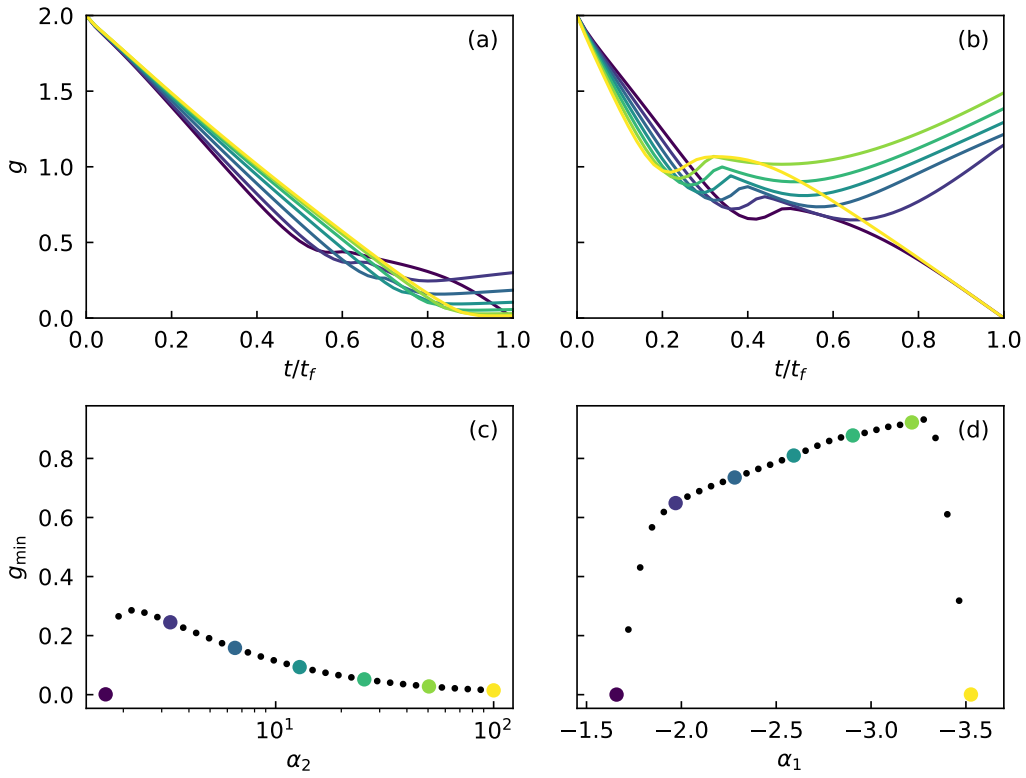


Figure 4.4: Energy gap g between the ground state and first excited state at different times t during an anneal of duration t_f when solving an instance of the 1Q-PCP using the (a) quadratic and (b) linear penalty methods. This problem has $n_p = 12$ products and a constraint of $A = 4$ promotions. Different values of the penalty strength parameters α_2 and α_1 are used, shown in different colours. The minimum value $g_{\min}$ of the energy gap is plotted against the penalty strength for the (c) quadratic and (d) linear penalty methods. Points corresponding to the penalty strengths used in the upper panels are plotted larger and in matching colours. Note that α_1 is decreasing along the x-axis in (d) so that the magnitude of the penalty strengths in (c) and (d) are both increasing along the x-axes. A logarithmic scale is used for the x-axis in (c).

The minimum value of g during the anneal, $g_{\min}$, is plotted in Fig. 4.4(c-d) for different choices of penalty strengths. For the linear penalty method, this is plotted for the range of values of α_1 that produce the correct ground state in $\mathcal{H}_I$. For the quadratic penalty method, there is no upper limit to the value of α_2 that produces the correct ground state. Therefore, the lowest value of α_2 in Fig. 4.4(c) is the lowest value that produces the correct ground state in $\mathcal{H}_I$ and the largest value is arbitrarily chosen to be 100. Considering that Fig. 4.4(c) uses a logarithmic scale for the x-axis, we observe that $g_{\min}$ is elevated for a larger range of penalty strength values with the quadratic penalty method than the linear method. This follows our previous observations that the quadratic penalty method is less sensitive to its penalty strength. We find that there is substantial variation in the size of $g_{\min}$ within the range of values of α_1 that produce the correct ground state in $\mathcal{H}_I$. This shows similar behaviour as the results in Fig. 4.3 for diabatic QA.

4.4 Dynamic range comparison

In our simulations of QA, the Ising Hamiltonians were normalised by a factor $\mathcal{N}$, as expressed in Eq. (2.48), such that the maximum coupling strength is 1 and the maximum local field strength is 3. This reflects the finite energy scales that a physical device can realise. The choice of penalty method can impact the maximum coupling and local field strengths, which impacts the normalisation factor. We refer to the normalisation factor when using the linear and quadratic penalty methods as $\mathcal{N}_L$ and $\mathcal{N}_Q$ respectively. As mentioned in Chapter 3, the linear penalty method uses no couplings and can require weaker local fields than the quadratic method, which benefits the dynamic range of qubit interactions.

In Fig. 4.5, we plot the average ratio between $\mathcal{N}_L$ and $\mathcal{N}_Q$ for the 1Q-PCP instances we consider in this chapter. For all of the problem sizes we have considered, $\mathcal{N}_Q$ is significantly larger than $\mathcal{N}_L$ on average. We believe that the larger effective dynamic range when using the linear penalty method can lead to a more efficient

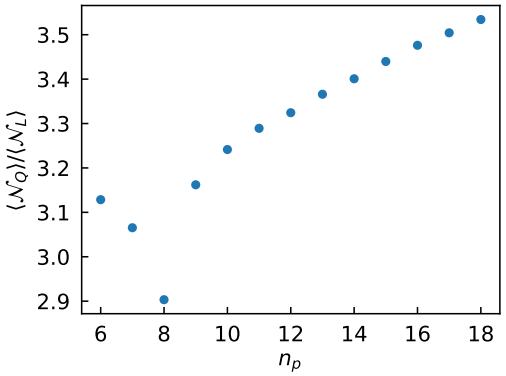


Figure 4.5: Ratio between the average normalisation factors of $\widetilde{\mathcal{H}}_I$ when using the quadratic penalty method, $\langle \mathcal{N}_Q \rangle$, and when using the linear penalty method, $\langle \mathcal{N}_L \rangle$, plotted against the number of products n_p for single-quarter promotion cannibalization problem instances. Figure adapted from [1].

exploration of the search space. The results in Sec. 4.5 and Sec. 4.6 serve as a good test of whether this is true.

4.5 Quantum annealing performance

We have calculated the QA success probability P_S and feasible probability P_F for different instances of the 1Q-PCP using the linear and quadratic penalty methods, which we plot against the number of products n_p in Fig. 4.6. The constraint C1 that we consider in these simulations is to select $A = 3$ products for promotion. The grey dashed line in Fig. 4.6(a) represents the probability of finding the optimal solution by selecting a solution at random. Since these problem instances have unique optimal solutions, the probability of randomly selecting the optimal solution is $\frac{1}{2^{n_p}}$. The success probability for every problem instance is above this line for both penalty methods, indicating that QA is consistently able to increase the overlap of the quantum state with the optimal state.

For the range of number of products we have considered in the QA simulations, Fig. 4.6(a) shows that the linear penalty method outperforms the quadratic penalty method on average in finding the optimal solution. For $n_p \geq 14$ products, there is no overlap between the distributions of the two penalty methods' success prob-

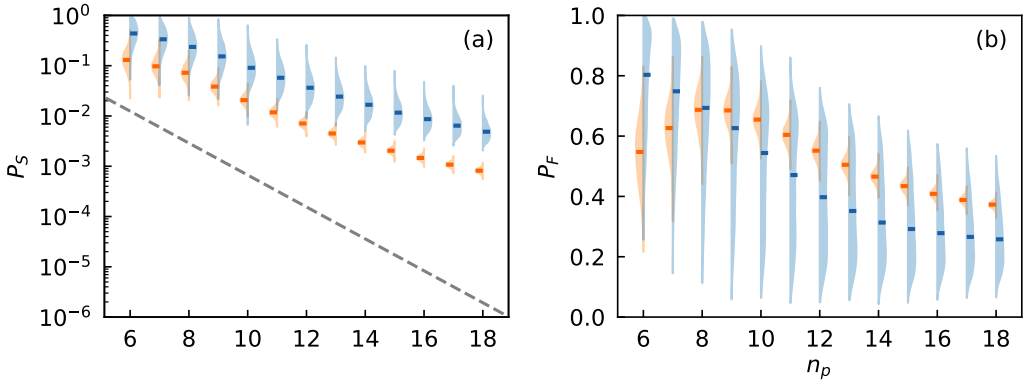


Figure 4.6: (a) The success probability P_S of QA using the quadratic (orange) and linear (blue) penalty methods with an anneal duration of $t_f = 10$ against the number of products n_p in the 1Q-PCP. The plot shows the distributions of P_S for the different problem instances alongside bars representing the median values of the distributions. The grey dashed line represents the probability of randomly guessing the optimal solution. Note that the y-axis scale and the bin widths of the histograms are logarithmic. (b) Similar plot for the probability P_F of measuring a feasible state with linear axes scales and linear histogram bin widths. Figure adapted from [1].

abilities. This indicates that there was not a single instance that was solved with the linear penalty method and could be solved more efficiently with the quadratic penalty method at those problem sizes. As the number of products is increased, the variance in the success probability across different instances shrinks when using the quadratic method but remains large for the linear method.

In Fig. 4.6(b), we find that for $n_p \geq 9$, the quadratic penalty method is on average more likely to sample feasible states. The variance of the distribution of feasible probabilities is much larger when using the linear method than when using the quadratic method. This indicates that with our penalty strength choices, the ability of the linear penalty method to effectively implement the constraint is highly dependent on the problem instance. This explains why the success probability distributions have similarly large variances for the linear method. Given the sensitivity of QA performance to the linear penalty strength in Fig. 4.3, it is possible that P_F could be increased for many instances with further fine-tuning of the penalty strength.

These simulations are of the ideal regime, where there is no noise introduced by the environment and qubits are all-to-all connected. Thus, while the advantage of using fewer couplings with the linear penalty method is more clear when the hardware does not support all-to-all connectivity, these simulation results indicate that the linear penalty method may also benefit performance on fault-tolerant quantum devices with high qubit connectivity. This is complementary to promising experimental results obtained on D-Wave annealers, presented in previous work [132; 136; 139] and in Chapter 5, where hardware limitations have a prominent effect on performance.

4.6 QAOA performance

We have performed simulations of the QAOA with $p = 8$ layers. The same problem instances were used as in the QA simulations in Sec. 4.5. The constraint C1 was set to have the value $A = 3$, which is the same as in the QA simulations. For each problem instance, we have run the QAOA with 80 different initialisations of the circuit parameters $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\gamma})$, which were classically optimised. We use $\boldsymbol{\theta}^*$ to refer to the optimised circuit parameters that produced the largest inferred success probabilities for each problem instance. Our simulations of the QAOA were more computationally demanding than the QA simulations for a given problem size. This is largely because of the variational nature of the QAOA, which requires a large number of quantum circuit evaluations to optimise $\boldsymbol{\theta}$. Due to time and computational resource limitations, we performed QAOA simulations with up to $n_p = 14$ products instead of 18.

In the simulations presented in this section, each of the variational parameters in $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ were initialised randomly within the interval $[0, 1)$. The variational parameters were optimised by the implementation of the COBYLA algorithm [124] in `scipy.optimize.minimize` using a maximum of 100 iterations of the optimiser. To estimate the average objective value while optimising $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, the quantum

circuit was sampled 1,000 times each time the variational parameters were changed. After the final iteration, the quantum circuit was sampled 1,000,000 times with the optimised values of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, from which the probabilities P_S and P_F were inferred. To reduce the effect of getting stuck in local minima when optimizing $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, the entire algorithm was repeated 80 times for each problem instance using different initialisations of the variational parameters, from which a best-performing run could be identified.

Fig. 4.7 plots the inferred success probabilities and inferred feasible probabilities against the number of products in the problem. These are plotted for the best found circuit parameters $\boldsymbol{\theta}^*$ in Fig. 4.7(a) and Fig. 4.7(c) and as an average over the runs with different initialisations of the QAOA angles in Fig. 4.7(b) and Fig. 4.7(d). Note that we use the term "inferred" to mean that these values are estimates of the true probabilities based on a finite number of samples from the quantum circuit. When averaged over all circuit parameter initialisations, the success probability is larger than that of random guessing for more than half of the instances at each problem size, whether the linear or quadratic penalty method is used. However, the average success probability is worse than that of random guessing for some instances, particularly at smaller problem sizes. This indicates that the classical optimiser is able to find circuit parameters that produce better-than-random success probabilities more often than not, despite struggling in some cases. When only considering the best found circuit parameters $\boldsymbol{\theta}^*$, all inferred success probabilities are better than that of random guessing.

For all values of n_p we have considered, the linear penalty method produces more favourable success and feasible probabilities on average compared to the quadratic penalty method. This is true when only considering the best found circuit parameters $\boldsymbol{\theta}^*$ and when averaging over different initialisations. The inferred probabilities of measuring a feasible state with the linear method are typically the same or better than for the quadratic method. Similarly to the results for QA, the larger variances for the linear penalty method in Fig. 4.7(d) indicate that the effectiveness of

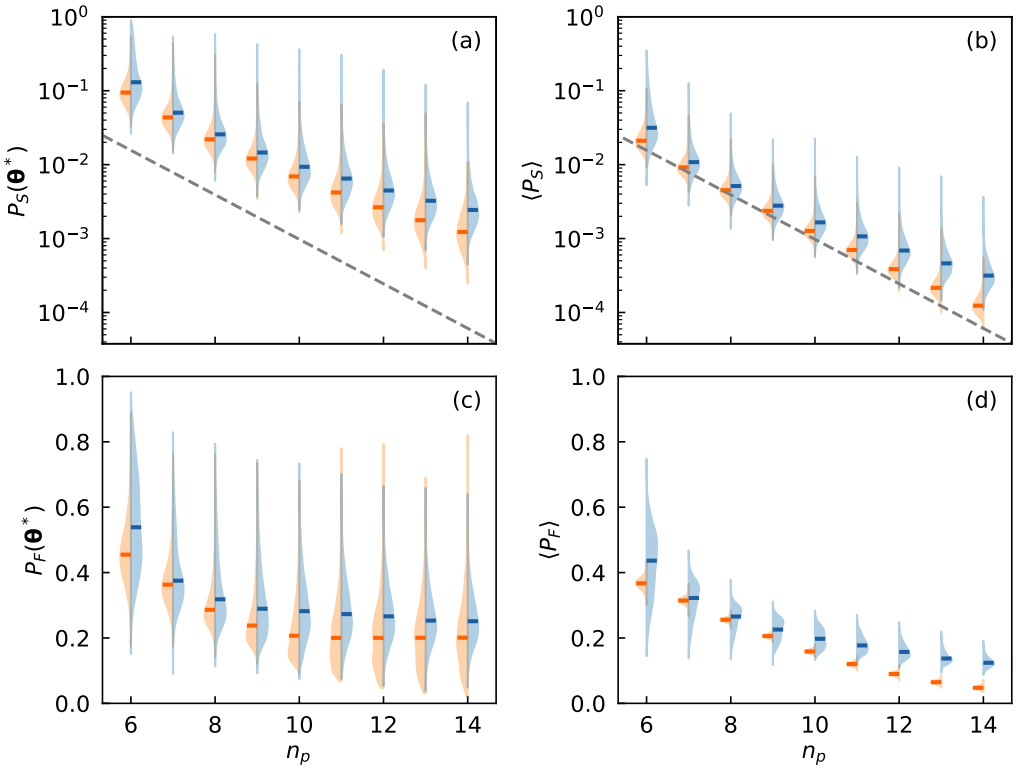


Figure 4.7: (a) Plot of the inferred success probability P_S of the QAOA with $p = 8$ layers and the best found circuit parameters $\boldsymbol{\theta}^*$ after optimising the circuit parameters with 80 different initialisations. This is shown for the quadratic (orange) and linear (blue) penalty methods against the number of products n_p in the single-quarter promotion cannibalization problem. The distributions of $P_S(\boldsymbol{\theta}^*)$ are plotted for the different problem instances alongside bars representing the median values of the distributions. The grey dashed line represents the probability of randomly guessing the optimal solution. Note that the y-axis scale and the bin widths of the histograms are logarithmic. (b) Same as (a) with P_S averaged over all 80 optimised circuit parameters $\boldsymbol{\theta}$ for each problem instance. (c) Similar plot for the inferred probability P_F of measuring a feasible state with the circuit generated by the parameters $\boldsymbol{\theta}^*$ that produced the largest inferred success probability. Axes scales and histogram bin widths are linear. (d) Same as (c) with P_F averaged over all 80 optimised circuit parameters $\boldsymbol{\theta}$ for each problem instance. Figure adapted from [1].

the linear penalty method is more variable across different problem instances and choices of penalty strength than for the quadratic penalty method. Since $\mathcal{H}_I$ does not strictly need to be normalised in the QAOA, one might question whether there is any benefit to having smaller interaction energy scales with the linear penalty method. Our results suggest that the linear penalty method continues to have advantages even in the case where there is no physical dynamic range limitation. We believe that this is because phase rotations of angles larger than 2π have the same effect as those with angles smaller than 2π, so there is a limit beyond which there is no benefit to rotating by larger angles. Hence, an effect similar to the physical dynamic range in QA is produced.

Comparing these results with the QA results, we find that the difference in average success probabilities between the two penalty methods is smaller for the QAOA simulations than the QA simulations. We note that QAOA and QA performance are dependent on the choices of the number of layers p and anneal time t_f respectively, so a fair comparison of the two algorithms would require these two parameters to be chosen in such a way that the effective anneal times are the same. Since the purpose of our analysis is not to compare these two algorithms, we have not done this. An interesting direction for future work would be to determine how the performance difference between the linear and quadratic penalty methods depends on p for the QAOA and t_f for QA.

Another factor that is not explored in our analysis is the time that would be required to execute the QAOA algorithm and how it differs between the two penalty methods. In real-world problems, where C matrices are typically sparse, encoding constraints with the linear penalty method would produce fewer nonzero couplings in $\mathcal{H}_I$. Therefore, the quantum circuit to implement the unitary $\mathrm{e}^{-\mathrm{i}\gamma_k \mathcal{H}_I}$ would require fewer two-qubit gates, making it more efficient to run the circuit with the linear method. Another way in which the runtime could differ between the penalty methods would be if it were easier to classically optimise the circuit parameters $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\gamma})$ using one penalty method than the other, if such a difference exists.

4.7 Chapter conclusions

In this chapter, we have performed a numerical analysis of the linear and quadratic penalty methods. We have studied the behaviour of the penalty methods with respect to changes in their penalty strengths. The linear penalty method is more sensitive to its penalty strength, meaning that it is typically necessary to search for good penalty strength values for each problem instance individually. This results in an overhead compared to using the quadratic penalty method. However, given that quantum optimisation is usually concerned with NP-hard problems that are expected to take exponentially long times to solve, we argue that running the solver more times is often a good trade-off to make for a better chance of sampling high-quality solutions.

Our simulations of QA and the QAOA indicate a performance enhancement when using the linear penalty method over the quadratic method. Given that our simulations assume all-to-all connectivity of the physical qubits and do not simulate any errors, this finding shows that the linear penalty method may be useful for future devices that have low logical error rates and fewer physical limitations than near-term devices. This is encouraging because one of the main theoretical appeals of the method, which is to produce more efficient mappings onto sparse hardware graphs, does not apply in this scenario. The other main appeal of the linear penalty method is that it can make more efficient use of the available dynamic range, which we believe is the reason behind its improved performance in these simulations. We suspect that the relative outperformance of the linear method would be greater at larger problem sizes, where the dynamic range effects are more prominent and the problems may be sparse.

Chapter 5

Quantum annealing experiments with linear Ising penalties

The majority of the results in this chapter have been published in [2].

5.1 Introduction

In recent years, noisy intermediate-scale quantum devices have become available on cloud platforms [127; 17], enabling experimental tests of quantum algorithms. The quantum annealers manufactured by D-Wave are among the most technologically mature quantum devices that are currently accessible, allowing for tests of larger problems than what can be encoded on other quantum devices that are available. D-Wave's Advantage devices have $\approx$ 5,000 physical qubits, which means that they can encode optimisation problems that are larger than those considered in the simulations in Chapter 4. For a review of the industrial applications of QA, see [28].

In this chapter, we conduct experiments on a D-Wave quantum annealer with the linear and quadratic penalty methods. For comparison, we also perform SA runs. We study the 1Q-PCP with $n_p = 100$ products and the 4Q-PCP with $n_p = 10$ products. For the 1Q-PCP, we consider the case where the constraint C1 is to choose $A = 50$ products to promote. While it would be unusual for a retailer to promote half of their available products in a single quarter, we can imagine a

more realistic scenario in which the 100 products in the problem are a selection of products that have been determined to be the most promising candidates for promotion out of a larger set of available products. In this scenario, the quantum optimiser is being used to select from these 100 candidate products. The C matrices we have used for the 1Q-PCP experiments have an average connectivity of ≈ 3.4. Even at this level of sparsity, all constrained problem instances that we have generated have optimal solutions with nonzero total cannibalisation. For the 4Q-PCP experiments, we set the seasonal scale factors in Eq. (2.33) to $\boldsymbol{\lambda} = (1.5, 1.0, 1.0, 1.5)^{\mathsf{T}}$. We set $A = 4$ for the constraints C1, and we set $B_{\min} = 1$ and $B_{\min} = 2$ for the constraints C2. The C matrices for the 4Q-PCP instances have an average connectivity of ≈ 5.1, and all instances have optimal solutions with nonzero total cannibalisation.

5.2 Minor embedding improvements

In Chapter 3, we mentioned that one of the advantages of using the linear penalty method over the quadratic method is that it does not introduce any new couplings to $\mathcal{H}_I$. This allows us to maintain the sparsity of the unconstrained problem. In QA on D-Wave hardware, the physical qubits are not fully connected, so $\mathcal{H}_I$ must be minor embedded onto the working graph of the annealer in order to facilitate the required connectivity between the logical variables. See Sec. 2.4 for a description of minor embedding. By replacing quadratic penalties with linear penalties, $\mathcal{H}_I$ will become more sparse and fewer physical qubits will be required in the minor embedding. One reason why this is desirable is that minor embeddings with shorter chains of physical qubits are less likely to result in chain breaks, which leads to better performance in minimising the energy of $\mathcal{H}_I$ [146; 147]. Another reason is that for a QPU with a constant number of qubits, reducing the size of the minor embedding increases the maximum problem size that can be encoded on the QPU. If the problem size is small enough that multiple problem instances can fit onto the

QPU, smaller minor embeddings allow for more copies of the problem to be encoded and solved in parallel, which saves computation time. We have not attempted this type of parallelisation in our D-Wave experiments.

5.2.1 Single-quarter problem minor embedding analysis

As an example of the improvement in minor embedding efficiency when using linear penalties, Fig. 5.1 shows two minor embeddings that were calculated by `find_embedding`. Both minor embeddings correspond to the logical graph of the 1Q-PCP instance with ID `100_0` embedded onto the working graph of the D-Wave Advantage_system6.3. In Fig. 5.1(b), the linear penalty method was used to encode the problem's constraint, and in Fig. 5.1(c), the quadratic penalty method was used. The minor embedding with the linear penalty method uses 189 physical qubits to represent the 100 logical variables, whereas the minor embedding with the quadratic penalty method uses 1,282 physical qubits. We expect that this substantial reduction in the number of physical qubits from using the linear penalty method should result in better performance of QA on this problem, which we analyse in Sec. 5.5. We note that the example minor embeddings shown in Fig. 5.1(b-c) are for one particular problem instance and a particular choice of the random seed parameter for the `find_embedding` function. Averaged over the minor embeddings we used for all 1,000 problem instances in our D-Wave experiments, $\approx$ 1,290 physical qubits are used with the quadratic penalty method and $\approx$ 157 physical qubits are used with the linear penalty method.

5.2.2 Comparison of minor embedding heuristics for complete graphs

Since the 1Q-PCP has a single constraint that involves all variables in the problem, the corresponding quadratic penalty produces nonzero couplings between all variables. This means that the interaction graph of $\mathcal{H}_I$ is complete (i.e. fully

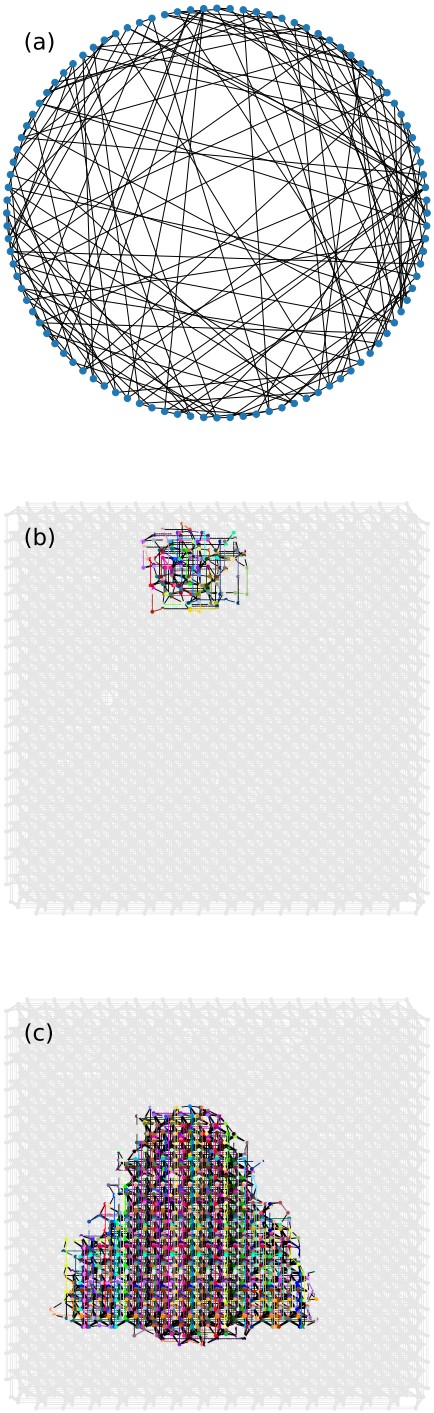


Figure 5.1: (a) Logical graph of a 1Q-PCP instance without any penalty functions applied. Variables are represented by nodes. Pairs of nodes are connected by an edge if there is a nonzero quadratic term for the corresponding pair of variables in the objective function due to a cannibalization interaction between the associated products. (b) An example minor embedding of the problem instance with a linear penalty applied onto the working graph of a D-Wave Advantage QPU. Here, nodes represent physical qubits and edges represent physical couplings. Chains of nodes representing the same logical variable are in the same colour. Qubits and couplings that are not used are coloured in grey. (c) An example minor embedding of the same problem instance with a quadratic penalty applied, resulting in a complete logical graph. Figure adapted from [2].

connected) for the single-quarter problem. D-Wave's Ocean library has a function called `find_clique_embedding` that can minor embed complete graphs onto the hardware graph of a D-Wave Advantage QPU using fewer physical qubits than the heuristic used in `find_embedding`. However, we have not used `find_clique_embedding` in any of the results presented outside this subsection because we have found that the quantum annealer performs slightly better on average when using the `find_embedding` function, even though it produces less efficient minor embeddings. We note that while `find_clique_embedding` is the more efficient heuristic for fully connected problems such as the 1Q-PCP, it is not as efficient as `find_embedding` for the 4Q-PCP, as seen in Sec. 5.2.3.

The reason for the better average performance when using `find_embedding` becomes apparent after overlaying the values of the variables in sampled solutions onto the minor embedding graph. We visualise this for the lowest energy solutions sampled using the `find_clique_embedding` and `find_embedding` heuristics for a particular problem instance in Fig. 5.2. In this figure, chains of qubits are coloured according to the value of the corresponding variable in the solution. In both solutions, roughly half of the chains correspond to variables equal to 0 and half equal to 1. Therefore, the problem's constraint of placing half of the products on promotion is close to being satisfied. Looking at the locations of the variables with the same values, we can see that each colour forms a cluster of qubit chains in such a way that there is a high degree of connectivity between the qubit chains within each cluster and less connectivity between qubits belonging to different clusters. In other words, it appears that qubit chains with more physical couplings between them are more likely to take the same logical value in the solutions shown in Fig. 5.2. However, no such structure exists in the objective function of the problem. Therefore, the choice of mapping of logical variables to physical qubits must be heavily influencing the sampled solutions. We believe this phenomenon is occurring because the chain lengths are long and the problem is too large for the quantum annealer to find low energy states of $\mathcal{H}_I$, so the structure of the minor embedding is playing a significant

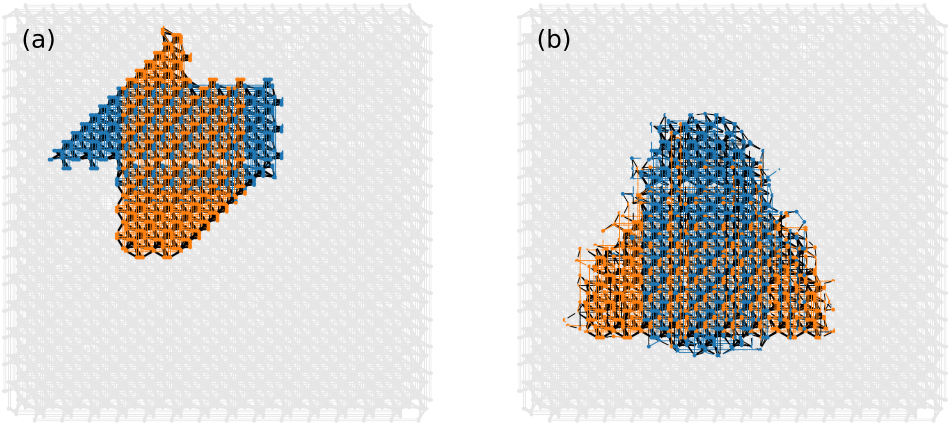


Figure 5.2: (a) A minor embedding produced by the `find_clique_embedding` function, mapping the 1Q-PCP onto the D-Wave Advantage hardware graph. The problem's constraint was encoded with the quadratic penalty method, resulting in a fully connected logical graph. Nodes and edges, representing physical qubits and couplings, form chains that are coloured according to the lowest energy solution sampled by the quantum annealer. Chains of qubits are coloured blue (orange) if the corresponding variable in the solution is equal to 0 (1). Edges representing couplings between two different logical variables are shown in black. Unused qubits and couplings are coloured grey. (b) A similar diagram showing a minor embedding produced by the `find_embedding` function. Note that `find_embedding` produces a different minor embedding depending on the value of its seed parameter, and the example in (b) is for one particular parameter value. Figure adapted from [2].

role in the dynamics of the annealer. Experimenting with different chain strengths did not solve this problem. We note that while the other solutions in the sample sets are not shown in Fig. 5.2, all solutions that were sampled for this problem instance display similar clusterings of variables, with the locations of the clusters being roughly the same but the value (i.e. colour) of each cluster changing between different solutions.

For the problem instance shown in Fig. 5.2, the lowest energy solution that was sampled using `find_embedding` to calculate the minor embedding has a lower energy than the lowest energy solution sampled using `find_clique_embedding`, despite the fact that `find_clique_embedding` produces a minor embedding with fewer physical qubits. This is because the minor embedding produced by `find_embedding` happens to place qubit chains in locations that produce a more

favourable solution when the clustering effect occurs. Since we have changed the value of the seed parameter of `find_embedding` between different problem instances, our results using this function include a variety of minor embeddings, with Fig. 5.2(b) being one example. In comparison, `find_clique_embedding` does not take a seed parameter as an input, so the minor embedding shown in Fig. 5.2(a) is produced for every problem with 100 variables.

5.2.3 Four-quarter problem minor embedding analysis

For the 4Q-PCP, we performed two sets of experiments. In the first set of experiments, we compared the strategy of applying linear penalties to the four constraints C1 against the strategy of using quadratic penalties for all constraints. For the 1,500 problem instances we ran on the D-Wave QPU, the minor embeddings used ≈ 139 physical qubits on average when using linear penalties and ≈ 168 physical qubits on average with the all-quadratic penalty scheme. While this shows an improvement with the linear penalty method, it is not as large of a difference as what we found in Sec. 5.2.1 for the 1Q-PCP experiments. There are two reasons why this is the case. Firstly, the 4Q-PCP includes the additional constraints C2 and C3, which we are implementing with quadratic penalties even in the penalty scheme involving linear penalties. Secondly, the C matrices we generated for these experiments are more dense than those considered in Sec. 5.2.1 because it is not possible to generate non-trivial problem instances with highly sparse C matrices at this problem size. In real-world problems, where a much larger number of products are considered, the C matrices would typically be much more sparse. Therefore, we expect the penalty scheme with linear penalties to result in a bigger improvement in minor embedding efficiency at larger problem sizes. However, we are not able to get good results from the D-Wave annealer at larger problem sizes due to the limitations in its performance, even if the problem is small enough to be embedded onto the QPU.

To quantify how the improvement in minor embedding efficiency from using the lin-

ear penalty method scales with problem size, we have calculated minor embeddings of 4Q-PCP instances onto the working graph of the D-Wave Advantage_system6.3 at different problem sizes. The C matrices we use here have a constant minimum connectivity of 3, which means they become more sparse as the number of products n_p is increased. In Fig. 5.3, we plot the average chain length after minor embedding against n_p. The orange data points are for the penalty scheme where all penalties are quadratic, and the blue data points are for the scheme where the constraints C1 are implemented with linear penalties. The average chain length in Fig. 5.3 is well-approximated by a linear fit for both penalty schemes. This indicates that the average number of physical qubits used, which is the average chain length multiplied by the number of variables $n = 5n_p$, scales quadratically with n_p for both penalty schemes. However, the gradient of the linear fit is much smaller for the scheme with linear penalties than the all-quadratic scheme, indicating a more favourable chain length scaling when linear penalties are used. Specifically, the gradient of the blue line of best fit in Fig. 5.3 is 0.034 ± 0.002 and the gradient of the orange line is 0.258 ± 0.003. This means that the average number of physical qubits used by the minor embedding is equal to $\frac{1}{5}(0.034 \pm 0.002)n^2 + O(n)$ with the linear penalty method and equal to $\frac{1}{5}(0.258 \pm 0.003)n^2 + O(n)$ with the quadratic penalty method.

In the second set of experiments using the 4Q-PCP, we considered the linear-quadratic-quadratic-linear (LQQL) and quadratic-linear-linear-quadratic (QLLQ) penalty schemes, which apply linear penalties to two of the four constraints C1. The LQQL scheme applies linear penalties to the first and last quarters, and the QLLQ scheme applies them to the second and third quarters. We compared these penalty schemes to the all-quadratic penalty scheme. The minor embeddings with the LQQL penalty scheme used ≈ 154.7 physical qubits on average, compared to an average of ≈ 167.8 with the all-quadratic scheme on the same problem instances. With the QLLQ penalty scheme, the minor embeddings used ≈ 155.0 physical qubits on average, compared to an average of ≈ 168.4 with the all-quadratic scheme

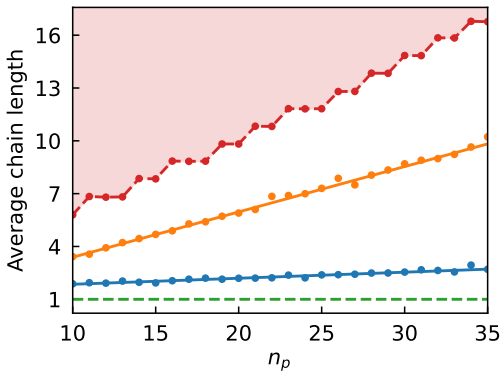


Figure 5.3: Average chain length after minor embedding the four-quarter promotion cannibalization problem onto the D-Wave Advantage hardware graph against the number of products n_p in the problem. We consider the case where the inequality constraints C2 introduce one slack variable per product. Hence, there are $5n_p$ logical variables in the problem. The points are averaged over 10 possible structures of C matrices that are sparse and have a minimum connectivity of 3. We compare the use of quadratic penalties (orange) and linear penalties (blue) for the constraints C1, using the `find_embedding` function to calculate the minor embeddings. These points are given linear fits. We also plot the average chain length after minor embedding with the `find_clique_embedding` function in red, with the red shaded region representing the regime in which switching to this embedding heuristic would reduce the number of physical qubits used. The green line represents the case where the problem can be directly mapped to the hardware, resulting in a chain length of 1. Figure adapted from [2].

on the same problem instances. The reductions in the number of physical qubits using the LQQL and QLLQ schemes are roughly half of what we observed for the scheme with linear penalties for all four constraints C1. This is consistent with the fact that there are half as many variables acted on with linear penalties in the LQQL and QLLQ schemes.

5.3 Dynamic range improvements

Another advantage of using the linear penalty method over the quadratic method is that it can lead to a larger effective dynamic range of qubit interactions, as described in Chapter 3. To make the most of the finite dynamic range available to a quantum annealer, we aim to reduce the strength of the largest magnitude couplings $J_{i,j}$ and local fields h_i in $\mathcal{H}_I$ so that the normalisation factor $\mathcal{N}$ in Eq. (2.48) is

reduced. To determine how significant of an effect the use of linear penalties have on dynamic range in the problems we have performed our experiments on, we have calculated the maximum coupling strength $\max(|J_{i,j}|)$ and maximum local field strength $\max(|h_i|)$ under different penalty schemes.

5.3.1 Single-quarter problem dynamic range analysis

For the 1,000 1Q-PCP instances that we ran on the D-Wave annealer, $\max(|J_{i,j}|)$ is $\approx 2.21\times$ larger and $\max(|h_i|)$ is $\approx 1.26\times$ larger when the quadratic penalty method is used than when the linear method is used. Therefore, switching to using linear penalties is expected to result in more efficient use of the available dynamic range by reducing the value of $\mathcal{N}$ in Eq. (2.48). We suspect that for this experiment, this improvement in effective dynamic range does not have as large an impact on performance as the improvement in minor embedding efficiency discussed in Sec. 5.2.1.

We used a constraint value of $A = 50$ promotions for the single-quarter problem experiments, which is exactly half of the total number of products n_p. It follows from Eq. (2.47) that when $A = \frac{n_p}{2}$, the quadratic penalty for the problem's constraint applies no local fields and only contributes to the couplings. As the value of A is increased or decreased away from $\frac{n_p}{2}$, the strength of the local fields in the quadratic penalty increases. In comparison, the strength of the local fields required for a linear penalty monotonically increases with the value of A. Therefore, the dynamic range improvements from switching to linear penalties may become more significant at other values of A where the local fields become the dominant contribution of the quadratic penalty. A particularly favourable case for the linear penalty method is $A < \frac{n_p}{2}$, where the local fields become weaker for the linear penalty method and stronger for the quadratic penalty method. This happens to be the realistic scenario for promotion cannibalisation problems because retailers typically promote less than half of their products in any given fiscal quarter.

5.3.2 Four-quarter problem dynamic range analysis

In the 4Q-PCP experiments where we compare the penalty scheme involving linear penalties for all constraints C1 to the all-quadratic penalty scheme, $\max(|J_{i,j}|)$ is $\approx 2.66\times$ smaller and $\max(|h_i|)$ is $\approx 2.25\times$ smaller on average for the scheme with linear penalties. This improvement is more significant than what we found for the single-quarter problem experiments in Sec. 5.3.1.

In the second set of 4Q-PCP experiments, where we considered the LQQL and QLLQ penalty schemes, the use of linear penalties for two out of four fiscal quarters does not result in as large improvements in effective dynamic range. With the LQQL penalty scheme, $\max(|J_{i,j}|)$ is $\approx 1.14\times$ smaller and $\max(|h_i|)$ is $\approx 1.13\times$ smaller on average compared to the all-quadratic scheme on the same problem instances. For the QLLQ scheme, there is no change in the value of $\max(|J_{i,j}|)$ or $\max(|h_i|)$ compared to the all-quadratic scheme. Therefore, for the constraints C1, using a combination of linear and quadratic penalties for the set of constraints is better at maintaining the minor embedding advantages of the linear penalty method than the dynamic range advantages. However, there may be scenarios with different types of constraints and objective functions for which this observation does not apply.

In order to understand why there is an improvement in effective dynamic range with the LQQL scheme but not the QLLQ scheme, recall that we have used the seasonal scale factors $(1.5, 1, 1, 1.5)^\mathsf{T}$ in our experiments on the four-quarter problem. This means that the unconstrained problem contributes larger Ising terms to $\mathcal{H}_I$ for the variables associated with Q1 and Q4 than for the variables associated with Q2 and Q3. Therefore, adding a quadratic penalty only to the variables for Q1 or Q4 will produce the same $\max(|J_{i,j}|)$ and $\max(|h_i|)$ values as adding quadratic penalties to all four quarters' variables. It is only possible to reduce $\max(|J_{i,j}|)$ and $\max(|h_i|)$ by omitting the quadratic penalties from both Q1 and Q4 variables, which the LQQL scheme does. Note that this is assuming we use the same quadratic penalty

strength for all four quarters' penalties, which is the case in our experiments.

An important takeaway from this is that when faced with an option as to which constraints to use linear penalty functions for, like in the scenario where we want to choose between the LQQL and QLLQ schemes, it is important to consider the affect that the choice has on the effective dynamic range. If the goal is to get the most dynamic range benefits out of the linear penalty method, the priority should be to use linear penalties for constraints that act on the variables that are limiting the dynamic range. These are the variables with either the strongest couplings or local fields, depending on whether the effective dynamic range is limited by $\max(|J_{i,j}|)$ or $\max(|h_i|)$.

5.4 Tuning penalty strengths

The linear penalty method is more sensitive to the value of the penalty strength than the quadratic penalty method. Therefore, we have used two different strategies for tuning these parameters in our D-Wave experiments. For the quadratic penalty strength α_2, we considered the performance of a SA algorithm at various different values of α_2 and chose the value that gave the best performance on average. For the linear penalty strength α_1, we used a different value for each problem instance. These values were selected uniformly at random from the interval of values that produce a ground state in $\mathcal{H}_I$ that satisfies the constraint. In practice, when the boundaries of this interval cannot be computed, a search strategy such as the one outlined in Sec. 3.4 can be performed to find a value of α_1 in this interval using multiple calls to the optimiser. Although tuning the linear penalty strength requires more calls to the optimiser than for the quadratic penalty method, we argue that this extra computation time is justified if the increase in performance from using the linear penalty method is large enough.

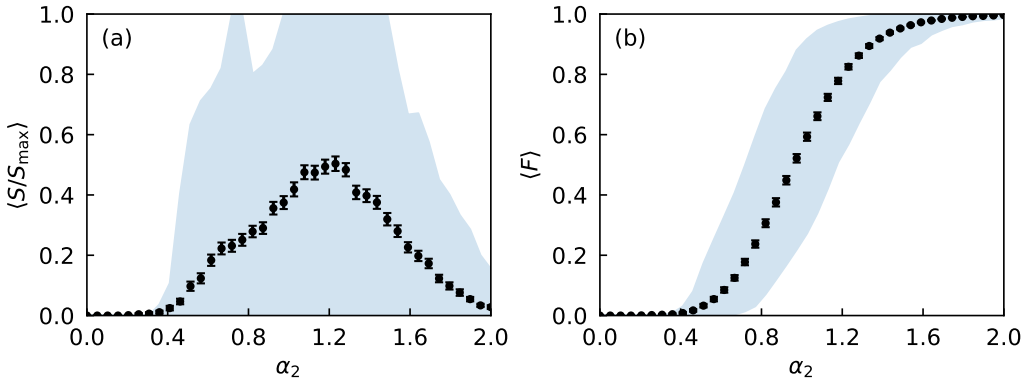


Figure 5.4: (a) Fraction S of sampled solutions that are optimal, normalised by the maximum measured fraction $S_{\max}$, plotted against the penalty strength α_2 for a simulated annealing algorithm solving the 1Q-PCP. Each point is averaged over 200 instances of the problem, which we denote with $\langle \cdot \rangle$. Error bars represent the standard error in the mean. The blue shaded area contains the 5th to 95th percentile values of $S/S_{\max}$. Note that the value of $S_{\max}$ is calculated for each instance separately and is a maximum over the plotted values of α_2. (b) Same as (a), with the y-axis showing the fraction F of samples that are feasible instead. Figure adapted from [2].

5.4.1 Single-quarter problem penalty strength tuning

For the experiments on the 1Q-PCP, we set the quadratic penalty strength to $\alpha_2 = 1.2$ for all problem instances. We chose this value based on the fraction S of optimal solutions and the fraction F of feasible solutions sampled by a SA algorithm as a function of α_2, which are plotted in Fig. 5.4 for 200 randomly selected problem instances. In this figure, S is normalised by its maximum value $S_{\max}$ for each instance to prevent the instances with large values of $S_{\max}$ from dominating the average. The figure plots values averaged over the problem instances, which we denote with $\langle \cdot \rangle$. Fig. 5.4(a) shows that $\langle \frac{S}{S_{\max}} \rangle$ peaks near $\alpha_2 = 1.2$, which is our choice of penalty strength. While the large confidence interval indicates that there are some instances for which $\alpha_2 = 1.2$ is not a near-optimal choice, the fact that $\langle \frac{S}{S_{\max}} \rangle$ reaches ≈ 0.5 implies that S is not far from its maximum value for most problem instances when $\alpha_2 = 1.2$. In Fig. 5.4(b), we can see that for the majority of problem instances, $\alpha_2 = 1.2$ produces a large fraction F of solutions that are feasible.

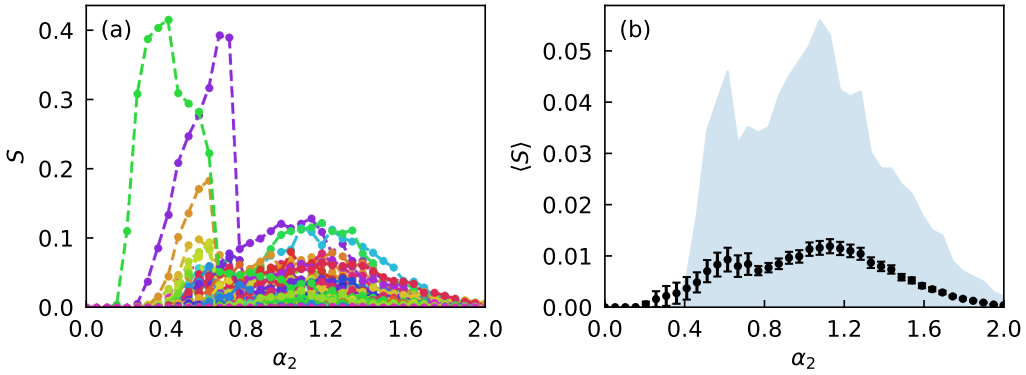


Figure 5.5: (a) Fraction S of simulated annealing samples that are optimal plotted against the quadratic penalty strength α_2 for 200 different instances of the single-quarter promotion cannibalization problem, shown in different colours. Dashed lines connecting the points for each instance are shown to guide the eye. (b) Plot showing the average value of S with error bars representing the standard error in the mean. The blue shaded area contains the 5th to 95th percentile values of S. Figure adapted from [2].

For the sake of transparency, we plot S for each instance individually and the average value $\langle S \rangle$ without normalisation in Fig. 5.5. This is for the same selection of problem instances as in Fig. 5.4. Comparing Fig. 5.4(a) and Fig. 5.5(b), we see that both $\langle S \rangle$ and $\langle \frac{S}{S_{\max}} \rangle$ are maximised at approximately the same value of α_2.

5.4.2 Four-quarter problem penalty strength tuning

The 4Q-PCP with 10 products has 44 constraints in total, which are grouped into the sets of constraints C1, C2, and C3. In our experiments, we used a different value of the quadratic penalty strength for each set of constraints. These penalty strengths are denoted $\alpha_2^{(\mathrm{C1})}$, $\alpha_2^{(\mathrm{C2})}$, and $\alpha_2^{(\mathrm{C3})}$. In principle, each of the 44 constraints could take a unique penalty strength, but tuning 44 parameters would be much more computationally challenging than tuning 3 parameters. It is unclear whether having these extra parameters would be of much benefit given the similarities in the subproblems that each constraint in a particular set acts on.

In line with the previous subsection, the values of $\alpha_2^{(\mathrm{C1})}$, $\alpha_2^{(\mathrm{C2})}$, and $\alpha_2^{(\mathrm{C3})}$ were determined based on an analysis of SA on the four-quarter problem. We chose the

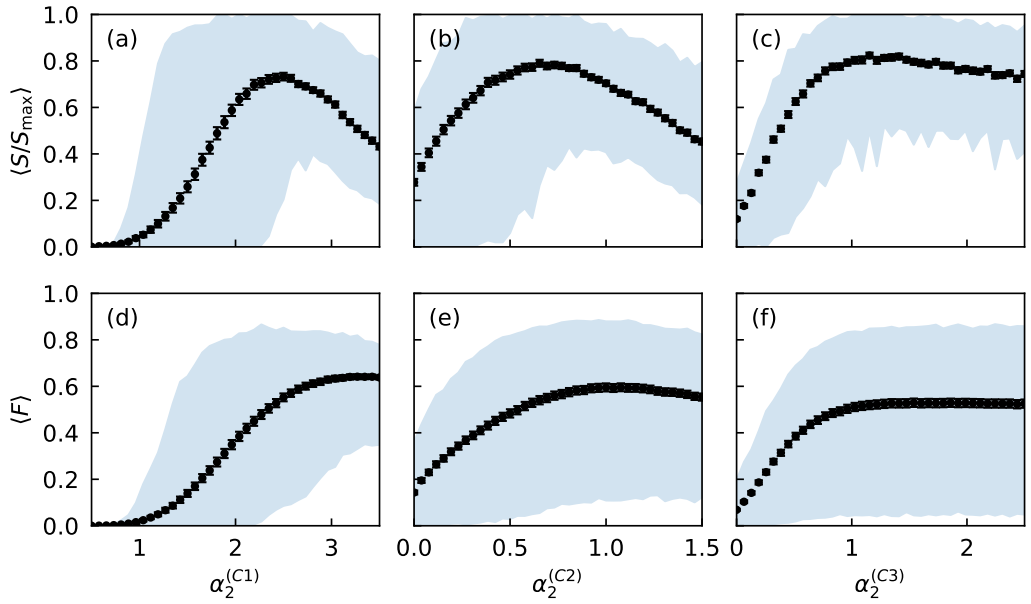


Figure 5.6: Normalised fraction $S/S_{\max}$ of simulated annealing samples that are optimal when solving the 4Q-PCP plotted against the quadratic penalty strength for the constraints (a) C1, (b) C2, and (c) C3. Only one quadratic penalty strength is varied in each plot while the others are kept at the constant values $\alpha_2^{(\mathrm{C1})} = 2.4$, $\alpha_2^{(\mathrm{C2})} = 0.6$, and $\alpha_2^{(\mathrm{C3})} = 1.2$. Each point is an average over 200 random problem instances. Error bars show the standard error in the mean, and the blue shaded areas contain the 5th to 95th percentile values. $S_{\max}$ is calculated for each instance separately and is a maximum over the plotted values of $\alpha_2^{(\mathrm{C}i)}$. In (d), (e), and (f), we show similar plots for the fraction F of sampled solutions that are feasible against the quadratic penalty strengths for the constraints C1, C2, and C3 respectively. Figure adapted from [2].

penalty strengths $\alpha_2^{(\mathrm{C1})} = 2.4$, $\alpha_2^{(\mathrm{C2})} = 0.6$, and $\alpha_2^{(\mathrm{C3})} = 1.2$ because they led to large fractions of optimal and feasible solutions being sampled on average, which is shown in Fig. 5.6.

While we cannot expect these SA results to exactly reflect the behaviour of a D-Wave annealer, the broadness of the peaks in Fig. 5.4(a) and Fig. 5.6(a-c) reassure us that the results from a quantum annealer would not be substantially impacted by small changes in the choices of quadratic penalty strengths.

5.5 Single-quarter problem on a D-Wave annealer

We have run the 1Q-PCP on a D-Wave annealer using the quadratic and linear penalty methods. Recall that the problem we consider involves 100 products and the constraint is to select 50 products to promote. Out of 10,000 problem instances that we randomly generated, there are 1,406 instances for which a value of α_1 that produces the correct ground state could not be found. The D-Wave runs in this subsection were performed on 1,000 instances that were randomly selected from the remaining 8,594 instances for which the linear penalty method is able to implement the desired constraint.

We compare the performance of the two penalty methods in experiments on the D-Wave annealer in Fig. 5.7. In Fig. 5.7(a), we see that the fraction F of sampled solutions that are feasible is typically larger when using the linear penalty method. For 117 out of the $1,000$ problem instances, the quantum annealer did not sample any feasible solutions using the quadratic penalty method. For the linear penalty method, this occurred for only 4 instances. This result shows that even though the linear penalty method uses fewer resources to implement constraints by not using any quadratic terms, it can still lead to more feasible solutions being sampled because of the benefits that result from the more efficient encoding.

Fig. 5.7(b) shows histograms of the approximation ratios R of the best feasible solutions sampled with each penalty method. With the quadratic penalty method, the value of R for the best sample is typically around 0.5 to 0.7, indicating that the quantum annealer is only able to sample low-quality feasible solutions. In comparison, the annealer consistently samples solutions with approximation ratios close to 1 with the linear penalty method. In fact, the optimal solution ($R = 1$) was found for 541 instances with the linear penalty method. These results show an example where there is a clear advantage of using the linear penalty method over the quadratic penalty method. This performance enhancement more than compensates for the extra optimiser calls required to tune the linear penalty

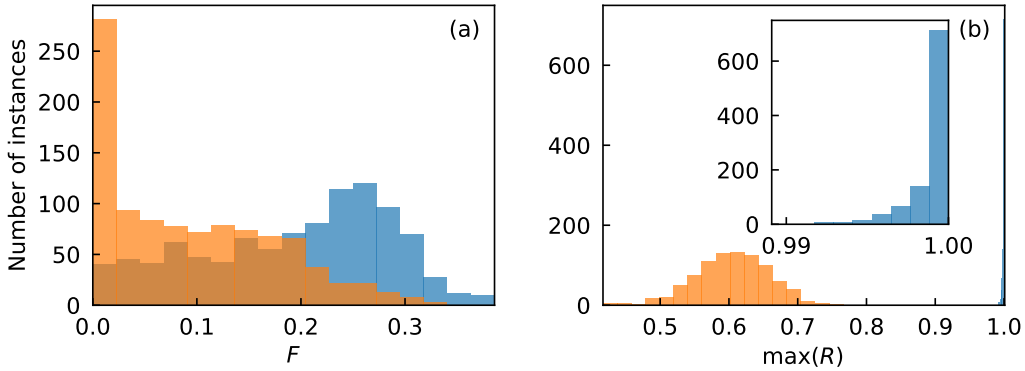


Figure 5.7: (a) Histograms of the fraction F of 1,000 D-Wave annealer samples that are feasible solutions for 1,000 1Q-PCP instances. We compare the use of the linear penalty method (blue) against the quadratic penalty method (orange) for encoding the problem's constraint. (b) Histograms of the maximum approximation ratio R for the same D-Wave sample sets. To make the shapes of the distributions clear, we have used different bin widths for the two histograms in (b). The inset zooms onto the histogram for the linear penalty method so that it is more visible. Figure adapted from [2].

strength. We suspect that in this example, the primary factor responsible for the increase in performance with the linear penalty method is the improvement in minor embedding efficiency.

5.6 Simulated annealing for the single-quarter problem

We have run the simulated annealing solver in the D-Wave Ocean software library on the same 1,000 single-quarter problem instances that were used in Sec. 5.5 for the QA experiments. These runs use the same values of α_1 and α_2, which were tuned as described in Sec. 5.4. In Fig. 5.8, we compare SA performance using the linear and quadratic penalty methods in the same way as Fig. 5.7. Since minor embedding is not necessary for classical methods, we have run the problem instances without minor embedding, and we present these results in Fig. 5.8(a-b). We have also solved the minor embedded problem instances with the same minor embeddings that were used for the QA experiments, which correspond to Fig. 5.8(c-d). This allows us

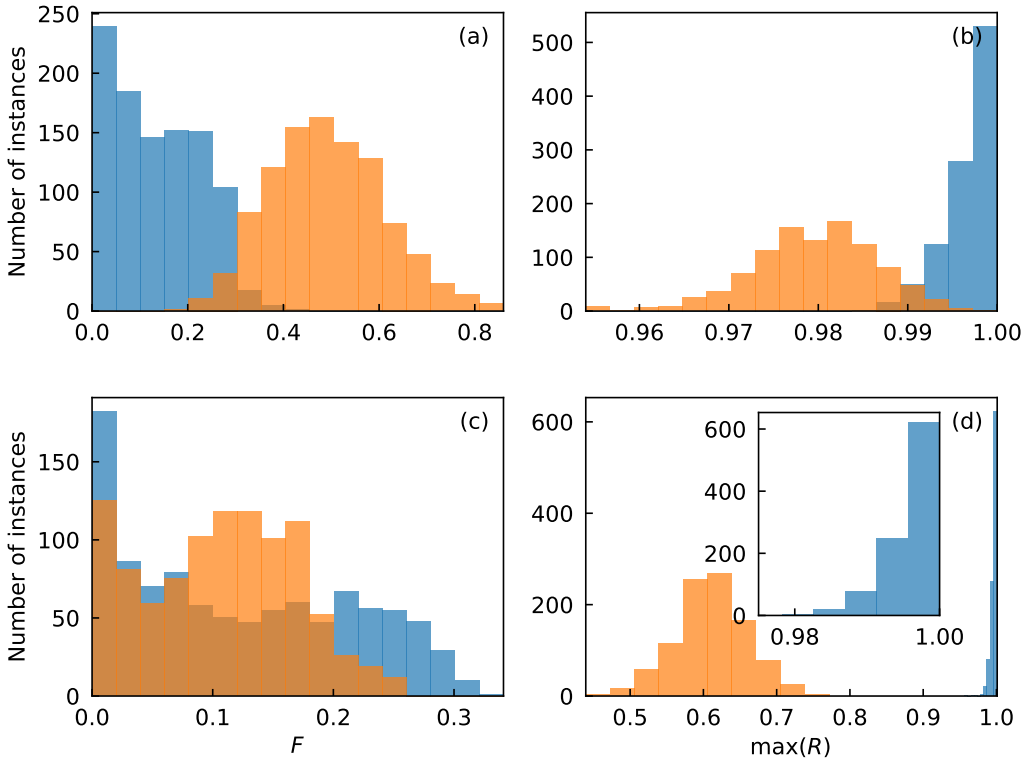


Figure 5.8: (a) Histograms of the fraction F of 1,000 SA samples that are feasible solutions for 1,000 1Q-PCP instances. We compare the use of the linear penalty method (blue) against the quadratic penalty method (orange) for encoding the problem's constraint. Minor embedding was not performed, and an anneal time of $t_f = 20$ was used. (b) Histograms of the maximum approximation ratio R for the same SA sample sets. (c) and (d) Similar histograms for problem instances that were minor embedded onto the D-Wave advantage hardware graph before being given to the SA solver. These samples were taken with an anneal time of $t_f = 1{,}000$. To make the shapes of the distributions clear, we have used different bin widths for the two histograms in (d). The inset zooms onto the histogram for the linear penalty method so that it is more visible.

to empirically study the impact of minor embedding on the performance of the two penalty methods, which is not possible to do on a D-Wave quantum annealer. Note that an anneal time of $t_f = 20$ was used for the results in Fig. 5.8(a-b) and $t_f = 1{,}000$ was used for Fig. 5.8(c-d). We chose these values so that the problem was not always easy to solve with both penalty methods.

In Fig. 5.8(a), we see that when minor embedding isn't performed, the linear penalty method leads to smaller fractions F of feasible solutions being sampled than the quadratic method. There are 3 problem instances for which a feasible solution

was not sampled with the linear penalty method, whereas the quadratic penalty method led to feasible solutions being sampled for every instance. However, there is a performance enhancement with the linear penalty method when considering the maximum approximation ratios $\max(R)$ in Fig. 5.8(b). The optimal solution was sampled for 234 instances with the linear penalty method and 0 instances with the quadratic penalty method. These results contrast with the QA results in Fig. 5.7, where the linear penalty method enhanced F and produced a more significant improvement in $\max(R)$

The results in Fig. 5.8(c-d) for SA with minor embedding are much more similar to those in Fig. 5.7 for D-Wave QA. In Fig. 5.8(c), there is a slight increase in F on average when using the linear penalty method, although there are 21 problem instances for which the linear method did not lead to any feasible samples as opposed to 10 instances with the quadratic method. Fig. 5.8(d) shows that with the linear penalty method, optimal and near-optimal solutions are sampled, whereas with the quadratic penalty method, the best sampled solutions for each instance have approximation ratios of ≈ 0.6. These results provide evidence that minor embedding efficiency improvements are a key factor responsible for the outperformance of the linear penalty method in the single-quarter problem experiments. However, even without minor embedding, the linear penalty method does produce a modest improvement in the quality of feasible solutions. This indicates that there are other benefits of using the linear penalty method that can also apply to classical methods.

5.7 Four-quarter problem on a D-Wave annealer

In this section, we analyse the performance of QA in solving the 4Q-PCP with $n_p = 10$ products using a variety of penalty schemes. Recall that we set the seasonal scale factors in Eq. (2.33) to $\boldsymbol{\lambda} = (1.5, 1.0, 1.0, 1.5)^\mathsf{T}$ and the constraints C1 require four out of the ten products to be promoted in each quarter. The constraints C2 require

each product to be promoted once or twice, while the constraints C3 do not allow consecutive promotions of the same product. For the four constraints C1, we either use the quadratic penalty method, the linear penalty method, or a combination of both. In all the penalty schemes we consider, the constraints C2 and C3 are implemented with quadratic penalties. Note that since the constraints C2 are inequalities, ten extra variables are introduced with the slack variable encoding used in Eq. (2.49), bringing the total variable count to 50 for this problem.

The C matrices used in this section for the 4Q-PCP have an average connectivity of ≈ 5.1, making them more dense than those used in the 1Q-PCP. This is due to the smaller number of promotions per quarter in this problem, which results in some optimal solutions having zero cannibalization if the C matrices are made more sparse. Our choice of problem size is limited by the maximum problem size for which the D-Wave annealer can produce good solutions. In real-world problems, the number of products and promotions would be much larger, so sparser C matrices would still result in optimal solutions with nonzero cannibalization.

5.7.1 Comparing the linear and quadratic penalty schemes

We first consider the penalty schemes where the four constraints C1 are all implemented with quadratic penalties or are all implemented with linear penalties. For 6,066 of the 10,000 problem instances we generated, we were able to find values of α_1 that produced a feasible ground state in $\mathcal{H}_I$ using the penalty scheme with linear penalties. The search strategy outlined in Sec. 3.6 was used to find the α_1 values. For 3,082 of these instances, it was possible to produce a feasible ground state using the same value of α_1 for each of the four linear penalties. In this subsection, we consider 1,500 instances that were randomly selected from the 3,082 instances that can be successfully constrained with a single α_1 parameter, and we use a single penalty strength for each of the four constraints C1. As before, the value of α_1 we used for each instance was chosen uniformly at random from the interval of values that produce a feasible ground state.

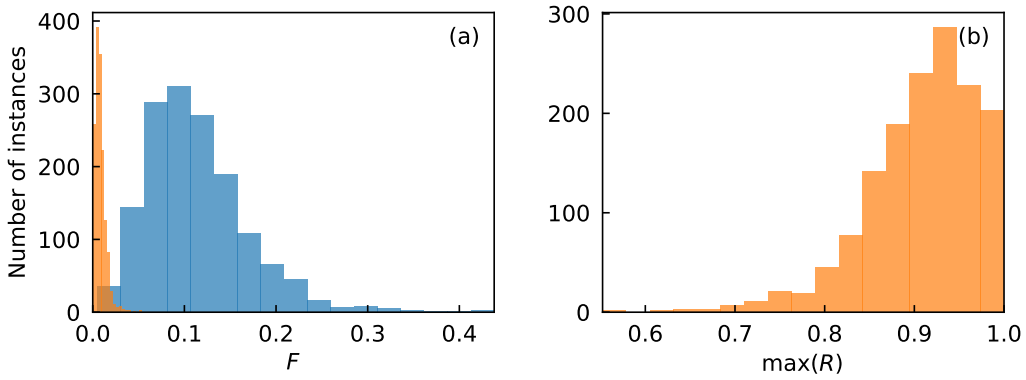


Figure 5.9: (a) Histograms of the fraction F of 1,000 D-Wave annealer samples that are feasible solutions for 1,500 4Q-PCP instances. We compare the use of the linear penalty method (blue) against the quadratic penalty method (orange) for encoding the constraints C1. To make the shapes of the distributions clear, we have used different bin widths for the two histograms. (b) Histogram of the maximum approximation ratios R for the D-Wave sample sets using quadratic penalties. A histogram for the sample sets using linear penalties is not shown because $\max(R)$ was equal to one for all but nine of the instances. Figure adapted from [2].

Fig. 5.9 shows histograms of the fraction F of samples that are feasible and the best approximation ratios $\max(R)$ for our QA sample sets. We can see that there are significant improvements in both performance metrics when switching the four penalties on each fiscal quarter from quadratic to linear. This is in alignment with our findings for the single-quarter problem. With the penalty scheme involving four linear penalties, feasible solutions were sampled for all 1,500 problem instances and optimal solutions were sampled for 1,491 of them. With the penalty scheme involving only quadratic penalties, no feasible solutions were found for 22 instances and optimal solutions were found for only 55 instances.

5.7.2 Using a combination of linear and quadratic penalties

In the previous subsection, we mentioned that there were 3,934 instances for which we could not find values of α_1 for each quarter that led to a feasible ground state of $\mathcal{H}_I$. This is either because no good combination of values of α_1 exist or because our search strategy was not able to find them in the specified number of iterations. This may occur more often at larger problem sizes or for problems with different

formulations. We propose that when this does happen, linear penalties that are not correctly implementing the intended constraint can be switched to quadratic penalties one by one until feasible solutions are sampled. The goal is to maintain some of the benefits of using the linear penalty method by keeping penalties linear where possible. In this subsection, we investigate whether this method would work well in practice.

Due to our choice of seasonal scale factors $\boldsymbol{\lambda} = (1.5, 1.0, 1.0, 1.5)^{\mathsf{T}}$, the objective function of the four-quarter problem is exactly the same for the first & fourth quarters' subproblems and the second & third quarters' subproblems. Because of this, the objective value of a solution is unchanged if the solution is "flipped" by swapping the promotion plans for the first & fourth quarters and for the second & third quarters. It can be easily shown that for the constraints C1, C2, and C3, a flipped solution is feasible if and only if the original solution is feasible. Therefore, for this example problem, the issue of a misbehaving linear penalty cannot be resolved by only switching that particular quarter's penalty to be quadratic because the flipped solution, which is infeasible, will continue to have a lower energy in $\mathcal{H}_I$ than the feasible solutions. Instead, penalties must be switched from linear to quadratic in pairs. Specifically, the only two combinations of linear and quadratic penalties for the constraints C1 that can possibly produce a feasible ground state for instances that cannot be constrained with four linear penalties are those where the first & fourth quarters or the second & third quarters have quadratic penalties applied. We refer to these as the QLLQ and LQQL penalty schemes respectively.

Out of the 3,934 problem instances for which we could not find good α_1 values for the scheme with four linear penalties, we found that 2,298 of the instances are able to be successfully constrained with the LQQL or QLLQ penalty scheme. Following a similar argument as above where we consider the objective values of flipped solutions, the two α_1 values can always be chosen to be the same in cases where the LQQL or QLLQ penalty scheme is successful in implementing the constraints. Therefore, we have used the same value of α_1 for each pair of linear penalties in

Penalty scheme	Fraction of problem instances for which a feasible solution was sampled
All-quadratic	≈ 0.96
LQQL	≈ 0.75
QLLQ	0.60

Table 5.1: Fraction of problem instances for which a feasible solution was found in at least one of the 1,000 D-Wave samples for the four-quarter promotion cannibalization problem. We compare the penalty scheme of applying quadratic penalties to all four quarters against the schemes of using linear penalties for the first and last quarters (LQQL) or the second and third quarters (QLLQ).

our analysis of the LQQL and QLLQ schemes. In the experiments on a D-Wave annealer, we chose the α_1 value for each instance randomly from the interval of values that produce feasible ground states, as before.

Table 5.1 compares the fraction of problem instances for which a feasible solution was sampled in our runs on the D-Wave annealer for the LQQL, QLLQ, and all-quadratic penalty schemes. We find that using quadratic penalties for all four constraints C1 results in feasible solutions being sampled more often than when using the LQQL or QLLQ schemes. The LQQL and QLLQ schemes were only applied to the subset of problem instances for which the penalty scheme was able to produce a feasible ground state, and the all-quadratic scheme was applied to all instances in either subset. Since the instances in each set are not the same, the results for the LQQL and QLLQ schemes are not directly comparable. With this caveat in mind, we observe that using the LQQL scheme resulted in feasible solutions being sampled for a larger fraction of instances than with the QLLQ scheme.

In Fig. 5.10, we compare the best approximation ratios achieved with the LQQL and QLLQ penalty schemes against those with the all-quadratic scheme. We find that the approximation ratios are very similar between the different penalty schemes. This differs from the previous QA experiments, where introducing linear penalties produced significantly higher quality solutions. The mean value of $\max(R)$ is slightly smaller for the LQQL scheme than for the all-quadratic scheme. For the

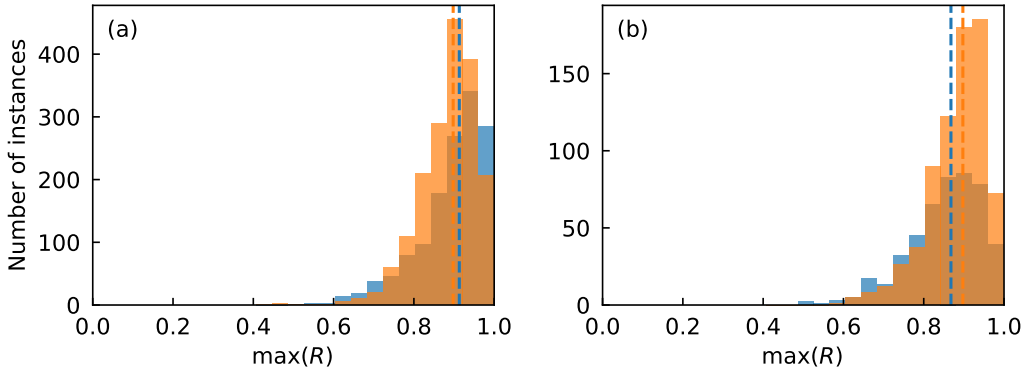


Figure 5.10: (a) Histograms of the maximum approximation ratios R of 1,000 D-Wave annealer samples for 1,815 4Q-PCP instances that can be successfully constrained with the LQQL penalty scheme. We compare the use of the LQQL penalty scheme (blue) against the all-quadratic penalty scheme (orange). Cases in which no feasible solution was found are not shown in the histograms. (b) Similar histograms comparing the QLLQ (blue) and all-quadratic (orange) penalty schemes for 780 instances that can be successfully constrained with the QLLQ penalty scheme. Note that in both (a) and (b), the area under the blue histogram is smaller than for the orange histogram because the LQQL and QLLQ schemes produced feasible samples for fewer instances than the all-quadratic scheme.

QLLQ scheme, the mean value of $\max(R)$ is slightly larger than for the all-quadratic scheme. However, we must question whether these differences are statistically significant. Moreover, the different penalty schemes found feasible solutions for different subsets of instances, so the instances included in the histograms in Fig. 5.10 are different for each penalty scheme. Therefore, a more rigorous analysis is required to compare the subtle differences in performance between the penalty schemes in a way that is more accurate than what can be inferred from Fig. 5.10.

To perform a more rigorous analysis, we compare the performance on a per-instance basis in order to avoid comparing results for instances that are more difficult to solve with results for easier instances. We leave out instances for which one of the penalty methods being compared resulted in no feasible solutions being sampled. Fig. 5.11 plots the difference in objective values $f(\mathbf{x})$ between the best feasible solutions sampled using the all-quadratic penalty scheme and the LQQL or QLLQ scheme. A negative difference in $f(\mathbf{x})$ means that a higher quality solution was sampled with the introduction of the linear penalties, whereas a positive difference indicates that

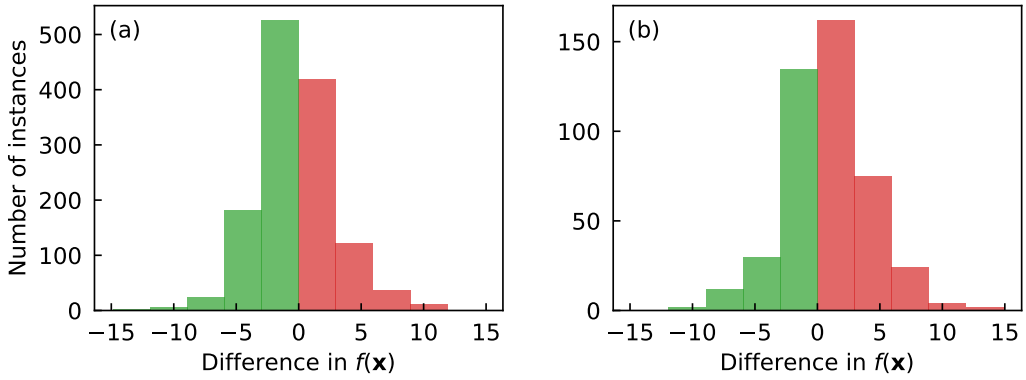


Figure 5.11: Histogram of the difference between the objective value $f(\mathbf{x})$ of the best feasible sample using the all-quadratic penalty scheme and (a) the LQQL penalty scheme or (b) the QLLQ penalty scheme. A negative difference (green) means that the LQQL or QLLQ scheme found a higher quality solution, and a positive difference (red) indicates that the all-quadratic scheme found a higher quality solution. There is one instance that is left out of (a) because the difference in $f(\mathbf{x})$ was zero. Figure adapted from [2].

the all-quadratic scheme produced the best solution. There are 741 instances for which the LQQL scheme improved the quality of the best solution compared to the all-quadratic scheme and 592 instances for which the LQQL scheme worsened the quality of the best solution. For the QLLQ scheme, there are 179 instances for which the scheme improved the quality of the best solution and 265 instances for which the scheme worsened the quality of the best solution.

It appears that LQQL scheme had a slight positive impact on the solution quality whereas QLLQ scheme had a slight negative impact, but it is unclear from this analysis alone whether these results are statistically significant. To determine this, we have performed hypothesis tests on the number of times n_b that the LQQL or QLLQ scheme found a lower energy feasible solution, i.e. performed better, than the all-quadratic penalty scheme and the number of times n_w it could only find higher energy feasible solutions, i.e. performed worse. We ignore one case where the best feasible solution had the same energy for both penalty schemes. Our null hypothesis is that the probability of the LQQL or QLLQ scheme producing better performance than the all-quadratic scheme is the same as the probability of the schemes producing worse performance. The statistical significance for this null

hypothesis is given by

$$p = \frac{1}{2^{n_b+n_w}} \sum_{k=n_b}^{n_b+n_w} \begin{pmatrix} n_b + n_w \\ k \end{pmatrix}, \tag{5.1}$$

where $\begin{pmatrix} n \\ k \end{pmatrix}$ is the binomial coefficient. This is effectively the probability that the penalty scheme being considered has happened to perform better than the all-quadratic scheme at least as many times if there was a 50% chance of it performing better for any given instance. Conventionally, $p < 0.05$ would indicate a statistically significant result that rejects the null hypothesis [148] and confirms that the penalty scheme under consideration performs better than the all-quadratic scheme. We can also use this as a test for whether the penalty scheme performs statistically significantly worse than the all-quadratic scheme. For this, we define $\widetilde{p} = 1 - p$. A value of $\widetilde{p} < 0.05$ would be a statistically significant result that confirms worse performance than the all-quadratic scheme.

Substituting the values $n_b = 741$ and $n_w = 592$ into Eq. (5.1) gives $p = 2.5 \times 10^{-5}$, which confirms the statistical significance of our result that the LQQL scheme finds higher quality solutions for more instances than the all-quadratic scheme. The values $n_b = 179$ and $n_w = 267$ give $\widetilde{p} = 1.2 \times 10^{-5}$, which shows that the result indicating that the QLLQ scheme finds higher quality solutions for fewer instances than the all-quadratic scheme is also statistically significant. All of these experiments were performed at similar times and on the same quantum hardware. Unless there were significant enough changes in the behaviour of the quantum hardware between the time that the LQQL or QLLQ runs were performed and the time that the all-quadratic penalty scheme runs were performed, the differences in performance are due to the effect that the choice of penalty scheme has.

One may wonder why switching half of the constraints from quadratic to linear penalties has such a small effect on performance compared to switching all penalties to linear. We suspect that the reason for this is that the dynamic range improvements from using linear penalties are a significant factor in improving per-

formance in this example problem, and having a single quadratic penalty on one of the quarters is enough to eliminate most of these dynamic range improvements, as detailed in Sec. 5.3.2. For this example problem, the LQQL and QLLQ schemes are instead more successful in improving the minor embedding efficiency, which we analysed in Sec. 5.2.3. We expect that at larger problem sizes, both the minor embedding and dynamic range improvements would be more significant than they are here and this would lead to a clear performance advantage in using the LQQL or QLLQ schemes. The expected improvements in dynamic range usage and minor embedding efficiency also depend on the formulation and sparsity of the problem. More significant improvements might be observed for other problems with multiple constraints.

The difference in dynamic range improvement between the LQQL and QLLQ penalty schemes could explain why the LQQL scheme performed slightly better than the all-quadratic scheme in finding low-energy solutions, whereas the QLLQ scheme performed slightly worse (Fig. 5.11). It could also be why we found that the LQQL scheme produces feasible solutions more often than the QLLQ scheme (Table 5.1). This underscores the importance of taking dynamic range considerations into account when selecting which penalties to turn linear. While the results in Sec. 5.5 clearly demonstrated the benefits to performance from having more efficient minor embeddings with the linear penalty method, the results in this section demonstrate that the dynamic range usage improvements can result in better performance too.

5.8 Chapter conclusions

In this Chapter, we have used a D-Wave quantum annealer to experimentally test the linear Ising penalty method for encoding constraints in combinatorial optimisation problems. We have looked at an example promotion cannibalisation problem involving a single constraint and another example involving multiple constraints. For both problems, we found that the majority of the instances that we generated

could be exactly constrained with linear penalties encoding the constraints C1. For the problem with multiple constraints, some of the instances that could not be exactly constrained with four linear penalties could be exactly constrained by switching two of the linear penalties to quadratic penalties.

For both the problem with a single constraint and the problem with multiple constraints, we observed significant improvements in the performance of the quantum annealer when using linear penalties instead of quadratic penalties. These improvements were demonstrated by increases in the fraction of sampled solutions that satisfy the constraints and the approximation ratios of the sampled solutions. The most dramatic improvement was observed in the quality of sampled solutions for the problem with a single constraint. Encoding this problem's constraint with a quadratic penalty resulted in the quantum annealer finding no high-quality solutions for any of the problem instances. Switching to a linear penalty function led to the quantum annealer finding the optimal solution for more than half of the problem instances and finding near-optimal solutions for most of the other instances. In these experiments, we found signs that improvements in minor embedding and effective dynamic range both contributed to better performance. We note that there are other methods for encoding constraints that we have not considered here, and our observation of a performance improvement with the linear penalty method is specifically a comparison against the quadratic penalty method.

For instances of the problem with multiple constraints that could not be successfully constrained with four linear penalties, switching two of the linear penalties to quadratic penalties resulted in a much smaller difference in performance when compared to the all-quadratic penalty scheme. We found that there was a small but statistically significant improvement in the quality of sampled solutions when the two linear penalties were applied to the constraints that have a larger impact on the effective dynamic range, despite having a negative impact on the feasibility of solutions. This suggests that a strategy of focusing on removing the quadratic penalties that are most detrimental to the dynamic range is worthwhile, which

is something that can be determined without performing any anneals. At larger problem sizes, where the minor embedding and dynamic range benefits of linear penalties are greater, we predict that there would be a more substantial performance advantage of linear over quadratic penalties.

Chapter 6

Variable penalty strength quantum approximate optimisation algorithm

6.1 Introduction

In this chapter, we propose a new variant of the QAOA for constrained optimisation problems in which the strengths of penalty functions are varied in each layer of the quantum circuit. This introduces additional variational parameters and enhances the algorithm's ability to find good solutions. We compare the performance of this variant against other variants of the QAOA when solving the 2Q-PCP in numerical simulations. We consider the use of quadratic and linear penalty functions in our simulations. The simulations are of error-free closed-system dynamics and assume all-to-all connectivity of the physical qubits.

6.2 Variable penalty strength QAOA

An Ising Hamiltonian that encodes an optimisation problem with a set of constraints $\mathcal{C}$ using the penalty method can be written as

$$\mathcal{H}_I = \mathcal{H}_{\text{objective}} + \sum_{c \in \mathcal{C}} \alpha_c \mathcal{H}_c, \tag{6.1}$$

where $\mathcal{H}_{\text{objective}}$ is the Ising Hamiltonian encoding the unconstrained objective function and $\alpha_c \mathcal{H}_c$ is a penalty for the constraint $c \in \mathcal{C}$ with penalty strength $\alpha_c \in \mathbb{R}$. $\mathcal{H}_c$ may take a variety of different forms, such as the quadratic and linear penalties that were defined in Sec. 2.8 and Sec. 3.2.

We propose a variant of the QAOA that includes additional variational parameters to control the strength of the penalty terms for each layer independently, and we refer to it as the variable penalty strength quantum approximate optimisation algorithm (VPS-QAOA). This takes inspiration from the MA-QAOA [84; 85], which we described in Sec. 2.5.2, in the sense that it aims to improve performance by including more variational parameters in the ansatz. The variable penalty strength ansatz places more focus on the problem's constraints by associating the extra variational parameters with the penalty functions. Therefore, we expect it to be most useful in scenarios where the standard QAOA struggles with satisfying the constraints.

The VPS-QAOA modifies the QAOA ansatz by replacing the phase separator unitary in Eq. (2.23) with

$$U_I(\gamma_l, \delta_l) = \mathrm{e}^{-\mathrm{i}\left(\gamma_l \mathcal{H}_{\text{objective}} + \delta_l \sum_{c \in \mathcal{C}} \alpha_c \mathcal{H}_c\right)}, \tag{6.2}$$

where $\boldsymbol{\delta}$ is a vector of p additional variational parameters that rescale the penalties and effectively allow the penalty strengths to be varied between layers. The standard QAOA is a special case of the VPS-QAOA where $\delta_l = \gamma_l \ \forall l$. This means that with optimal values of the variational parameters, the expected value of $\mathcal{H}_I$, as defined in Eq. (6.1) with constant penalty strengths, cannot be higher for the

VPS-QAOA ansatz than for the QAOA ansatz with the same number of layers, as has been shown for the MA-QAOA [85]. Hence, the guarantee from the adiabatic theorem of producing an arbitrarily large probability of measuring a ground state of $\mathcal{H}_I$ with enough layers of the QAOA, which we discussed in Sec. 2.5.1, also applies to the VPS-QAOA. It is also guaranteed that the expectation value of $\mathcal{H}_I$ with optimal parameter choices cannot increase after adding a layer to the VPS-QAOA ansatz for the same reason that was discussed in Sec. 2.5.1 for the QAOA.

The number of additional variational parameters introduced in the VPS-QAOA is typically much smaller than in the MA-QAOA. The total number of parameters in the VPS-QAOA ansatz is $3p$, whereas the MA-QAOA uses $\frac{pn(n+3)}{2}$ parameters when the Ising Hamiltonian has nonzero values for all possible couplings and local fields. In comparison, the standard QAOA uses $2p$ variational parameters. As detailed in Sec. 2.5.2, the large number of variational parameters introduced in the MA-QAOA makes the task of classically optimising the parameters more challenging. Therefore, it may be the case that an ansatz with more parameters than the standard QAOA but with fewer parameters than the MA-QAOA is preferable in some situations. This motivates the study of variants like the VPS-QAOA that introduce new parameters but not as many as the MA-QAOA.

For problems with multiple constraints, it may be beneficial to define the phase separator unitary as

$$U_I(\gamma_l, \delta_l) = \mathrm{e}^{-\mathrm{i}\left(\gamma_l \mathcal{H}_{\text{objective}} + \sum_{c\in\mathcal{C}} \delta_{l,c}\alpha_c \mathcal{H}_c\right)}, \tag{6.3}$$

thereby assigning a different variational parameter $\delta_{l,c}$ to each penalty. This gives the classical optimiser more fine-grained control over the penalty strengths at the expense of increasing the total number of parameters to $2p + p|\mathcal{C}|$, where $|\mathcal{C}|$ is the number of constraints. It is also possible to use an approach in which similar types of constraints share the same variational parameter. For example, when solving the 4Q-PCP, one could define three variational parameters $\delta_{l,\mathrm{C1}}$, $\delta_{l,\mathrm{C2}}$, and $\delta_{l,\mathrm{C3}}$ associated with the penalties for the sets of constraints C1, C2, and C3 respect-

ively. We do not consider approaches involving multiple variable penalty strength parameters per layer. Therefore, all of our results concerning the VPS-QAOA use the phase separator defined in Eq. (6.2).

6.3 Ansatzes considered

In this chapter, we compare the performance of a variety of different QAOA variants that each have different ansatzes. The ansatzes we consider are those of the standard QAOA, the VPS-QAOA, the MA-QAOA, and another variant that we refer to as the two-angle quantum approximate optimisation algorithm (2A-QAOA). We use circuits with $p = 6$ layers for all of these ansatzes and also consider a $p = 1$ version of the MA-QAOA ansatz.

The 2A-QAOA is a variant of the QAOA that we have defined for the purpose of having a performance comparison that involves a similar number of variational parameters as the VPS-QAOA. Like the VPS-QAOA, the phase separator unitary for each layer of the 2A-QAOA ansatz includes two variational parameters, or angles, γ_l and δ_l. The difference is that instead of associating δ_l with the penalties, we randomly select half of the couplings and half of the local fields in $\mathcal{H}_I$ and associate δ_l with those. The phase separator unitary for the 2A-QAOA can be written as

$$U_I(\gamma_l) = \prod_{i,j\in E_1} \mathrm{e}^{-\mathrm{i}\gamma_l J_{i,j}\sigma_i^z\sigma_j^z} \prod_{i,j\in E_2} \mathrm{e}^{-\mathrm{i}\delta_l J_{i,j}\sigma_i^z\sigma_j^z} \prod_{i\in V_1} \mathrm{e}^{-\mathrm{i}\gamma_l h_i\sigma_i^z} \prod_{i\in V_2} \mathrm{e}^{-\mathrm{i}\delta_l h_i\sigma_i^z}, \qquad (6.4)$$

where E_1 (E_2) is the set of qubit pairs with labels (i, j) that have nonzero couplings $J_{i,j}$ associated with the variational parameters $\boldsymbol{\gamma}$ ($\boldsymbol{\delta}$) and V_1 (V_2) is the set of qubits with labels i that have nonzero local fields h_i associated with the variational parameters $\boldsymbol{\gamma}$ ($\boldsymbol{\delta}$).

In our simulations of the QAOA and the 2A-QAOA in this chapter, we treat the penalty strength as an additional variational parameter to be optimised alongside the other variational parameters. This allows the penalty strengths to be tuned

Variant	Number of layers p	Number of variational parameters d
QAOA	6	13
VPS-QAOA	6	18
2A-QAOA	6	19
MA-QAOA	1	96
MA-QAOA	6	576

Table 6.1: Number of variational parameters in different variants of the QAOA when solving the 2Q-PCP with $n_p = 8$ products and $n = 16$ qubits using quadratic penalty functions to encode all constraints. Note that our implementations of the QAOA and the 2A-QAOA involve a parameter that controls the strength of the penalty functions.

individually for each instance instead of tuning across all instances, as we did in Chapter 4. We use the same penalty strength α_2 for all the constraints in the 2Q-PCP when using the quadratic penalty method. In cases where linear penalties are used, a penalty strength $\alpha_1 = -\alpha_2$ with the same magnitude but opposite sign is used. This means that in all cases, a single variational parameter is used to control all penalty strengths. Hence, the number of variational parameters becomes $2p + 1$ for the QAOA and $3p + 1$ for the 2A-QAOA. Better performance may be achieved by using different parameters for each penalty or each set of constraints C1 and C3, but we have not attempted this. For the VPS-QAOA and MA-QAOA, the penalty strengths can already be effectively controlled by the existing variational parameters in the ansatzes, so we do not need to introduce any more parameters.

In Table 6.1, we compare the number of variational parameters d in the different variations of the QAOA that are simulated in this chapter. The standard QAOA uses the fewest number of variational parameters. The 2A-QAOA uses one more variational parameter than the VPS-QAOA to control the penalty strength. The MA-QAOA uses significantly more variational parameters than all of the other variants, even when the ansatz has only one layer. Note that the logical graph of the 2Q-PCP objective function is not fully connected, and d would be even larger for the MA-QAOA when solving a fully connected problem.

6.4 QAOA implementation

In the simulations in this chapter, we have made a variety of modifications to the implementation of the QAOA compared to the more standard implementation used in Chapter 4. These modifications are based on the results of previous work that have shown improvements in performance [149; 150; 151; 86]. We apply these modifications to all QAOA variants considered in this chapter. Here, we describe what these changes are.

This first modification is motivated by previous studies that have found that better values of the variational parameters can be found by providing a different cost function to the classical optimiser instead of $\langle f(\mathbf{x}) \rangle$ [149; 150]. The modified cost functions used in these studies give a higher weighting to sampled solutions with lower cost values. In the implementation used in this chapter, we define the cost function of the classical optimiser as the average residual energy of the best 20% of the sampled solutions. In other words, each time the quantum circuit is repeatedly sampled, we discard the 80% of samples with the highest objective values and calculate the average residual energy of the remaining solutions. The residual energy E_{res} is related to the objective function through Eq. (2.54). This definition of the cost function gives an incentive for the classical optimiser to increase the quality of the best sampled solutions at the expense of the quality of the other samples.

Another change compared to the implementation of the QAOA in Chapter 4 is that we use the SPSA method [125] instead of the COBYLA method as the classical optimiser. In each iteration, the COBYLA method evaluates the cost function at the vertices of a simplex in the parameter space in order to build a linear approximation of the cost function [124]. This means that for an ansatz with d parameters, the quantum circuit must be evaluated $d + 1$ times in each COBYLA iteration. In comparison, each iteration of the SPSA method involves evaluating the cost function at two perturbed positions along a randomly chosen direction

in the parameter space and calculating an approximate gradient of the cost [125]. Hence, the method requires two circuit evaluations per iteration, regardless of the number of parameters. This means that more iterations can be performed with SPSA than COBYLA given a fixed number of circuit evaluations when the number of variational parameters is large. Indeed, we perform 50,000 SPSA iterations for each simulation in this chapter, whereas the QAOA simulations in Chapter 4 used a maximum of 100 COBYLA iterations. The quantum circuit is sampled 1,000 times to estimate the cost value each time the variational parameters are changed by the SPSA method.

Finally, we have changed the initialisation strategy of the variational parameters to allow for both positive and negative initial values, inspired by previous work that has found that the optimal values of the angles β_k and γ_k often have opposite signs [151; 86]. Specifically, for all QAOA variants, we initialise the angles in $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ by selecting each value uniformly at random from the interval $[-1, 1)$. For the standard QAOA and the 2A-QAOA, the penalty strength α_2 is initialised uniformly at random in the interval $[0, 8)$. If linear penalties are used, α_1 is set to $-\alpha_2$. For the VPS-QAOA and MA-QAOA, we set $\alpha_2 = 4$, although this can effectively be rescaled by the variational parameters in these ansatzes. A penalty strength of $\alpha_2 = 4$ was also used in the definition of the cost function for all variants. For the VPS-QAOA and 2A-QAOA, we initialised the angles in $\boldsymbol{\delta}$ uniformly at random in the interval $[-1, 1)$. In this chapter, each run of an algorithm was performed with a single set of initial variational parameters for any particular problem instance. Different random initial parameters were chosen for each instance.

6.5 SPSA hyperparameters

The SPSA optimiser takes two hyperparameters, which can be varied in each iteration k. One of these is the perturbation size c_k, which specifies the magnitude of the perturbation that is used to calculate an approximate gradient of the cost.

The other is the learning rate a_k, which sets the scale of the parameter update step. That is, a_k controls how large a step to take in the direction that is assumed to have a negative gradient. In many implementations of the SPSA, a_k and c_k are set to decay according to the schedules

$$a_k = \frac{a}{(A + k + 1)^{\alpha}} \tag{6.5}$$

and

$$c_k = \frac{c}{(k + 1)^{\gamma}}, \tag{6.6}$$

where A is the stability constant, a is the numerator of the learning rate, c is the numerator of the perturbation size, α controls the decay rate of the learning rate, and γ controls the decay rate of the perturbation size [152; 123]. We set $\alpha = 0.602$ and $\gamma = 0.101$, which are commonly used values [152; 123]. We use the value $A = 0.1N_{\text{iter}}$, where N_{iter} is the total number of SPSA iterations, to match what is used in the PennyLane library [152].

In our numerical work, we set $a = c$, which effectively leaves one hyperparameter that needs to be assigned. We have independently tuned this value for each variant of the QAOA considered in this chapter. To do this, we have simulated each QAOA variant solving 100 instances of the 2Q-PCP with a and c set to 0.001, 0.01, 0.1, and 1. These simulations used quadratic penalty functions to encode all constraints. Fig. 6.1 plots the mean value of the SPSA optimiser's cost function against the iteration for each QAOA variant and each value of a and c. Solid lines show the mean costs for the perturbed variational parameters in each SPSA iteration. Points representing the mean costs for the final unperturbed variational parameters are also plotted. There is sometimes a large drop in mean cost for the final unperturbed variational parameters compared to the mean costs of the previous iterations. We believe this is because small perturbations in the variational parameters near local optima can have a large impact on performance, which means that the cost for the parameters at the unperturbed position can be much smaller in some cases.

We find that for the QAOA, VPS-QAOA, and 2A-QAOA, a value of $a = c = 0.01$ produces the lowest cost value on average. This is the value that we use for these variants in the rest of this chapter. Note that the points for the final mean costs using $a = c = 0.01$ and $a = c = 0.1$ in Fig. 6.1(b) for the VPS-QAOA are on top of each other and have overlapping standard errors. The hyperparameter value of $a = c = 0.01$ produces lower mean costs in most of the previous iterations, which is why we chose to use this value of a and c for the VPS-QAOA. For the MA-QAOA, $a = c = 0.01$ leads to better performance on average, so we use this value for the rest of the simulations of this variant. The MA-QAOA appears to be more sensitive to the values of a and c than the other variants, especially with $p = 6$ layers, where all other parameter values lead to minimal reductions in the cost function on average.

6.6 Performance comparison

We compare the performance of the different QAOA variants when solving 5,000 randomly selected instances of the 2Q-PCP with $n_p = 8$ products in simulations. The primary motivation is to understand whether the VPS-QAOA variant is useful.

6.6.1 Performance with quadratic penalties

In Fig. 6.2, we plot the average values of various performance metrics against the iteration of the SPSA optimiser for each QAOA variant listed in Table 6.1. Here, we have used the quadratic penalty method to encode all constraints in the definition of the cost function for the SPSA optimiser and in the implementations of the phase separator unitaries. Fig. 6.2(a) shows that after optimisation of the variational parameters, the cost function is lower on average with the VPS-QAOA than with the standard QAOA or the 2A-QAOA, although the differences are small relative to the final mean cost. The standard errors are small enough to show statistical significance in this result. This indicates that a modest performance improvement

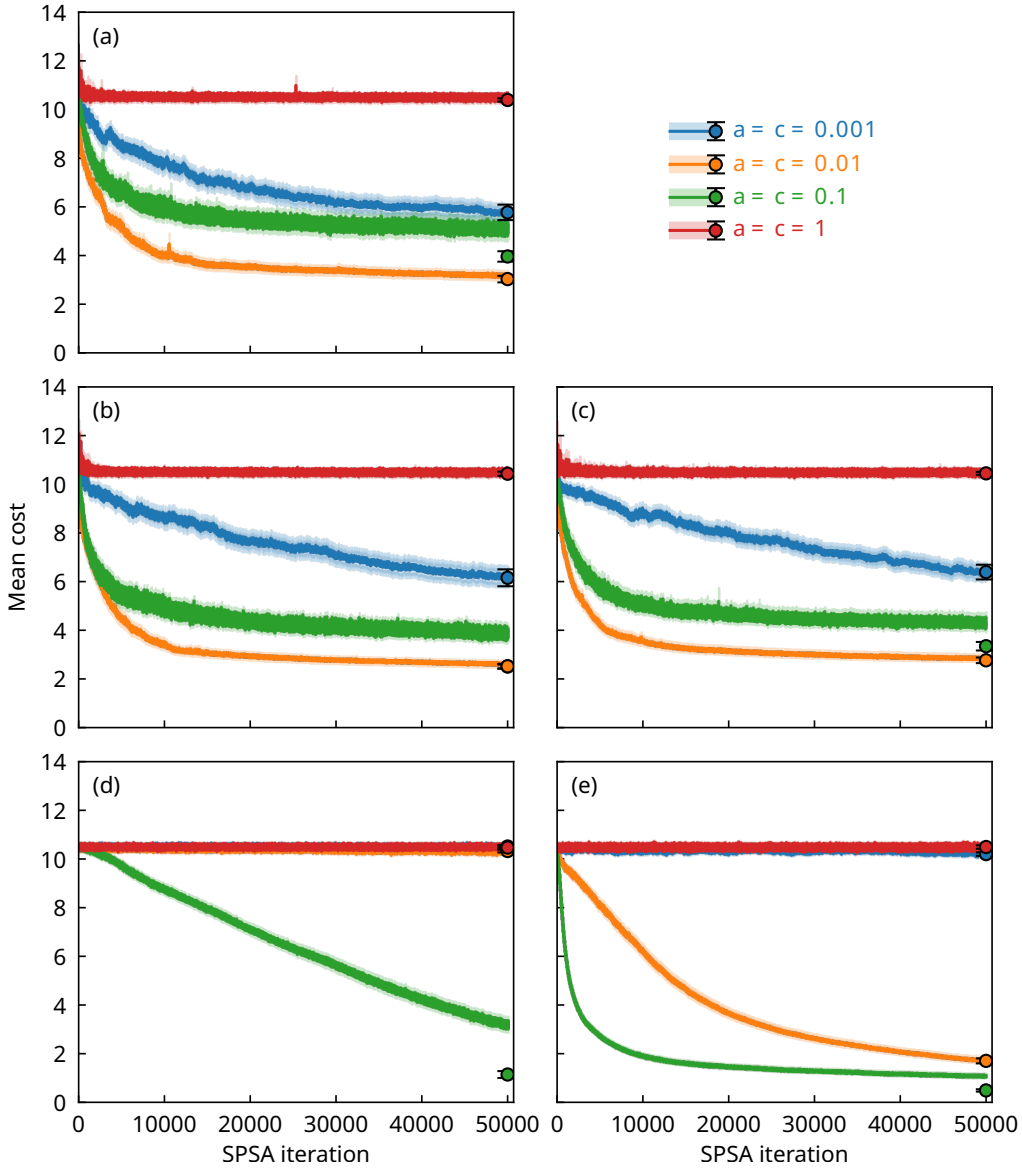


Figure 6.1: Average cost function value plotted against the SPSA iteration for the (a) QAOA, (b) VPS-QAOA, (c) 2A-QAOA, and (d) MA-QAOA with $p = 6$ layers as well as the (e) MA-QAOA with $p = 1$ layer when solving 100 instances of the 2Q-PCP with $n_p = 8$ products. Quadratic penalty functions are used to encode all constraints. The cost function is defined as the average QUBO objective value of the best 20% of sampled solutions. Solid lines show the mean costs for the perturbed variational parameters at each iteration. Lightly coloured shaded regions represent the standard error in the mean but are not visible for most lines. The mean costs for the final optimised parameters at the unperturbed positions are plotted as points with error bars representing the standard error in the mean. Different values of the SPSA hyperparameters a and c are shown in different colours. Different random initial values of the variational parameters are used for each one of the 100 instances.

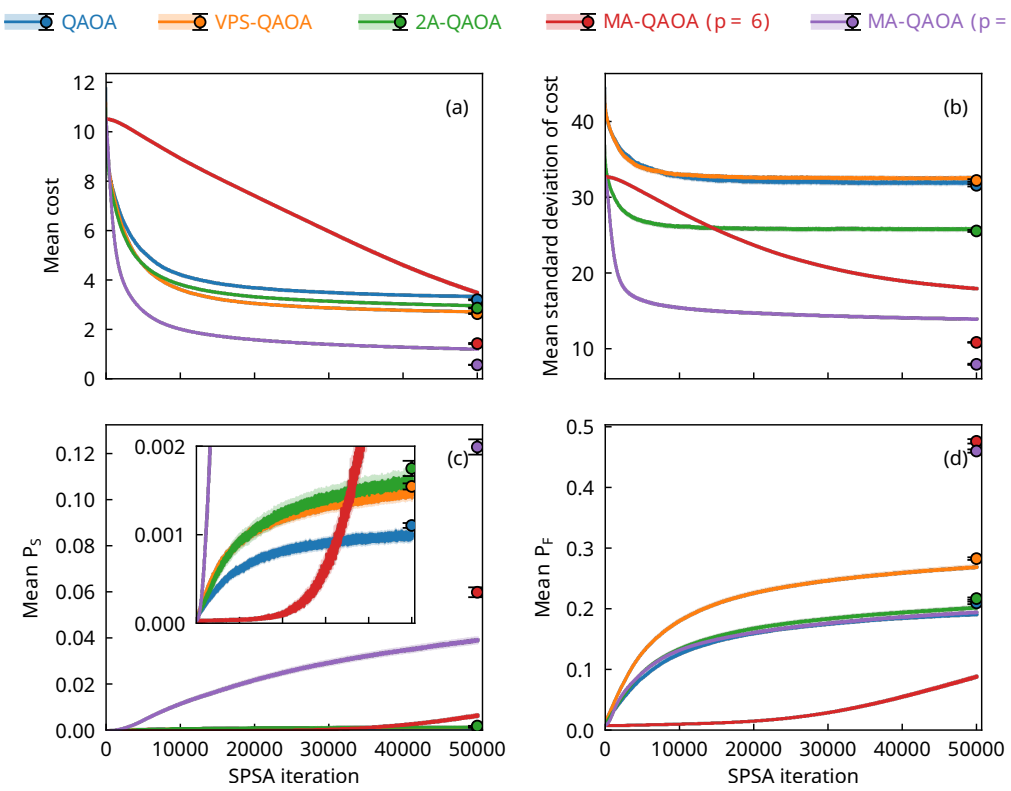


Figure 6.2: Comparison of the performance of different QAOA variants when solving the 2Q-PCP with $n_p = 8$ products using the quadratic penalty method to encode all constraints. At each iteration of the SPSA algorithm, we plot the (a) residual energy of the best 20% of sampled solutions, (b) standard deviation of residual energies, (c) inferred success probability P_S, and (d) inferred feasible probability P_F averaged over 5,000 different problem instances. We consider the standard QAOA (blue), VPS-QAOA (orange), 2A-QAOA (green), and MA-QAOA (red) with $p = 6$ layers as well as the MA-QAOA with $p = 1$ layer (purple). Solid lines show the average values for the perturbed variational parameters at each iteration. Lightly coloured shaded regions represent the standard error in the mean but are not visible for most lines. The mean values for the final optimised parameters at the unperturbed positions are plotted as points with error bars representing the standard error in the mean. Different random initial values of the variational parameters are used for each instance. The inset in (c) zooms into the bottom region of the plot.

could be achieved with the VPS-QAOA compared to the standard QAOA or a variant with a similar number of parameters, such as the 2A-QAOA.

The MA-QAOA with $p = 1$ layer achieves a significantly lower mean cost than all other ansatzes. This agrees with previous results showing that better performance can be achieved with this ansatz than the standard QAOA ansatz [84; 85; 86].

The MA-QAOA with $p = 6$ layers has the second lowest mean cost in the final iteration, despite the fact that it involves more layers and variational parameters than the $p = 1$ ansatz. Interestingly, this variant has the largest difference in mean cost between the last iteration with perturbed variational parameters and the final value with unperturbed parameters. This may be because the cost landscape for the MA-QAOA has the most narrow valleys at its local minima, which would mean that small perturbations from a minimum would produce large changes in the cost. The slope of the curve in Fig. 6.2(a) for the MA-QAOA with $p = 6$ remains steep until the final SPSA iteration. For the other variants, the slopes of the mean cost curves become shallow in later iterations, and the majority of the reduction in the mean cost occurs in the first 10,000 SPSA iterations. This indicates that the variational parameters of the MA-QAOA with $p = 6$ did not come as close to convergence as the parameters of the other ansatzes. Hence, it is likely that the MA-QAOA with $p = 6$ would perform significantly better if more SPSA iterations were used and potentially outperform the MA-QAOA with $p = 1$.

In Fig. 6.2(b) we plot the mean standard deviation of the cost, which is a measure of the variation in the quality of the sampled solutions. The final mean standard deviation is lowest for the MA-QAOA, which shows that this variant is better at producing solutions of similar quality each time the quantum circuit is sampled. The 2A-QAOA produces a lower mean standard deviation than the QAOA and the VPS-QAOA, which have similar values. Therefore, while the additional parameter introduced in the VPS-QAOA has benefitted performance, it appears that it has not significantly changed the variation in solution quality.

Looking at the mean success probabilities in Fig. 6.2(c), it is apparent that the MA-QAOA with $p = 1$ layer performs drastically better than the other variants, outperforming the standard QAOA by a factor of ≈ 110 on average with the final optimised variational parameters. With $p = 6$ layers, this factor drops to ≈ 54. The VPS-QAOA and 2A-QAOA show much more modest improvements in final success probability. Despite the VPS-QAOA producing a lower mean cost than

the 2A-QAOA, the 2A-QAOA produces a higher mean success probability. The VPS-QAOA outperforms the QAOA by a factor of ≈ 1.4, whereas the 2A-QAOA outperforms it by a factor of ≈ 1.6.

In Fig. 6.2(d), mean feasible probabilities are plotted, again showing the best performance with the MA-QAOA variant. This metric also shows a comparatively large improvement for the MA-QAOA variant when sampling the quantum circuit with the final unperturbed variational parameter values as opposed to the perturbed values. The VPS-QAOA has a significantly larger feasible probability than the QAOA and the 2A-QAOA, which indicates that associating the additional variational parameter with the penalty strength is particularly useful for increasing the feasible probability. For this problem, the feasible probability is relatively high for all QAOA variants. For example, the standard QAOA produces a mean feasible probability of $\langle P_F \rangle \approx 0.21$, compared to a mean success probability of $\langle P_S \rangle \approx 0.0011$. For reference, the probability of randomly guessing the optimal solution is $\frac{1}{2^{16}} \approx 2 \times 10^{-5}$. In cases where satisfying the constraints is more difficult, the ability of the VPS-QAOA to increase the feasible probability may become more beneficial for finding the optimal solution. Therefore, the VPS-QAOA variant may be more useful at larger problem sizes or for problems with more challenging constraints.

The fact that the MA-QAOA performs better with $p = 1$ than $p = 6$ in terms of success probability and cost is a demonstration that adding more variational parameters to an ansatz sometimes hurts performance by making it more difficult to classically optimise the parameters. In practice, picking the optimal number of variational parameters to use in an ansatz requires making a compromise between the ansatz's expressive power and the difficulty to optimise the parameters. The point at which the best compromise is made depends on various factors such as the problem size, number of layers, and the classical optimiser being used. The first two factors are constrained by the limitations of the quantum hardware being used. For example, the error rate places a limitation on the circuit depth and therefore

the number of layers, and the problem size is limited by the number of qubits available. At larger values of n or p, where there are more variational parameters in the MA-QAOA than in our examples, it may be best to use an ansatz with fewer variational parameters than the MA-QAOA but more parameters than the standard QAOA. The performance of the VPS-QAOA compared to the QAOA and the 2A-QAOA shows that this variant, or a similar variant involving variable penalties, is a good candidate for problems with difficult constraints.

We note that more intelligent strategies can be used for the optimisation of the variational parameters, which may work better for some variants than others. For example, a strategy has been suggested that optimises the parameters one layer at a time [153]. This approach begins with a single-layered ansatz and optimises its variational parameters. Then, a new layer is appended and its parameters are optimised while the previous layer's parameters are frozen. This is repeated until all p layers have been added and have optimised parameters. This strategy may produce better performance than the results we have presented, particularly for the MA-QAOA with $p = 6$, for which it appears that the classical optimiser did not come close to converging. Another interesting strategy, which could be a direction for future work, would be to start with the standard QAOA ansatz and continuously add more angles to the ansatz while using the optimised parameter values of the previous ansatz as the initial angles for the next ansatz. For example, the standard QAOA ansatz could be used to provide a warm start for the 2A-QAOA ansatz, which could be used as a warm start for an ansatz with more angles, and so on.

6.6.2 Performance with linear penalties

We have also performed simulations of some of the QAOA variants with the constraints C1 encoded with linear penalties in the ansatz. We have still used the quadratic penalty method to encode all constraints in the cost function that is classically optimised. The aim is to assess whether the performance improvements observed in Sec. 4.6 from using linear penalties carry over to the QAOA variants

considered in this chapter with the implementation changes discussed in Sec. 6.4. We have not simulated the MA-QAOA with linear penalties. However, we note that since the linear penalty method produces Ising Hamiltonians with sparser logical graphs for this problem, the MA-QAOA ansatz would require fewer variational parameters in the phase separator unitary. This may benefit the algorithm by making the classical optimisation of the parameters easier, which would be interesting to investigate in future work.

In Fig. 6.3, we plot the mean values of the same performance metrics that are plotted in Fig. 6.2 for the three different QAOA variants we have simulated with linear penalties. The mean values with the final optimised variational parameters for the simulations using all-quadratic penalties are also shown for comparison. One observation is that the mean values of all performance metrics with optimised variational parameters are statistically significantly improved by the use of linear penalties for all three QAOA variants we have considered. This demonstrates that the linear penalty method can complement other algorithmic techniques for improving the performance of quantum optimisation algorithms.

Fig. 6.3(c) shows that the VPS-QAOA produced the largest mean success probability of the three QAOA variants with linear penalties. However, the error bars for the final value with optimised variational parameters overlap with the error bars for the 2A-QAOA. In comparison, with all-quadratic penalties, the 2A-QAOA was the best performing variant. This shows that using linear penalties with variable penalty strengths together is particularly effective. This is supported by Fig. 6.3(b), which shows that the VPS-QAOA has the largest decrease in the variability of the quality of sampled solutions from changing quadratic penalties to linear penalties. With all-quadratic penalties, the mean standard deviation of the cost for the VPS-QAOA is similar to that of the QAOA, whereas with linear penalties it is similar to that of the 2A-QAOA.

Comparing Fig. 6.3(a) and Fig. 6.2(a), we find that the mean costs drop more suddenly in the earlier SPSA iterations when using linear penalties compared to using

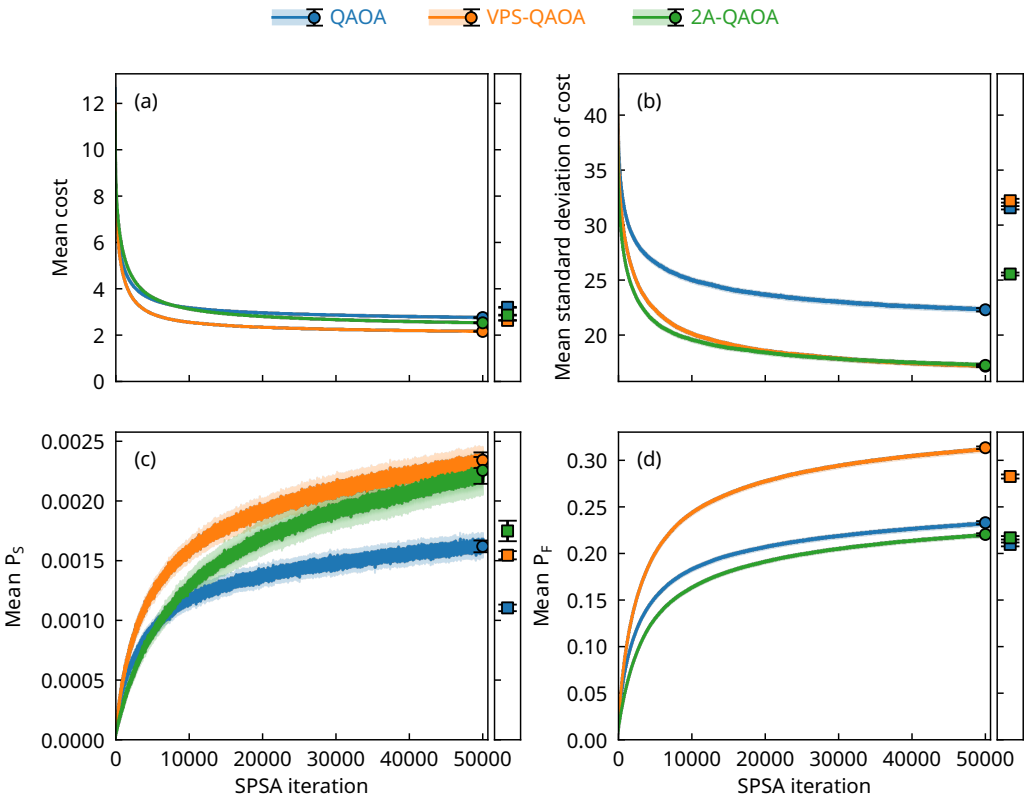


Figure 6.3: Comparison of the performance of different QAOA variants when solving the 2Q-PCP with $n_p = 8$ products using the linear penalty method to encode the constraints C1. At each iteration of the SPSA algorithm, we plot the (a) residual energy of the best 20% of sampled solutions, (b) standard deviation of residual energies, (c) inferred success probability P_S, and (d) inferred feasible probability P_F averaged over 5,000 different problem instances. We consider the standard QAOA (blue), VPS-QAOA (orange), and 2A-QAOA (green) with $p = 6$ layers. Solid lines show the average values for the perturbed variational parameters at each iteration. Lightly coloured shaded regions represent the standard error in the mean but are not visible for most lines. The mean values for the final optimised parameters at the unperturbed positions are plotted with circular markers and error bars representing the standard error in the mean. On separate axes on the right side of each plot, the mean values for the final optimised parameters when using all-quadratic penalties are plotted with square markers for comparison. Different random initial values of the variational parameters are used for each instance.

all-quadratic penalties. This indicates that optimising the variational parameters is easier with linear penalties than with quadratic penalties, and ansatzes with more variational parameters could be used with linear penalties while still being able to find good values of the parameter values. Furthermore, there are not any large changes in any of the plotted values in Fig. 6.2 when the final unperturbed parameters are used instead of the perturbed parameters compared to the data for the same variants in Fig. 6.2. This may be because the local minima in the cost landscape are wider when linear penalties are used, which is another reason to think that optimising the variational parameters is easier with linear penalties.

6.7 Analysis of variational parameters

Here, we analyse the values of the variational parameters for the VPS-QAOA and the MA-QAOA after they have been optimised with the SPSA method. We use the data from the simulations using all-quadratic penalties to solve the 2Q-PCP, which we also analysed in Sec. 6.6.1. We note that while the optimised parameter values are good in the sense that they produce significantly better performance than randomly chosen parameter values, they are not necessarily the global optimal values.

6.7.1 VPS-QAOA variational parameters

In Fig. 6.4(a), we plot the mean absolute value of each optimised parameter in the VPS-QAOA against the layer l of the ansatz. The reason we take the absolute value is that otherwise, the averages would be close to zero. It can be seen that the mean absolute value of β_l decreases with l while the mean absolute values of the parameters γ_l and δ_l remain elevated. This resembles the standard QA schedule, in which the magnitude of the mixer Hamiltonian is reduced over time compared to the magnitude of the Ising Hamiltonian. This provides some support to previous findings that indicate that fixing the QAOA parameters based on a

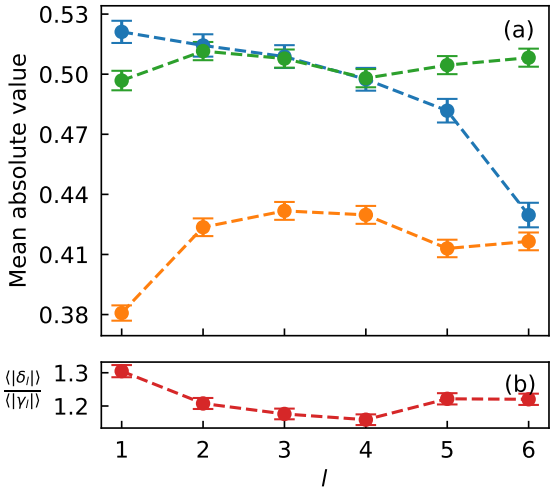


Figure 6.4: (a) Mean absolute value of the variational parameters β_l (blue), γ_l, (orange), and δ_l (green) of the VPS-QAOA plotted against the layer l of the ansatz. These parameters have been optimised by the SPSA method for 50,000 instances of the 2Q-PCP with 8 products, using quadratic penalties to encode constraints. Error bars show the standard error in the mean. (b) Ratio between the mean absolute values of δ_l and γ_l plotted against l with error bars that are propagated from the standard errors [155]. Dashed lines connect the points to guide the eye.

typical QA schedule can be an effective strategy that avoids the need for variational parameters [154]. However, we point out that the values plotted in Fig. 6.4(a) are averages over different simulations and the optimised parameter values for a particular instance may not necessarily follow this trend.

Fig. 6.4(b) plots $\frac{\langle|\delta_l|\rangle}{\langle|\gamma_l|\rangle}$, which represents the amount by which penalties are scaled in each layer l. This plot shows that, on average, the magnitude of the penalty strength is highest in the first layer, then decreases, and then increases again in the last two layers. The error bars show that these changes are statistically significant. The fact that the optimised penalty strengths are statistically significantly different between the different layers is another indication that there is some benefit to being able to vary the penalty strength. These results indicate that fixing the variable penalty strengths to be strongest in the initial and final layers could be a good strategy.

6.7.2 MA-QAOA variational parameters

According to [89], it was observed in [156] that optimised MA-QAOA variational parameters cluster around multiples of $\frac{\pi}{4}$. The analysis in [156] was performed on the MaxCut problem, in which case there are no local fields in $\mathcal{H}_I$. As far as we are aware, our work is the first to study the MA-QAOA in the context of a problem that has nonzero local fields in $\mathcal{H}_I$. It would therefore be interesting to see if the clustering of angles around multiples of $\frac{\pi}{4}$ can also be seen in the results of our simulations.

In Fig. 6.5, we show histograms of the frequency density of optimised variational parameter values for the MA-QAOA with $p = 1$ layer. Separate histograms are shown for the parameters in $\boldsymbol{\beta}$, the parameters in $\boldsymbol{\gamma}$ that correspond to couplings, and the parameters in $\boldsymbol{\gamma}$ that correspond to local fields. We find that the parameters $\boldsymbol{\beta}$ are strongly clustered around multiples of $\frac{\pi}{4}$, agreeing with the results in [156]. The parameters in $\boldsymbol{\gamma}$ that correspond to couplings show some clustering near the same areas, but it appears these clusters are centred around multiples of a value that is slightly less than $\frac{\pi}{4}$. There are also smaller clusters of these parameters at other values.

The histogram in Fig. 6.5 for the parameters in $\boldsymbol{\gamma}$ that correspond to local fields has a more complicated structure than the other histograms. Between $-\frac{\pi}{4}$ and $\frac{\pi}{4}$, there are many clusters of parameter values. They appear to be periodic, with a period of $\approx \frac{\pi}{28}$. There is also a pattern in the relative sizes of clusters. Specifically, they alternate between having higher and lower frequency density peaks as the parameter value is changed. Outside the region between $-\frac{\pi}{4}$ and $\frac{\pi}{4}$, much of the structure is lost.

These results show that the clustering of values around multiples of $\frac{\pi}{4}$ can be observed in our simulations for the parameters $\boldsymbol{\beta}$, but the parameters $\boldsymbol{\gamma}$ show a different pattern. This is likely due to the fact that we are analysing a different problem to what was analysed in [156] that has nonzero local fields in the Ising

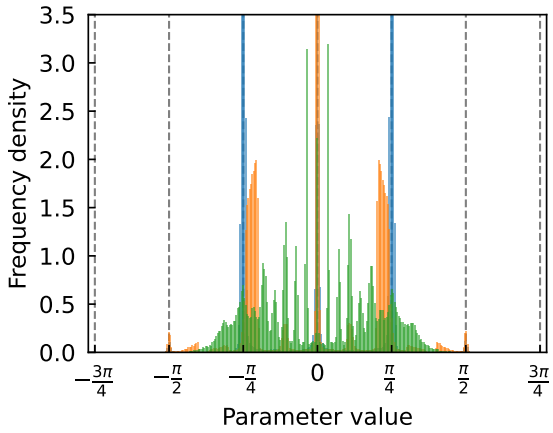


Figure 6.5: Histograms of the frequency density of variational parameter values in the MA-QAOA with $p = 1$ layer after optimisation with the SPSA method for the 2Q-PCP problem with 8 products. Separate histograms are shown for the parameters $\boldsymbol{\beta}$ (blue), the parameters in $\boldsymbol{\gamma}$ that correspond to couplings (orange), and the parameters in $\boldsymbol{\gamma}$ that correspond to local fields (green). The histograms contain all parameters for 5,000 different problem instances. Note that the y-axis limits cut off some peaks in order to make the rest of the data more visible. In particular, the central orange peak has a frequency density of ≈ 16 and the blue peaks at $-\frac{\pi}{4}$ and $\frac{\pi}{4}$ have frequency densities of ≈ 10.5.

Hamiltonian. Nevertheless, we still observe a significant amount of clustering in all variational parameters, which indicates that it may be easier to classically optimise them by exploiting this structure, as suggested in previous work [89].

We have also analysed the optimised variational parameter values of the MA-QAOA with $p = 6$ layers. Frequency density histograms of the parameter values are shown in Fig. 6.6. The variational parameters for each layer of the ansatz are shown in a separate plot. Fig. 6.6(a) for the parameters in the first layer looks similar to Fig. 6.5 for the $p = 1$ case. In Fig. 6.6(a), there are clusters in the same positions, but they are less pronounced. The plots for subsequent layers in Fig. 6.6 show much less structure, although the clustering of the parameters in $\boldsymbol{\beta}$ is still prominent.

The lack of clustering in Fig. 6.6 compared to Fig. 6.5 may be because SPSA optimiser did not get close to the optimal variational parameter values for the MA-QAOA with $p = 6$. We also suggested this in Sec. 6.6.1 when noting the apparent lack of convergence of the mean cost for the $p = 6$ MA-QAOA in Fig. 6.2(a). It is

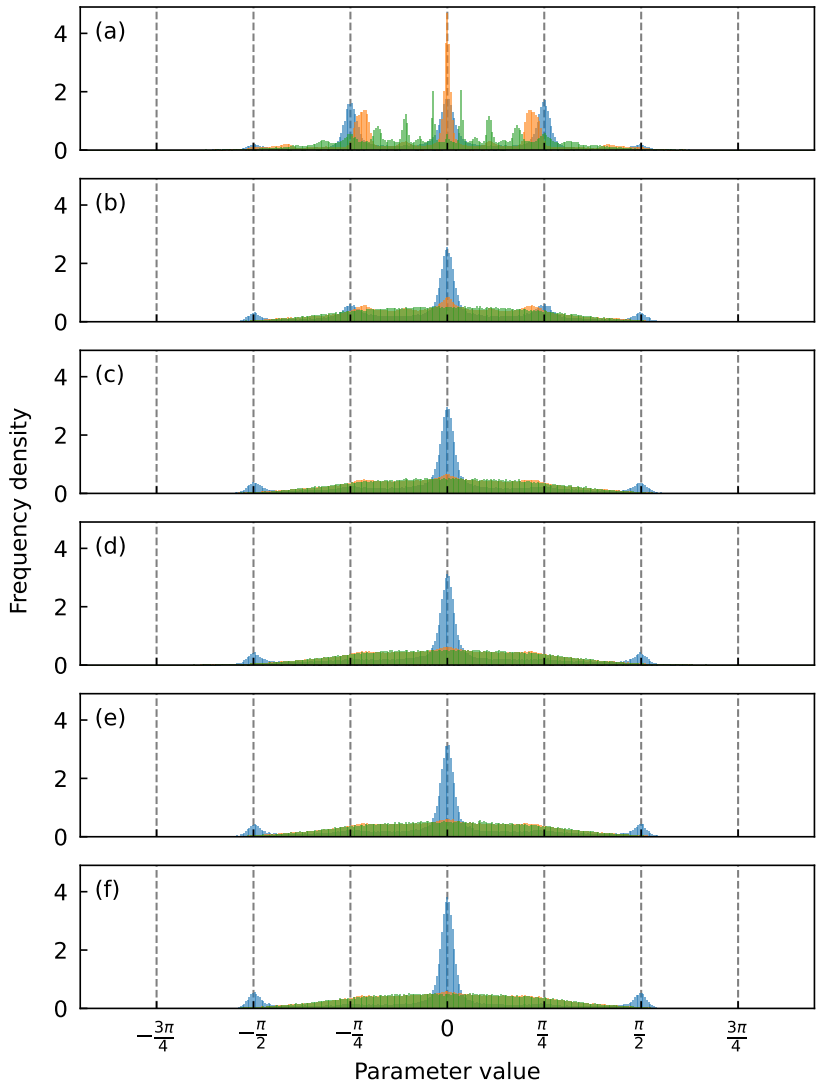


Figure 6.6: Histograms of the frequency density of variational parameter values in the MA-QAOA with $p = 6$ layers after optimisation with the SPSA method for the 2Q-PCP problem with 8 products. Different plots are shown for the ansatz layers (a) $l = 1$, (b) $l = 2$, (c) $l = 3$, (d) $l = 4$, (e) $l = 5$, and (f) $l = 6$. In each plot, separate histograms are shown for the parameters in $\boldsymbol{\beta}$ (blue), the parameters in $\boldsymbol{\gamma}$ that correspond to couplings (orange), and the parameters in $\boldsymbol{\gamma}$ that correspond to local fields (green). The histograms contain all parameters for 5,000 different problem instances.

interesting that with a limited number of iterations, the optimiser appears to have made more progress in optimising the variational parameters of the first layer of the ansatz than the other layers.

6.8 Chapter conclusions

In this chapter, we have introduced a new variant of the QAOA that we refer to as the VPS-QAOA, which uses additional variational parameters to implement variable penalty strengths. We have found that it gives a modest improvement in performance compared to the standard QAOA in our simulations. While this chapter is only concerned with the gate-based QAOA algorithm and its variants, the concept of variational penalty strengths could be extended to the continuous-time setting of QA in future work. Comparing the VPS-QAOA to another QAOA variant with a similar number of variational parameters shows that the main strength of the VPS-QAOA is in increasing the probability of satisfying constraints, which implies that the variant is particularly suitable for problems that are highly constrained or have constraints that are difficult to satisfy. We have also demonstrated that the linear penalty method can be used alongside the variable penalty strength approach, and this further improves performance in our simulations.

A comparison with the MA-QAOA shows that by using more variational parameters, the MA-QAOA can achieve significantly better performance than the QAOA and the VPS-QAOA. However, our results show that the SPSA optimiser struggles to optimise the MA-QAOA variational parameters as the number of layers is increased, whereas the other variants showed signs of convergence of their variational parameters with the same number of layers. This implies that for larger numbers of layers or larger problem sizes, where the MA-QAOA ansatz has more variational parameters, it may be preferable to use a variant like the VPS-QAOA that does not introduce as many additional parameters but is still able to outperform the standard QAOA.

Finally, we have analysed the values of the optimised variational parameters for the VPS-QAOA and the MA-QAOA. We have found that on average, the optimiser chooses variable penalty strengths that are weaker in the middle layers than in the first and last layers, with the first layer having the strongest penalty strength. For the MA-QAOA, we have found that the variational parameters in the mixer unitary cluster around multiples of $\frac{\pi}{4}$, which has previously been reported [89; 156]. The variational parameters in the phase separator unitary also form some clusters but have a more complicated structure.

Chapter 7

Conclusions

In this thesis, we have studied algorithmic techniques for improving the performance of constrained quantum optimisation. The quantum algorithms and example customer data science problems that our work is based on were introduced in Chapter 2.

In Chapter 3, we explored the use of linear Ising penalties for encoding Hamming weight equality constraints. We explained that linear penalties avoid some of the drawbacks of the quadratic penalty method, which is the standard approach to encoding constraints in quantum optimisation methods such as QA and the QAOA. In particular, linear penalties do not introduce any additional couplings to the Ising Hamiltonian and often involve smaller energy scales than quadratic penalties. Changing the penalty strength of a linear penalty changes the Hamming weight of the associated variables in the ground state of the Ising Hamiltonian in a monotonic fashion. Based on this observation, we outlined a search strategy for efficiently finding a good penalty strength value for cases where a single linear penalty is applied. When multiple linear penalties are applied, we explained that their penalty strengths need to be tuned individually, and we proposed a strategy to do so. In cases where not all linear penalties are effective in implementing their constraints, we showed that it is possible to switch some of the linear penalties to quadratic penalties and produce the desired ground state in the Ising Hamiltonian.

In Chapter 4, we compared the linear and quadratic penalty methods in numerical simulations of QA and the QAOA. In these simulations, physical qubits were assumed to be all-to-all connected, which means that there were no minor embedding considerations. Therefore, the main benefit of the linear penalty method in this scenario is its efficient use of the available dynamic range. We found that for the problem instances we considered, using the linear penalty method led to a less severe normalisation of the Ising Hamiltonian than when using the quadratic penalty method, confirming the improvement in effective dynamic range. Our results showed that QA and QAOA performance benefitted from the use of linear penalties on average. This is an encouraging sign that the linear penalty method is useful even when we are not limited by the interaction graph of the quantum hardware.

In Chapter 5, we performed experimental tests of QA on a D-Wave annealer, using the quadratic and linear penalty methods. We ran experiments with the 1Q-PCP, which involves a single constraint, and the 4Q-PCP, which involves many constraints. The large number of qubits on the annealer's QPU allowed us to consider larger problem sizes than those that were simulated in Chapter 4. For both problems, we observed clear improvements in performance when using the linear penalty method on problem instances that it could exactly constrain. For the 1Q-PCP, the difference in minor embedding efficiency between the two penalty methods was drastic, and this was likely the main reason for the better performance of the linear penalty method. In our tests with the 4Q-PCP, we found signs that improvements in the effective dynamic range had a noticeable impact on performance. For example, the quantum annealer found significantly higher quality solutions when all four constraints on the number of promotions per quarter could be implemented with linear penalties than when only two of the constraints could be. In this case, the difference in dynamic range efficiency between the two penalty schemes appears to be more substantial than the difference in minor embedding efficiency.

Our findings indicate that the linear penalty method could play a role in enhancing the performance of quantum optimisation algorithms when applied to certain

constraints. Future research applying the linear penalties to problems with different structures than that of the promotion cannibalisation problems we considered would be beneficial in identifying the most suitable applications of the linear penalty method. As quantum computing hardware matures, it would be interesting to test the performance of the linear penalty method on larger numbers of variables and on different hardware platforms. Another potential direction for future research is to determine whether the linear penalty method can be applied to constraints that are not k-hot constraints.

In Chapter 6, we introduced the VPS-QAOA, which modifies the standard QAOA ansatz to allow the penalty strengths to be varied between the layers of the quantum circuit. We found modest performance improvements of the VPS-QAOA over the standard QAOA in numerical simulations on the 2Q-PCP. Comparing the VPS-QAOA to another ansatz that involves a similar number of variational parameters only shows an improvement in the ability to satisfy constraints, not in the ability to find optimal solutions. We argue that in scenarios where the constraints are harder to satisfy, the VPS-QAOA could have an advantage in finding optimal solutions. We demonstrated that the linear penalty method can be used in conjunction with the VPS-QAOA and that combining the approaches works well in practice. We also performed simulations of the MA-QAOA, which is a variant of the QAOA that has many more variational parameters than the other variants we considered. Our results showed that the MA-QAOA achieved significantly better performance than the other approaches. However, we observed signs that the larger numbers of variational parameters in the MA-QAOA are harder to optimise. This implies that as the problem size or number of layers is increased, it may be better to use variants with fewer variational parameters. Hence, variants like the MA-QAOA may be desirable as they can improve performance without introducing as many variational parameters.

A clear direction for future work on the VPS-QAOA is to apply it to problems other than the 2Q-PCP. Since our results showed that the main advantage of us-

ing variable penalty strengths was in increasing the feasible probability, it would be interesting to see if the approach is more effective when applied to problems with constraints that are more difficult to satisfy. The VPS-QAOA ansatz that we considered in our numerical simulations only introduced a single additional variational parameter, but it is likely that better performance could be achieved by assigning different variational parameters to different constraints. Our analysis of the MA-QAOA showed that it is important to consider the trade-off between the expressiveness of the ansatz and the difficulty to optimise the variational parameters, which is in agreement with previous work. A comparison of the VPS-QAOA and other QAOA variants with different numbers of variational parameters on real quantum hardware would provide more insight into what the best trade-off is in practice. As quantum hardware matures and practical limits on circuit depth and numbers of qubits are pushed further, it is likely that the best choice of ansatz will change. The concept of variable penalty strengths is not just relevant for the QAOA—it could be applied in other situations where penalty functions are used, such as QA.

Bibliography

[1] P. Mirkarimi, I. Shukla, D. C. Hoyle, R. Williams, and N. Chancellor. Quantum optimization with linear Ising penalty functions for customer data science. *Physical Review Research*, 6(4):043241, 2024. doi:10.1103/PhysRevResearch.6.043241.

[2] P. Mirkarimi, D. C. Hoyle, R. Williams, and N. Chancellor. Experimental demonstration of improved quantum optimization with linear Ising penalties. *New Journal of Physics*, 26(10):103005, 2024. doi:10.1088/1367-2630/ad7e4a.

[3] P. 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. doi:10.1007/BF01011339.

[4] P. Benioff. Quantum Mechanical Models of Turing Machines That Dissipate No Energy. *Physical Review Letters*, 48(23):1581–1585, 1982. doi:10.1103/PhysRevLett.48.1581.

[5] Y. Manin. *Vychislimoe i nevychislimoe*. Sovetskoe Radio, 1980.

[6] A. Y. Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997. doi:10.1070/RM1997v052n06ABEH002155.

[7] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, 1982. doi:10.1007/BF02650179.

[8] P. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. doi:10.1109/SFCS.1994.365700.

[9] L. K. Grover. A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 212–219, 1996. doi:10.1145/237814.237866.

[10] C. Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746–2751, 1999. doi:10.1103/PhysRevA.60.2746.

[11] A. Gilliam, S. Woerner, and C. Gonciulea. Grover Adaptive Search for Constrained Polynomial Binary Optimization. *Quantum*, 5:428, 2021. doi:10.22331/q-2021-04-08-428.

[12] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. doi:10.1038/s41586-019-1666-5.

[13] 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. doi:10.1038/nature10012.

[14] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis. Demonstration of the trapped-ion quantum CCD computer architecture. *Nature*, 592(7853):209–213, 2021. doi:10.1038/s41586-021-03318-4.

[15] S. Ebadi, T. T. Wang, H. Levine, A. Keesling, G. Semeghini, A. Omran, D. Bluvstein, R. Samajdar, H. Pichler, W. W. Ho, S. Choi, S. Sachdev, M. Greiner, V. Vuletić, and M. D. Lukin. Quantum phases of matter on a 256-atom programmable quantum simulator. *Nature*, 595(7866):227–232, 2021. doi:10.1038/s41586-021-03582-4.

[16] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu, P. Hu, X.-Y. Yang, W.-J. Zhang, H. Li, Y. Li, X. Jiang, L. Gan, G. Yang, L. You, Z. Wang, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020. doi:10.1126/science.abe8770.

[17] G. S. Ravi, K. N. Smith, P. Gokhale, and F. T. Chong. Quantum Computing in the Cloud: Analyzing job and machine characteristics. *Proceedings - 2021 IEEE International Symposium on Workload Characterization, IISWC 2021*, pages 39–50, 2021. doi:10.1109/IISWC53511.2021.00015.

[18] A. D. King, A. Nocera, M. M. Rams, J. Dziarmaga, R. Wiersema, W. Bernoudy, J. Raymond, N. Kaushal, N. Heinsdorf, R. Harris, K. Boothby,

F. Altomare, M. Asad, A. J. Berkley, M. Boschnak, K. Chern, H. Christiani, S. Cibere, J. Connor, M. H. Dehn, R. Deshpande, S. Ejtemaee, P. Farre, K. Hamer, E. Hoskinson, S. Huang, M. W. Johnson, S. Kortas, E. Ladizinsky, T. Lanting, T. Lai, R. Li, A. J. R. MacDonald, G. Marsden, C. C. McGeoch, R. Molavi, T. Oh, R. Neufeld, M. Norouzpour, J. Pasvolsky, P. Poitras, G. Poulin-Lamarre, T. Prescott, M. Reis, C. Rich, M. Samani, B. Sheldan, A. Smirnov, E. Sterpka, B. Trullas Clavera, N. Tsai, M. Volkmann, A. M. Whiticar, J. D. Whittaker, W. Wilkinson, J. Yao, T. J. Yi, A. W. Sandvik, G. Alvarez, R. G. Melko, J. Carrasquilla, M. Franz, and M. H. Amin. Beyond-classical computation in quantum simulation. *Science*, 388(6743): 199–204, 2025. doi:10.1126/science.ado6285.

[19] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. Bonilla Ataides, N. Maskara, I. Cong, X. Gao, P. Sales Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin. Logical quantum processor based on reconfigurable atom arrays. *Nature*, 626(7997): 58–65, 2024. doi:10.1038/s41586-023-06927-3.

[20] R. Acharya, D. A. Abanin, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya, R. Babbush, D. Bacon, B. Ballard, J. C. Bardin, J. Bausch, A. Bengtsson, A. Bilmes, S. Blackwell, S. Boixo, G. Bortoli, A. Bourassa, J. Bovaird, L. Brill, M. Broughton, D. A. Browne, B. Buchea, B. B. Buckley, D. A. Buell, T. Burger, B. Burkett, N. Bushnell, A. Cabrera, J. Campero, H.-S. Chang, Y. Chen, Z. Chen, B. Chiaro, D. Chik, C. Chou, J. Claes, A. Y. Cleland, J. Cogan, R. Collins, P. Conner, W. Courtney, A. L. Crook, B. Curtin, S. Das, A. Davies, L. De Lorenzo, D. M. Debroy, S. Demura, M. Devoret, A. Di Paolo, P. Donohoe, I. Drozdov, A. Dunsworth, C. Earle, T. Edlich, A. Eickbusch, A. M. Elbag, M. Elzouka, C. Erickson, L. Faoro, E. Farhi, V. S. Ferreira, L. F. Burgos, E. Forati, A. G. Fowler, B. Foxen, S. Gan-

jam, G. Garcia, R. Gasca, É. Genois, W. Giang, C. Gidney, D. Gilboa, R. Gosula, A. G. Dau, D. Graumann, A. Greene, J. A. Gross, S. Habegger, J. Hall, M. C. Hamilton, M. Hansen, M. P. Harrigan, S. D. Harrington, F. J. H. Heras, S. Heslin, P. Heu, O. Higgott, G. Hill, J. Hilton, G. Holland, S. Hong, H.-Y. Huang, A. Huff, W. J. Huggins, L. B. Ioffe, S. V. Isakov, J. Iveland, E. Jeffrey, Z. Jiang, C. Jones, S. Jordan, C. Joshi, P. Juhas, D. Kafri, H. Kang, A. H. Karamlou, K. Kechedzhi, J. Kelly, T. Khaire, T. Khattar, M. Khezri, S. Kim, P. V. Klimov, A. R. Klots, B. Kobrin, P. Kohli, A. N. Korotkov, F. Kostritsa, R. Kothari, B. Kozlovskii, J. M. Kreikebaum, V. D. Kurilovich, N. Lacroix, D. Landhuis, T. Lange-Dei, B. W. Langley, P. Laptev, K.-M. Lau, L. Le Guevel, J. Ledford, J. Lee, K. Lee, Y. D. Lensky, S. Leon, B. J. Lester, W. Y. Li, Y. Li, A. T. Lill, W. Liu, W. P. Livingston, A. Locharla, E. Lucero, D. Lundahl, A. Lunt, S. Madhuk, F. D. Malone, A. Maloney, S. Mandrà, J. Manyika, L. S. Martin, O. Martin, S. Martin, C. Maxfield, J. R. McClean, M. McEwen, S. Meeks, A. Megrant, X. Mi, K. C. Miao, A. Mieszala, R. Molavi, S. Molina, S. Montazeri, A. Morvan, R. Movassagh, W. Mruczkiewicz, O. Naaman, M. Neeley, C. Neill, A. Nersisyan, H. Neven, M. Newman, J. H. Ng, A. Nguyen, M. Nguyen, C.-H. Ni, M. Y. Niu, T. E. O'Brien, W. D. Oliver, A. Opremcak, K. Ottosson, A. Petukhov, A. Pizzuto, J. Platt, R. Potter, O. Pritchard, L. P. Pryadko, C. Quintana, G. Ramachandran, M. J. Reagor, J. Redding, D. M. Rhodes, G. Roberts, E. Rosenberg, E. Rosenfeld, P. Roushan, N. C. Rubin, N. Saei, D. Sank, K. Sankaragomathi, K. J. Satzinger, H. F. Schurkus, C. Schuster, A. W. Senior, M. J. Shearn, A. Shorter, N. Shutty, V. Shvarts, S. Singh, V. Sivak, J. Skruzny, S. Small, V. Smelyanskiy, W. C. Smith, R. D. Somma, S. Springer, G. Sterling, D. Strain, J. Suchard, A. Szasz, A. Sztein, D. Thor, A. Torres, M. M. Torunbalci, A. Vaishnav, J. Vargas, S. Vdovichev, G. Vidal, B. Villalonga, C. V. Heidweiller, S. Waltman, S. X. Wang, B. Ware, K. Weber, T. Weidel, T. White, K. Wong, B. W. K. Woo, C. Xing, Z. J. Yao, P. Yeh,

B. Ying, J. Yoo, N. Yosri, G. Young, A. Zalcman, Y. Zhang, N. Zhu, and N. Zobrist. Quantum error correction below the surface code threshold. *Nature*, 638(8052):920–926, 2025. doi:10.1038/s41586-024-08449-y.

[21] P. Cazals, A. François, L. Henriet, L. Leclerc, M. Marin, Y. Naghmouchi, W. d. S. Coelho, F. Sikora, V. Vitale, R. Watrigant, M. W. Garzillo, and C. Dalyac. Identifying hard native instances for the maximum independent set problem on neutral atoms quantum processors, 2025. URL `http://arxiv.org/abs/2502.04291`. arXiv preprint arXiv:2502.04291.

[22] H. Munoz-Bauza and D. Lidar. Scaling Advantage in Approximate Optimization with Quantum Annealing. *Physical Review Letters*, 134(16):160601, 2025. doi:10.1103/PhysRevLett.134.160601.

[23] R. Orús, S. Mugel, and E. Lizaso. Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4:100028, 2019. doi:10.1016/j.revip.2019.100028.

[24] D. Venturelli and A. Kondratyev. Reverse quantum annealing approach to portfolio optimization problems. *Quantum Machine Intelligence*, 1(1-2):17–30, 2019. doi:10.1007/s42484-019-00001-w.

[25] D. M. Fox, K. M. Branson, and R. C. Walker. mRNA codon optimization with quantum computers. *PLOS ONE*, 16(10):1–16, 2021. doi:10.1371/journal.pone.0259101.

[26] K. Kitai, J. Guo, S. Ju, S. Tanaka, K. Tsuda, J. Shiomi, and R. Tamura. Designing metamaterials with quantum annealing and factorization machines. *Physical Review Research*, 2(1):013319, 2020. doi:10.1103/PhysRevResearch.2.013319.

[27] T. Stollenwerk, B. O'Gorman, D. Venturelli, S. Mandra, O. Rodionova, H. Ng, B. Sridhar, E. G. Rieffel, and R. Biswas. Quantum Annealing Applied to De-Conflicting Optimal Trajectories for Air Traffic Management.

IEEE Transactions on Intelligent Transportation Systems, 21(1):285–297, 2020. doi:10.1109/TITS.2019.2891235.

[28] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt. Quantum annealing for industry applications: introduction and review. *Reports on Progress in Physics*, 85(10):104001, 2022. doi:10.1088/1361-6633/ac8c54.

[29] K. Czerniachowska. A genetic algorithm for the retail shelf space allocation problem with virtual segments. *OPSEARCH*, 59(1):364–412, 2022. doi:10.1007/s12597-021-00551-3.

[30] S. Subramanian and H. D. Sherali. A fractional programming approach for retail category price optimization. *Journal of Global Optimization*, 48(2): 263–277, 2010. doi:10.1007/s10898-009-9491-2.

[31] P. Mirkarimi, I. Shukla, D. C. Hoyle, R. Williams, and N. Chancellor. Quantum optimization with linear ising penalty functions for customer data science [dataset]. 2024. doi:http://doi.org/10.15128/r2fq977t82m.

[32] P. Mirkarimi, D. C. Hoyle, R. Williams, and N. Chancellor. Experimental demonstration of improved quantum optimization with linear ising penalties [dataset]. 2024. doi:http://doi.org/10.15128/r2j6731386t.

[33] S. G. Brush. History of the Lenz-Ising Model. *Reviews of Modern Physics*, 39(4):883–893, 1967. doi:10.1103/RevModPhys.39.883.

[34] A. Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2: 1–14, 2014. doi:10.3389/fphy.2014.00005.

[35] B. Lodewijks. Mapping NP-hard and NP-complete optimisation problems to Quadratic Unconstrained Binary Optimisation problems, 2019. URL `http://arxiv.org/abs/1911.08043`. arXiv preprint arXiv:1911.08043.

[36] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. doi:10.1112/plms/s2-42.1.230.

[37] H. Vollmer. *Introduction to Circuit Complexity*. Texts in Theoretical Computer Science An EATCS Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 978-3-642-08398-3. doi:10.1007/978-3-662-03927-4.

[38] D. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989. doi:10.1098/rspa.1989.0099.

[39] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, 1995. doi:10.1103/PhysRevA.52.3457.

[40] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 10th edition, 2011. ISBN 1107002176.

[41] R. Raussendorf and H. J. Briegel. A One-Way Quantum Computer. *Physical Review Letters*, 86(22):5188–5191, 2001. doi:10.1103/PhysRevLett.86.5188.

[42] R. Raussendorf, D. E. Browne, and H. J. Briegel. Measurement-based quantum computation on cluster states. *Physical Review A*, 68(2):022312, 2003. doi:10.1103/PhysRevA.68.022312.

[43] H. Goto. Universal quantum computation with a nonlinear oscillator network. *Physical Review A*, 93(5):050301, 2016. ISSN 2469-9926. doi:10.1103/PhysRevA.93.050301.

[44] V. Kendon. Quantum computing using continuous-time evolution. *Interface Focus*, 10(6):20190143, 2020. doi:10.1098/rsfs.2019.0143.

[45] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum Computation by Adiabatic Evolution, 2000. URL `http://arxiv.org/abs/quant-ph/0001106`. arXiv preprint arXiv:quant-ph/0001106.

[46] B. Apolloni, C. Carvalho, and D. de Falco. Quantum stochastic optimization. *Stochastic Processes and their Applications*, 33(2):233–244, 1989. doi:10.1016/0304-4149(89)90040-9.

[47] R. L. Somorjai. Novel approach for computing the global minimum of proteins. 1. General concepts, methods, and approximations. *The Journal of Physical Chemistry*, 95(10):4141–4146, 1991. doi:10.1021/j100163a045.

[48] T. Kadowaki and H. Nishimori. Quantum annealing in the transverse Ising model. *Physical Review E*, 58(5):5355–5363, 1998. doi:10.1103/PhysRevE.58.5355.

[49] J. Brooke, D. Bitko, T. F. Rosenbaum, and G. Aeppli. Quantum annealing of a disordered magnet. *Science*, 284(5415):779–781, 1999. doi:10.1126/science.284.5415.779.

[50] A. Perdomo-Ortiz, S. E. Venegas-Andraca, and A. Aspuru-Guzik. A study of heuristic guesses for adiabatic quantum computation. *Quantum Information Processing*, 10(1):33–52, 2011. doi:10.1007/s11128-010-0168-z.

[51] Q.-H. Duan, S. Zhang, W. Wu, and P.-X. Chen. An Alternative Approach to Construct the Initial Hamiltonian of the Adiabatic Quantum Computation. *Chinese Physics Letters*, 30(1):010302, 2013. doi:10.1088/0256-307X/30/1/010302.

[52] N. Chancellor. Modernizing quantum annealing using local searches. *New Journal of Physics*, 19(2):023024, 2017. doi:10.1088/1367-2630/aa59c4.

[53] M. Ohkuwa, H. Nishimori, and D. A. Lidar. Reverse annealing for the fully connected p-spin model. *Physical Review A*, 98(2):022314, 2018. doi:10.1103/PhysRevA.98.022314.

[54] J. Roland and N. J. Cerf. Quantum search by local adiabatic evolution. *Physical Review A*, 65(4):042308, 2002. doi:10.1103/PhysRevA.65.042308.

[55] M. Born and V. Fock. Beweis des Adiabatensatzes. *Zeitschrift für Physik*, 51 (3-4):165–180, 1928. doi:10.1007/BF01343193.

[56] T. Albash and D. A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002, 2018. doi:10.1103/RevModPhys.90.015002.

[57] M. H. S. Amin. Consistency of the Adiabatic Theorem. *Physical Review Letters*, 102(22):220401, 2009. doi:10.1103/PhysRevLett.102.220401.

[58] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev. Adiabatic Quantum Computation Is Equivalent to Standard Quantum Computation. *SIAM Review*, 50(4):755–787, 2008. doi:10.1137/080734479.

[59] J. D. Biamonte and P. J. Love. Realizable Hamiltonians for universal adiabatic quantum computers. *Physical Review A*, 78(1):012352, 2008. doi:10.1103/PhysRevA.78.012352.

[60] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4):2493–2496, 1995. doi:10.1103/PhysRevA.52.R2493.

[61] D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing - STOC '97*, pages 176–188, New York, New York, USA, 1997. ACM Press. doi:10.1145/258533.258579.

[62] E. Knill, R. Laflamme, and W. H. Zurek. Resilient Quantum Computation. *Science*, 279(5349):342–345, 1998. doi:10.1126/science.279.5349.342.

[63] J. Roffe. Quantum error correction: an introductory guide. *Contemporary Physics*, 60(3):226–245, 2019. doi:10.1080/00107514.2019.1667078.

[64] K. C. Young, M. Sarovar, and R. Blume-Kohout. Error Suppression and Error Correction in Adiabatic Quantum Computation: Techniques and Challenges. *Physical Review X*, 3(4):041013, 2013. doi:10.1103/PhysRevX.3.041013.

[65] A. Callison, M. Festenstein, J. Chen, L. Nita, V. Kendon, and N. Chancellor. Energetic Perspective on Rapid Quenches in Quantum Annealing. *PRX Quantum*, 2(1):010338, 2021. doi:10.1103/PRXQuantum.2.010338.

[66] E. Crosson, E. Farhi, C. Y.-Y. Lin, H.-H. Lin, and P. Shor. Different Strategies for Optimization Using the Quantum Adiabatic Algorithm, 2014. URL `http://arxiv.org/abs/1401.7320`. arXiv preprint arXiv:1401.7320.

[67] D. Wecker, M. B. Hastings, and M. Troyer. Training a quantum optimizer. *Physical Review A*, 94(2):022309, 2016. doi:10.1103/PhysRevA.94.022309.

[68] L. Hormozi, E. W. Brown, G. Carleo, and M. Troyer. Nonstoquastic Hamiltonians and quantum annealing of an Ising spin glass. *Physical Review B*, 95 (18):1–9, 2017. doi:10.1103/PhysRevB.95.184416.

[69] L. Fry-Bouriaux, D. T. O'Connor, N. Feinstein, and P. A. Warburton. Locally suppressed transverse-field protocol for diabatic quantum annealing. *Physical Review A*, 104(5):052616, 2021. doi:10.1103/PhysRevA.104.052616.

[70] N. Feinstein, L. Fry-Bouriaux, S. Bose, and P. A. Warburton. Effects of XX catalysts on quantum annealing spectra with perturbative crossings. *Physical Review A*, 110(4):042609, 2024. doi:10.1103/PhysRevA.110.042609.

[71] D-Wave Quantum. D-Wave Advantage2 data sheet, 2025. URL `https://www.dwavequantum.com/`. [Online; accessed 25-May-2025].

[72] D-Wave Quantum. QPU-Specific Physical Properties: Advantage_system6.3, 2023. URL `https://docs.dwavesys.com/`. D-Wave User Manual 09-1272A-C [Online; accessed 25-March-2024].

[73] A. D. King, S. Suzuki, J. Raymond, A. Zucca, T. Lanting, F. Altomare, A. J. Berkley, S. Ejtemaee, E. Hoskinson, S. Huang, E. Ladizinsky, A. J. R. MacDonald, G. Marsden, T. Oh, G. Poulin-Lamarre, M. Reis, C. Rich, Y. Sato, J. D. Whittaker, J. Yao, R. Harris, D. A. Lidar, H. Nishimori, and M. H. Amin. Coherent quantum annealing in a programmable 2,000 qubit Ising chain. *Nature Physics*, 18(11):1324–1328, 2022. doi:10.1038/s41567-022-01741-6.

[74] V. Choi. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quantum Information Processing*, 7(5):193–209, 2008. doi:10.1007/s11128-008-0082-9.

[75] A. D. King and C. C. McGeoch. Algorithm engineering for a quantum annealing platform, 2014. URL `http://arxiv.org/abs/1410.2628`. arXiv preprint arXiv:1410.2628.

[76] E. Farhi, J. Goldstone, and S. Gutmann. A Quantum Approximate Optimization Algorithm, 2014. URL `http://arxiv.org/abs/1411.4028`. arXiv preprint arXiv:1411.4028.

[77] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021. doi:10.1038/s42254-021-00348-9.

[78] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler, S.-T. Wang, M. Greiner, V. Vuletić, and M. D. Lukin. Quantum optimization of maximum independent set using Rydberg atom arrays. *Science*, 376 (6598):1209–1215, 2022. doi:10.1126/science.abo6587.

[79] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):4812, 2018. doi:10.1038/s41467-018-07090-4.

[80] M. Larocca, S. Thanasilp, S. Wang, K. Sharma, J. Biamonte, P. J. Coles, L. Cincio, J. R. McClean, Z. Holmes, and M. Cerezo. A Review of Barren Plateaus in Variational Quantum Computing, 2024. URL `http://arxiv.org/abs/2405.00781`. arXiv preprint arXiv:2405.00781.

[81] M. Larocca, P. Czarnik, K. Sharma, G. Muraleedharan, P. J. Coles, and M. Cerezo. Diagnosing barren plateaus with tools from quantum optimal control. *Quantum*, 6, 2022. doi:10.22331/Q-2022-09-29-824.

[82] B. Zhang, A. Sone, and Q. Zhuang. Quantum computational phase transition in combinatorial problems. *npj Quantum Information*, 8(1):87, 2022. doi:10.1038/s41534-022-00596-2.

[83] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer. A review on Quantum Approximate Optimization Algorithm and its variants. *Physics Reports*, 1068:1–66, 2024. doi:10.1016/j.physrep.2024.03.002.

[84] E. Farhi, J. Goldstone, S. Gutmann, and H. Neven. Quantum Algorithms for Fixed Qubit Architectures, 2017. URL `http://arxiv.org/abs/1703.06199`. arXiv preprint arXiv:1703.06199.

[85] R. Herrman, P. C. Lotshaw, J. Ostrowski, T. S. Humble, and G. Siopsis. Multi-angle quantum approximate optimization algorithm. *Scientific Reports*, 12(1):6781, 2022. doi:10.1038/s41598-022-10555-8.

[86] I. Gaidai and R. Herrman. Performance analysis of multi-angle QAOA for $p > 1$. *Scientific Reports*, 14(1):18911, 2024. doi:10.1038/s41598-024-69643-6.

[87] V. Srivastava, R. P, S. Singhal, and D. Bhowmik. Improved Performance of Multi-Angle Quantum Approximate Optimization Algorithm (ma-

QAOA) Compared to QAOA On Simulation and Experimental Hardware Platforms. In *2025 17th International Conference on COMmunication Systems and NETworks (COMSNETS)*, pages 1130–1135. IEEE, 2025. doi:10.1109/COMSNETS63942.2025.10885567.

[88] S. Kazi, M. Larocca, M. Farinati, P. J. Coles, M. Cerezo, and R. Zeier. Analyzing the quantum approximate optimization algorithm: ansätze, symmetries, and Lie algebras, 2024. URL `http://arxiv.org/abs/2410.05187`. arXiv preprint arXiv:2410.05187.

[89] K. Shi, R. Herrman, R. Shaydulin, S. Chakrabarti, M. Pistoia, and J. Larson. Multiangle QAOA Does Not Always Need All Its Angles. In *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, pages 414–419. IEEE, 2022. doi:10.1109/SEC54971.2022.00062.

[90] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983. doi:10.1126/science.220.4598.671.

[91] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009. ISBN 0262033844.

[92] V. Granville, M. Krivanek, and J.-P. Rasson. Simulated annealing: a proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, 1994. doi:10.1109/34.295910.

[93] A. Nolte and R. Schrader. A Note on the Finite Time Behaviour of Simulated Annealing. In *Operations Research Proceedings 1996*, pages 175–180, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. doi:10.1007/978-3-642-60744-8_32.

[94] L. Meredith and D. Maki. Product cannibalization and the role of prices. *Applied Economics*, 33(14):1785–1793, 2001. doi:10.1080/00036840010015769.

[95] C. Aguilar-Palacios, S. Munoz-Romero, and J. L. Rojo-Alvarez. Causal Quantification of Cannibalization During Promotional Sales in Grocery Retail. *IEEE Access*, 9:34078–34089, 2021. doi:10.1109/ACCESS.2021.3062222.

[96] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. doi:10.1007/978-0-387-40065-5.

[97] N. Van Thoai. Solution Methods for General Quadratic Programming Problem with Continuous and Binary Variables: Overview. In *Studies in Computational Intelligence*, volume 479, pages 3–17, Heidelberg, 2013. Springer International Publishing. doi:10.1007/978-3-319-00293-4_1.

[98] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush. Quantum Simulation of Electronic Structure with Linear Depth and Connectivity. *Physical Review Letters*, 120(11):110501, 2018. ISSN 0031-9007. doi:10.1103/PhysRevLett.120.110501.

[99] B. O'Gorman, W. J. Huggins, E. G. Rieffel, and K. B. Whaley. Generalized swap networks for near-term quantum computing, 2019. URL `http://arxiv.org/abs/1905.05118`. arXiv preprint arXiv:1905.05118.

[100] T. Hagge. Optimal fermionic swap networks for Hubbard models, 2020. URL `http://arxiv.org/abs/2001.08324`. arXiv preprint arXiv:2001.08324.

[101] A. Hashim, R. Rines, V. Omole, R. K. Naik, J. M. Kreikebaum, D. I. Santiago, F. T. Chong, I. Siddiqi, and P. Gokhale. Optimized SWAP networks with equivalent circuit averaging for QAOA. *Physical Review Research*, 4(3): 033028, 2022. doi:10.1103/PhysRevResearch.4.033028.

[102] I. Hen and F. M. Spedalieri. Quantum Annealing for Constrained Optimization. *Physical Review Applied*, 5(3):034007, 2016. doi:10.1103/PhysRevApplied.5.034007.

[103] I. Hen and M. S. Sarandy. Driver Hamiltonians for constrained optimization in quantum annealing. *Physical Review A*, 93(6):062312, 2016. doi:10.1103/PhysRevA.93.062312.

[104] W. Lechner, P. Hauke, and P. Zoller. A quantum annealing architecture with all-to-all connectivity from local interactions. *Science Advances*, 1(9):1–5, 2015. doi:10.1126/sciadv.1500838.

[105] M. Drieb-Schön, K. Ender, Y. Javanmard, and W. Lechner. Parity Quantum Optimization: Encoding Constraints. *Quantum*, 7:951, 2023. doi:10.22331/q-2023-03-17-951.

[106] M. Streif, M. Leib, F. Wudarski, E. Rieffel, and Z. Wang. Quantum algorithms with local particle-number conservation: Noise effects and error correction. *Physical Review A*, 103(4):042412, 2021. doi:10.1103/PhysRevA.103.042412.

[107] S. Hadfield, Z. Wang, B. O'Gorman, E. Rieffel, D. Venturelli, and R. Biswas. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms*, 12(2):34, 2019. doi:10.3390/a12020034.

[108] Z. Wang, N. C. Rubin, J. M. Dominy, and E. G. Rieffel. XY mixers: Analytical and numerical results for the quantum alternating operator ansatz. *Physical Review A*, 101(1):012320, 2020. doi:10.1103/PhysRevA.101.012320.

[109] J. Cook, S. Eidenbenz, and A. Bartschi. The Quantum Alternating Operator Ansatz on Maximum k-Vertex Cover. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 83–92. IEEE, 2020. doi:10.1109/QCE49297.2020.00021.

[110] T. Vyskocil and H. Djidjev. Embedding equality constraints of optimization problems into a quantum annealer. *Algorithms*, 12(4):1–24, 2019. doi:10.3390/A12040077.

[111] T. Vyskočil, S. Pakin, and H. N. Djidjev. Embedding Inequality Constraints for Quantum Annealing Optimization. In *Quantum Technology and Optimization Problems*, volume 11413 of *Lecture Notes in Computer Science*, pages 11–22. Springer International Publishing, 2019. doi:10.1007/978-3-030-14082-3_2.

[112] H. Djidjev. Automaton-based methodology for implementing optimization constraints for quantum annealing. In *Proceedings of the 17th ACM International Conference on Computing Frontiers*, pages 118–125. ACM, 2020. ISBN 9781450379564. doi:10.1145/3387902.3392619.

[113] J. A. Montañez-Barrera, D. Willsch, A. Maldonado-Romo, and K. Michielsen. Unbalanced penalization: a new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms. *Quantum Science and Technology*, 9(2):025022, 2024. doi:10.1088/2058-9565/ad35e4.

[114] D. De Santis, S. Tirone, S. Marmi, and V. Giovannetti. Optimized QUBO formulation methods for quantum computing, 2024. URL `http://arxiv.org/abs/2406.07681`. arXiv preprint arXiv:2406.07681.

[115] D. Herman, R. Shaydulin, Y. Sun, S. Chakrabarti, S. Hu, P. Minssen, A. Rattew, R. Yalovetzky, and M. Pistoia. Constrained optimization via quantum Zeno dynamics. *Communications Physics*, 6(1):219, 2023. doi:10.1038/s42005-023-01331-9.

[116] L. T. Brady and S. Hadfield. Iterative Quantum Algorithms for Maximum Independent Set: A Tale of Low-Depth Quantum Algorithms, 2023. URL `http://arxiv.org/abs/2309.13110`. arXiv preprint arXiv:2309.13110.

[117] G. van Rossum and F. L. Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica, 1995.

[118] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus,

S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi:10.1038/s41586-020-2649-2.

[119] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G.-L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, and Y. Vázquez-Baeza. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020. doi:10.1038/s41592-019-0686-2.

[120] J. D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi:10.1109/MCSE.2007.55.

[121] M. Zaman, K. Tanahashi, and S. Tanaka. PyQUBO: Python Library for Mapping Combinatorial Optimization Problems to QUBO Form. *IEEE Transactions on Computers*, 71(4):838–850, 2022. doi:10.1109/TC.2021.3063618.

[122] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL `https://www.gurobi.com`.

[123] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023. URL `doi.org/10.5281/zenodo.2573505`.

[124] M. J. D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, Dordrecht, 1994. doi:10.1007/978-94-015-8330-5_4.

[125] J. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37 (3):332–341, 1992. doi:10.1109/9.119632.

[126] D-Wave Quantum. D-Wave Ocean SDK, 2025. URL `https://docs.dwavequantum.com/`. [Online; accessed 25-May-2025].

[127] J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2: 79, 2018. doi:10.22331/q-2018-08-06-79.

[128] A. Rocchetto, S. C. Benjamin, and Y. Li. Stabilizers as a design tool for new forms of the Lechner-Hauke-Zoller annealer. *Science Advances*, 2(10), 2016. doi:10.1126/sciadv.1601246.

[129] N. Chancellor. Domain wall encoding of discrete variables for quantum annealing and QAOA. *Quantum Science and Technology*, 4(4):045004, 2019. doi:10.1088/2058-9565/ab33c2.

[130] J. Chen, T. Stollenwerk, and N. Chancellor. Performance of Domain-Wall Encoding for Quantum Annealing. *IEEE Transactions on Quantum Engineering*, 2:1–14, 2021. doi:10.1109/tqe.2021.3094280.

[131] J. Berwald, N. Chancellor, and R. Dridi. Understanding domain-wall encoding theoretically and experimentally. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 381(2241), 2023. doi:10.1098/rsta.2021.0410.

[132] M. Ohzeki. Breaking limitation of quantum annealer in solving optimization problems under constraints. *Scientific Reports*, 10(1):3126, 2020. doi:10.1038/s41598-020-60022-5.

[133] D. Willsch, M. Willsch, H. De Raedt, and K. Michielsen. Support vector machines on the D-Wave quantum annealer. *Computer Physics Communications*, 248:107006, 2020. doi:10.1016/j.cpc.2019.107006.

[134] R. Fletcher. *Penalty Functions*, pages 87–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983. doi:10.1007/978-3-642-68874-4_5.

[135] P. D. de la Grand'rive and J.-F. Hullo. Knapsack Problem variants of QAOA for battery revenue optimisation, 2019. URL `http://arxiv.org/abs/1908.02210`. arXiv preprint arXiv:1908.02210.

[136] S. Yu and T. Nabil. Applying the Hubbard-Stratonovich Transformation to Solve Scheduling Problems Under Inequality Constraints With Quantum Annealing. *Frontiers in Physics*, 9:1–10, 2021. doi:10.3389/fphy.2021.730685.

[137] R. L. Stratonovich. On a method of calculating quantum distribution functions. In *Soviet Physics Doklady*, volume 2, page 416, 1957.

[138] J. Hubbard. Calculation of Partition Functions. *Physical Review Letters*, 3 (2):77–78, 1959. doi:10.1103/PhysRevLett.3.77.

[139] M. Kuramata, R. Katsuki, and K. Nakata. Larger Sparse Quadratic Assignment Problem Optimization Using Quantum Annealing and a Bit-Flip Heuristic Algorithm. In *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 556–565. IEEE, 2021. doi:10.1109/ICIEA52957.2021.9436749.

[140] T. Takabayashi, T. Goto, and M. Ohzeki. Subgradient Method Using Quantum Annealing for Inequality-Constrained Binary Optimization Problems. *Journal of the Physical Society of Japan*, 94(5):054003, 2025. doi:10.7566/JPSJ.94.054003.

[141] R. Beier and B. Vöcking. Random knapsack in expected polynomial time. *Journal of Computer and System Sciences*, 69(3):306–329, 2004. doi:10.1016/j.jcss.2004.04.004.

[142] J. Jooken, P. Leyman, and P. De Causmaecker. A new class of hard problem instances for the 0–1 knapsack problem. *European Journal of Operational Research*, 301(3):841–854, 2022. doi:10.1016/j.ejor.2021.12.009.

[143] A. Callison and N. Chancellor. Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond. *Physical Review A*, 106 (1):010101, 2022. doi:10.1103/PhysRevA.106.010101.

[144] L. Landau. Zur theorie der energieubertragung. II. *Physikalische Zeitschrift der Sowjetunion*, 2:46, 1932.

[145] C. Zener. Non-adiabatic crossing of energy levels. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 137(833):696–702, 1932. doi:10.1098/rspa.1932.0165.

[146] A. D. King and W. Bernoudy. Performance benefits of increased qubit connectivity in quantum annealing 3-dimensional spin glasses, 2020. URL `http://arxiv.org/abs/2009.12479`. arXiv preprint arXiv:2009.12479.

[147] M. K. Haghighi and N. Dimopoulos. Minimum-Length Chain Embedding for the Phase Unwrapping Problem on D-Wave's Pegasus Graph. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 2, pages 318–319. IEEE, 2023. doi:10.1109/QCE57702.2023.10261.

[148] E. L. Lehmann. The Fisher, Neyman-Pearson Theories of Testing Hypotheses: One Theory or Two? *Journal of the American Statistical Association*, 88(424):1242–1249, 1993. doi:10.1080/01621459.1993.10476404.

[149] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner. Improving variational quantum optimization using CVaR. *Quantum*, 4:1–16, 2020. doi:10.22331/q-2020-04-20-256.

[150] L. Li, M. Fan, M. Coram, P. Riley, and S. Leichenauer. Quantum optimization with a novel Gibbs objective function and ansatz architecture search. *Physical Review Research*, 2(2):1–10, 2020. doi:10.1103/PhysRevResearch.2.023074.

[151] S. Boulebnane and A. Montanaro. Solving Boolean Satisfiability Problems With The Quantum Approximate Optimization Algorithm. *PRX Quantum*, 5(3):030348, 2024. doi:10.1103/PRXQuantum.5.030348.

[152] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. Di Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri, P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, and N. Killoran. PennyLane: Automatic differentiation of hybrid quantum-classical computations, 2018. URL `http://arxiv.org/abs/1811.04968`. arXiv preprint arXiv:1811.04968.

[153] L. Lavagna, A. Ceschini, A. Rosato, and M. Panella. A Layerwise-Multi-Angle Approach to Fine-Tuning the Quantum Approximate Optimization

Algorithm. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2024. doi:10.1109/IJCNN60899.2024.10650075.

[154] J. A. Montanez-Barrera and K. Michielsen. Towards a universal QAOA protocol: Evidence of a scaling advantage in solving some combinatorial optimization problems, 2024. URL `http://arxiv.org/abs/2405.09169`. arXiv preprint arXiv:2405.09169.

[155] I. G. Hughes and T. P. A. Hase. *Measurements and their uncertainties: a practical guide to modern error analysis*. Oxford University Press, Oxford, United Kingdom, 1 edition, 2010. ISBN 9780199566334.

[156] R. Herrman. Investigating parameters in the multi-angle approximate optimization algorithm. In *APS March Meeting 2022*, volume 67, page G00.075, 2022.

Colophon

This thesis is based on a template developed by Matthew Townson and Andrew Reeves. It was typeset with LaTeX 2_ε. It was created using the *memoir* package, maintained by Lars Madsen, with the *madsen* chapter style. The font used is Latin Modern, derived from fonts designed by Donald E. Kunith.