

B.Comp. Dissertation

**Data-driven and Physics-inspired Machine Learning:
Benchmarking quantum-inspired quadratic
unconstrained binary optimisation solvers**

By

Guo Yulong

Department of Computer Science

School of Computing

National University of Singapore

2023/2024

B.Comp. Dissertation

**Data-driven and Physics-inspired Machine Learning:
Benchmarking quantum-inspired quadratic
unconstrained binary optimisation solvers**

By

Guo Yulong

Department of Computer Science

School of Computing

National University of Singapore

2023/2024

Project No: H0811550

Advisor: Assoc Prof Stéphane Bressan

Deliverables:

Report: 1 Volume

Abstract

The quadratic unconstrained binary optimisation (QUBO) problem represents a cornerstone in combinatorial optimisation that encapsulates various scientific and industrial optimisation problems. However, the QUBO problem is NP-hard and is traditionally extremely difficult to solve efficiently due to the exponentially increasing search space. Our work aims to measure the performance of different quantum-inspired QUBO solvers that utilise quantum effects to search for optimal solutions. We benchmark the D-Wave Quantum Annealer, Neural Network Quantum States (NNQS), and Quantum Approximate Optimization Algorithm (QAOA) solvers with classical commercial QUBO solvers. We use a dataset with various combinatorial problems—not-all-equal 3-satisfiability, max-cut, and the Sherrington-Kirkpatrick model (SK model), and the solvers will be evaluated based on the average normalised energy and success probability. Our results show that the D-wave and NNQS solvers outperform the QAOA solver but cannot match the performance of commercial QUBO solvers. However, the D-wave solver shows promising performance compared to the GUROBI solver for some problem sizes when we match the solver runtimes. We further investigate the performance of the NNQS solver when using different architectures and training algorithms. Our results show that the NNQS with a Restricted Boltzmann Machine and a continuous training algorithm perform best across all datasets.

Subject Descriptors:

10002950.10003714.10003716.10011136 Discrete optimization

10010405.10010432.10010441 Applied computing Physics

10010147.10010257.10010293.10010294 Neural networks

Keywords:

Quadratic unconstrained binary optimisation, Ising model, Quantum annealing, Neural network quantum states, Quantum Approximate Optimization Algorithm

Implementation Software and Hardware:

Linux, GPUs, Tesla A100, Python, Tensorflow, Dimod, Qiskit

Acknowledgement

I would like to thank my friends and family for their invaluable support during the project. I would also like to thank my project advisor, Associate Professor Stéphane Bressan, research fellow Lu Jianlong, and past and present research group members for their invaluable guidance and support.

List of Figures

3.1	Energy landscape before (left) and after (right) quantum annealing	16
3.2	Structure of Restricted Boltzmann Machine	19
3.3	Structure of Multilayer Perceptron with one hidden layer and one real-valued output node	20
3.4	Circuit diagram of the QAOA algorithm circuit with $p = 1$	21
4.1	A view of the D-Wave pegasus topology—each line represents a qubit and intersections represent available couplers	25
4.2	Annealing functions $A(s)$ and $B(s)$ as a function of the normalized anneal fraction s	26
4.3	Circuit diagram with the Hadamard gates, problem Hamiltonian operator, and mixing Hamiltonian operator and (left to right)	29
5.1	Performance of different solvers for NAE3SAT by problem size	36
5.2	Average performance of different solvers for NAE3SAT	37
5.3	Performance of different solvers for max-cut by problem size	38
5.4	Average performance of different solvers for max-cut	39
5.5	Performance of different solvers for SK model by problem size	40
5.6	Average performance of different solvers for SK model	41
5.7	Average runtime in log scale taken by different solvers for QUBO problems by size	42
5.8	Performance of D-wave solver against GUROBI and Fixstar by problem type and size	44
6.1	Average performance of different NNQS types for NAE3SAT	49
6.2	Average performance of different NNQS types for max-cut	50
6.3	Average performance of different NNQS types for SK model	50
B.1	Fitted $A(s)$ and $B(s)$ equations (red) with discrete values (blue) against the normalised annealing fraction s	B-2
C.1	Performance of different NNQS types for NAE3SAT by problem size	C-2
C.2	Performance of different NNQS types for max-cut by problem size	C-3
C.3	Performance of different NNQS types for SK model by problem size	C-4
D.1	Average runtime taken by different solvers for NAE3SAT by problem size	D-1
D.2	Average runtime taken by different solvers for max-cut by problem size .	D-2
D.3	Average runtime taken by different solvers for SK model by problem size	D-3

List of Tables

5.1	Average runtime (seconds) of the D-wave solver by problem type and size. Dashes indicate that the D-wave solver could not embed problems of that size.	43
5.2	average normalised energy for different solvers	45
5.3	Success probability for different solvers	46
6.1	average normalised energy for different NNQS types	52
6.2	Success probability for different NNQS types	52
B.1	Discrete points of annealing functions $A(s)$ and $B(s)$	B-1
E.1	Average number of embedded qubits for the D-wave solver by problem type and size	E-1
E.2	Average number of quantum gates in the quantum circuit used by the QAOA solver by problem type and size	E-2
E.3	Average depth of the quantum circuit used by the QAOA solver by problem type and size	E-2

Table of Contents

Title	i
Abstract	ii
Acknowledgement	iii
List of Figures	iv
List of Tables	v
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
2 Background and Preliminaries	4
2.1 Quadratic unconstrained binary optimisation	4
2.1.1 Example QUBO problem	5
2.1.2 Practical applications	6
2.2 The Ising Model	7
2.2.1 Ising Hamiltonian	7
2.2.2 Converting QUBO to Ising	8
2.2.3 Converting Ising to QUBO	9
2.3 Solving for the ground state of the Ising model	10
2.4 Wave functions, Observables and Ansatzes	10
2.5 Sampling methods	11
3 Methods for solving QUBO problems	14
3.1 Classical	14
3.2 Quantum Annealing	16
3.3 Neural-Network Quantum States	18
3.4 Hybrid quantum-classical	20
4 Methodology	23
4.1 Solvers	23
4.1.1 D-Wave Quantum Annealing	23
4.1.2 Neural-Network Quantum States	26
4.1.3 Quantum Approximate Optimization Algorithm	28

4.1.4	GUROBI Optimizer	29
4.1.5	Fixstars Amplify QUBO Solver	30
4.2	Benchmark datasets	30
4.3	Performance evaluation	32
5	Benchmarking QUBO solvers	34
5.1	Related benchmarking work	34
5.2	Results and Discussion	35
5.2.1	NAE3SAT	35
5.2.2	Max-cut	37
5.2.3	SK model	39
5.3	Time-Constrained Solver Comparison	41
5.4	Solver logs	43
5.5	Conclusion	43
6	NNQS exploration	47
6.1	Architectures and training algorithms	47
6.2	Results and Discussion	49
6.2.1	NAE3SAT	49
6.2.2	Max-cut	49
6.2.3	SK model	50
6.3	Conclusion	51
7	Conclusion	53
7.1	Contributions	53
7.2	Future work	55
References		56
A	Reformulating the Knapsack problem as a QUBO problem	A-1
B	Curve fitting for NNQS	B-1
C	NNQS exploration performance by sizes	C-1
C.1	NAE3SAT	C-1
C.2	Max-cut	C-1
C.3	SK model	C-1
D	Average runtime of solvers by problem type and sizes	D-1
D.1	NAE3SAT	D-1
D.2	Max-cut	D-2
D.3	SK model	D-2
E	Metadata of solvers by problem type and sizes	E-1
E.1	D-wave	E-1
E.2	QAOA	E-1
F	D-wave Quenching	F-1

Chapter 1

Introduction

1.1 Motivation

The quadratic unconstrained binary optimisation (QUBO) problem can represent various types of combinatorial optimisation problems and has vast applications in both operational research and industry (Kochenberger, Hao, Glover, Lewis, Lu, Wang, & Wang, 2014).

Most problems that involve choosing a set of binary decisions to optimise an objective function can be expressed as a QUBO problem. Examples include the max-cut problem, knapsack problem, or even problems with more complex constraints, such as the travelling salesman problem (Lucas, 2014). However, QUBO problems are NP-hard and are extremely hard to solve efficiently (Barahona, 1982Kochenberger et al., 2014). Traditionally, QUBO-solving methods are specialised for a specific problem domain to leverage each problem domain's unique characteristics, limiting the versatility and portability of QUBO-solving methods (Glover, Kochenberger, Hennig, & Du, 2022).

The QUBO problem shares similarities with the Ising model in Physics, where the spin variables are either -1 or 1 , instead of the binary variables used in QUBO. The two models are equivalent, and the correspondence between the QUBO problem and the Ising

model has opened the doors to solving QUBO problems with quantum computational methods (Glover et al., 2022). The equivalence enhances the problem domains that could be modelled as QUBO problems and allows for QUBO problems to be solved by more general quantum-inspired solving methods (Glover et al., 2022).

With a broad range of classical and quantum-inspired methods available to tackle QUBO problems, it is imperative to consolidate and benchmark existing QUBO-solving solvers to highlight their performance when solving different kinds of QUBO problems across a range of problem sizes.

1.2 Objectives

In the first section of the report, we aim to measure the performance of 3 quantum-inspired QUBO-solving solvers:

1. Quantum Annealing
2. Neural Network Quantum States
3. Quantum Approximate Optimization Algorithm

We will also use classical solvers as a baseline for comparison with quantum-inspired solvers. We will explore two classical solvers:

4. Fixstars Amplify QUBO Solver
5. GUROBI Optimizer

For each solver, we will calculate the success rate (probability of solving optimally) and normalised energy (relative performance of candidate solution) across QUBO problems of different types and sizes. More details of the solvers and performance metrics are available in chapter 3 and chapter 4. The results are detailed in chapter 5.

In the second section of the report, we will further explore how Neural Network Quantum States can be used to simulate the quantum annealing process to solve QUBO problems. We systematically compare two neural network architectures and three different training schedules highlighted in chapter 6 for Neural Network Quantum States. The results are also detailed in chapter 6.

The results of the study will allow us to understand the current landscape of quantum-inspired QUBO solvers and how they compare to state-of-the-art classical commercial solvers. We will also be able to compare the performance of different architectures and training schedules for Neural Network Quantum States.

Chapter 2

Background and Preliminaries

2.1 Quadratic unconstrained binary optimisation

The quadratic unconstrained binary optimisation (QUBO) problem is defined as

$$\arg \min_{x \in \{0,1\}^n} x^\top Q x \quad (2.1)$$

where $Q \in M_{n \times n}(\mathbb{R})$ is an upper triangular square matrix with real coefficients and x is a binary input vector (Kochenberger et al., 2014). The matrix Q characterises the QUBO problem. $x^\top Q x$ is also known as the objective function for the problem. Some characteristics of QUBO problems are summarised below:

- The possible input space grows exponentially with the size of the problem n , making the QUBO problem NP-hard and difficult to solve efficiently.
- In general, the solution of a QUBO problem does not need to be unique as multiple input x 's can produce the same objective value.
- The objective function $x^\top Q x$ can also be expressed as $\sum_{i=1}^n \sum_{j=1}^n Q_{ij} x_i x_j$ which intuitively means that we sum up the coefficients Q_{ij} where both x_i and x_j are 1.

- The density d of a QUBO problem refers to the number of non-zero elements above the main diagonal of Q (quadratic terms) divided by the total number of elements above the main diagonal.
- We can also express a QUBO problem as a maximisation problem by finding

$$\arg \max_{x \in \{0,1\}^n} -x^\top Q x.$$

Most problems that optimise an objective function concerning a set of binary decisions can be reformulated as a QUBO problem (Glover et al., 2022). Thus, the QUBO problem model has applications in a wide range of combinatorial optimisation problems such as Max-Cut (Kochenberger, Hao, Lu, Wang, & Glover, 2013), number partitioning (Alidaee, Glover, Kochenberger, & Rego, 2005), and machine scheduling problems (Alidaee, 2009).

Once an optimisation problem is expressed in a QUBO format, we can utilise general QUBO-solving methods to efficiently obtain solutions to the original problem without specialising in a particular problem domain (Kochenberger et al., 2014). The following sections will describe multiple examples of QUBO problems and reformulations.

2.1.1 Example QUBO problem

Consider the objective function

$$f(x_1, x_2, x_3) = -8x_1 + 6x_2 + 3x_3 - 2x_1x_2 + 4x_2x_3 \quad (2.2)$$

where $x_1, x_2, x_3 \in \{0, 1\}$ and we want to minimise f over all possible (x_1, x_2, x_3) (if we wanted to maximise f instead we can simply multiply all the coefficients of f by -1). Since the variables are binary, $x_i^2 = x_i$ for all i , we can redefine the objective function as

$$f(x) = x^\top Q x, \text{ where } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, Q = \begin{bmatrix} -8 & -2 & 0 \\ 0 & 6 & 4 \\ 0 & 0 & 3 \end{bmatrix} \quad (2.3)$$

The coefficients for Q_{ii} are equal to the coefficient of x_i in the original objective function. The coefficients for Q_{ij} where $i < j$ is the coefficient of $x_i x_j$. The coefficients in the lower triangular portion of Q is 0. This simple QUBO problem can be solved by enumerating all the possible inputs to obtain the optimal solution of $x_1 = 1, x_2 = 0, x_3 = 0$, and $f(1, 0, 0) = -8$.

A more complex example is the knapsack problem, a combinatorial optimisation problem with inequality constraints. Appendix A explains how the knapsack problem can be reformulated as a QUBO problem.

2.1.2 Practical applications

There are numerous practical scenarios where optimisation problems can be reformulated as QUBO problems.

- (Yarkoni, Huck, Schuelldorf, Speitkamp, Tabrizi, Leib, Back, & Neukart, 2021) uses real-world data of the location of DB Schenker shipping hubs in Europe to solve the shipment rerouting problem, which aims to reduce the total distance travelled to fulfil a set of shipments.
- (Lang, Zielinski, & Feld, 2022) uses a QUBO formulation to solve portfolio optimisation problems using real-world stock data sets of the New York Stock Exchange.
- (Tavares, 2008) formulates an image binarisation method as a QUBO problem where the objective is to segment an image into its foreground and background, which has further possible medical applications to improve x-ray imaging.

These practical applications of the QUBO problem motivate efficient methods for solving QUBO problems, which are explained further in chapter 3.

2.2 The Ising Model

The Ising model in Physics, proposed by Ernst Ising in 1925, can be thought of as a model of a magnet (Ising, 1925) and has been widely studied in Physics for its phase transition properties (Cipra, 1987). The Ising model serves as the bridge that allows for QUBO problems to be solved with quantum-based methods (Lucas, 2014). In the classical Ising model, a magnet consists of n molecules that are ‘constrained to lie on the sites of a regular lattice’ (Baxter, 2016). Each molecule i can be seen as a ‘microscopic magnet’ that points along some axis and has a ‘spin’ (s_i) that is either +1 (parallel to the axis) or -1 (anti-parallel to the axis). With n particles, the system can have 2^n states, each corresponding to a configuration of the individual molecule spins.

2.2.1 Ising Hamiltonian

In quantum mechanics, the Hamiltonian of a system \hat{H} is a linear operator that represents the total energy of a system and is a critical component that governs the evolution of the system (Griffiths & Schroeter, 2018). Section 2.4 discusses the Hamiltonian and general quantum mechanics in greater detail. For this study, we can treat the Hamiltonian as a function of the quantum system that maps a quantum state to an energy level. The possible energy levels of the system are precisely the eigenvalues of its Hamiltonian, and the corresponding eigenvectors are the possible states mapped to a specific energy level (Zen, My, Tan, Hébert, Gattobigio, Miniatura, Poletti, & Bressan, 2020).

In the Ising model with n particles, the Hamiltonian has two components — the external field term (characterised by $\mathbf{h} = (h_1, h_2, \dots, h_n) \in \mathbb{R}^n$) and the interaction term between molecules (characterised by a strictly upper triangular matrix $\mathbf{J} \in \mathbf{M}_{n \times n}(\mathbb{R})$) (Lucas, 2014). We can express the Hamiltonian as a function of the n spins from each particle:

$$\hat{H}(s) = \hat{H}(s_1, s_2, \dots, s_n) = - \sum_{1 \leq i < j \leq N} J_{ij} s_i s_j - \sum_{i=1}^N h_i s_i \quad (2.4)$$

We can view \mathbf{h} as the interaction of each particle with an external magnetic field and \mathbf{J} as the coupling between pairs of spins. A positive J_{ij} term means that the energy is low when spins s_i and s_j are aligned, while a negative J_{ij} term means that the energy is low when the spins are anti-aligned. A large magnitude of h_i means the spin s_i tends to be aligned or anti-aligned with an external magnetic field in the ground state. The Ising model was initially proposed to study phase transitions of the system at certain critical temperatures by solving for the system's ground state at different temperatures (Ising, 1925). To find the ground state—the state of spins that minimises the total system energy of the Ising model—we have to solve for $\arg \min \hat{H}(s)$ for $s \in \{0, 1\}^N$. This optimisation problem is equivalent to a corresponding QUBO problem up to a change in the variable domain (from spins to binary variables). The equivalence of the QUBO problem and the Ising model is one of the most significant applications of QUBO (Glover et al., 2022). The following subsections will show how to convert a QUBO problem into an Ising model and vice versa.

2.2.2 Converting QUBO to Ising

Given a QUBO problem with QUBO matrix Q , we can use the conversion, $x_i = \frac{s_i+1}{2}, s_i \in \{-1, 1\}$ to change the spin variables into binary variables. The objective function $f(x)$ of the QUBO problem can be expressed as

$$\begin{aligned} f(x) &= f(x_1, \dots, x_n) \\ &= \sum_{1 \leq i < j \leq n} Q_{ij}(x_i x_j) + \sum_{i=1}^N Q_{ii} x_i \\ &= \sum_{1 \leq i < j \leq n} \frac{1}{4} Q_{ij}(s_i + 1)(s_j + 1) + \sum_{i=1}^N \frac{1}{2} Q_{ii}(s_i + 1) \end{aligned}$$

If we group the constant terms into k and let $a_i = \sum_{1 \leq j \leq N, j \neq i} \frac{1}{4} Q_{\min(i,j) \max(i,j)} + \frac{1}{2} Q_{ii}$, we can express the objective function as:

$$f(x) = \sum_{1 \leq i < j \leq n} \frac{1}{4} Q_{ij} s_i s_j + \sum_{i=1}^N a_i s_i + k$$

Removing the constant k , which is irrelevant for optimisation, we can reformulate the QUBO problem as an Ising model with $h_i = -a_i$ and $J_{ij} = -\frac{1}{4}Q_{ij}$ for $i \neq j$. Finding the ground state for the Ising model is the same problem as finding $\arg \min_{x \in \{0,1\}^n} x^\top Q x$, and the solution to the ground state of the Ising model can be converted into a solution for QUBO problem using $x_i = \frac{s_i+1}{2}$. However, the optimal objective function value may differ due to the constant k .

2.2.3 Converting Ising to QUBO

Given the Hamiltonian of an Ising problem, we use the conversion $s_i = 2x_i - 1, x_i \in \{0,1\}$ to change the spin variables into binary variables:

$$\begin{aligned} \hat{H}(s) &= - \sum_{1 \leq i < j \leq n} J_{ij} s_i s_j - \sum_{i=1}^N h_i s_i \\ &= - \sum_{1 \leq i < j \leq n} J_{ij} (2x_i - 1)(2x_j - 1) - \sum_{i=1}^N h_i (2x_i - 1) \\ &= - \sum_{1 \leq i < j \leq n} J_{ij} (4x_i x_j - 2x_i - 2x_j + 1) - \sum_{i=1}^N (2h_i x_i - h_i) \end{aligned}$$

If we group the constant terms into k and let $a_i = 2h_i + \sum_{1 \leq j \leq N, j \neq i} 2J_{\min(i,j) \max(i,j)}$, we can express the Hamiltonian as:

$$\hat{H}(s) = - \sum_{1 \leq i < j \leq n} 4J_{ij} x_i x_j - \sum_{i=0}^N a_i x_i + k$$

Removing the constant k , which is irrelevant for optimisation, we can reformulate the Ising model Hamiltonian as a QUBO matrix Q such that $Q_{ii} = -a_i$ and $Q_{ij} = -4J_{ij}$ for $i < j$. Finding $\arg \min_{x \in \{0,1\}^n} x^\top Q x$ is now the same problem as finding the ground state for the original Ising model, and the solution to the ground state of the Ising model can

be mapped to a solution for QUBO problem using $s_i = 2x_i - 1$. However, the optimal objective function value may differ due to the constant k .

2.3 Solving for the ground state of the Ising model

In one and two-dimensional Ising models, each particle interacts with a small number of neighbours, and the ground state can be solved by calculating the partition function (Onsager, 1944) or by using the transfer matrix method (Kramers & Wannier, 1941). However, the Ising models of interest are of higher dimensions and exact methods for finding their ground states are computationally infeasible for large systems since the computational resources required scale exponentially (Barahona, 1982). However, there are ways to approximate the ground state, such as with the Metropolis-Hastings algorithm (Hastings, 1970), which can be used to increase the probability of finding the system's ground state. More details for solving Ising models with higher dimensions can be found in chapter 3.

2.4 Wave functions, Observables and Ansatzes

Quantum physics is built around wave functions and operators (Griffiths & Schroeter, 2018). Wave functions represent the state of the system and live in a Hilbert space, which is the set of all square-integrable functions:

$$f(x) \text{ such that } \int_{-\infty}^{\infty} |f(x)|^2 dx < \infty$$

which is generally an infinite dimensional complex vector space and allows for the wave function, denoted as Ψ , to be normalised so that $\int_{-\infty}^{\infty} |\Psi|^2 dx = 1$. Under the statistical interpretation of quantum mechanics, $|\Psi(x)|^2$ would also represent the probability of a system being in state x once it is measured. The probabilistic nature of the wave function also implies that a wave function can exist as a superposition of states. For this project, we

will adopt the statistical interpretation and view the wave function as simply a probability distribution over all possible state configurations.

Quantum operators represent the observables—measurable system quantities such as position, momentum, or energy. Quantum operators are hermitian linear operators that act on a wave function to map states to real values. The operator’s eigenvalues have to be real as they correspond to one of the possible measured values of the represented quantity. When an operator acts on a wavefunction that exists as a superposition of multiple states, it returns an expected value of the observable instead. The Hamiltonian is the energy operator, and the eigenvalues of the Hamiltonian are all the possible energies that the system could have. We will only be concerned with the Hamiltonian operator in this study.

For an Ising model, the wavefunction encodes the probabilities of each spin configuration, and the Hamiltonian maps each wavefunction to an energy level. If a wave function only has one possible configuration, the Hamiltonian will return the eigenvalue that matches the energy level of the state. If the wave function is in a superposition of states, the Hamiltonian will return a linear combination of the eigenvalues instead.

An Ansatz is an educated guess for the solution to a problem and often involves reducing the complexity or size of the problem (Blekos, Brand, Ceschini, Chou, Li, Pandya, & Summer, 2024). When solving for the ground state of a wave function with variational methods, an Ansatz represents a subspace of the infinite-dimensional Hilbert space in which the wave function lives. It is important to ensure that the wave function is general enough to closely approximate the space of all solutions yet is specific enough to be computed.

2.5 Sampling methods

Gibbs and Metropolis-Hastings (MH) sampling are Markov Chain Monte Carlo (MCMC) sampling methods. When direct sampling is difficult, they are used to sample from a

complex probability distribution $\mathbb{P}(X)$. We will first introduce MH sampling and then Gibbs sampling as a special case of MH sampling.

Metropolis-Hastings sampling

Metropolis-Hastings sampling (Hastings, 1970) proceeds as follows:

1. Initialise the system with a random sample \mathbf{x}
2. Draw new candidate x^* from $q(x^*|x)$.
3. Accept the new candidate by replacing $\mathbf{x} \leftarrow x^*$ with probability $\alpha = \min\left(1, \frac{\mathbb{P}(x^*)/q(x^*|x)}{\mathbb{P}(x)/q(x|x)}\right)$.
4. Repeat steps 2 and 3 for a certain number of iterations or until \mathbf{x} fulfils some convergence criteria.

Gibbs sampling

Gibbs sampling (Geman & Geman, 1984) is used when directly sampling from the joint distribution $\mathbb{P}(X)$ is difficult, but it is easier to sample from the separate conditional distributions. It proceeds as follows:

1. Initialise the visible layer of the RBM with a random sample \mathbf{s} .
2. Update the hidden layer by sampling from their respective conditional probability distribution given the current visible layer configuration.
3. Update the visible layer by sampling from their respective conditional probability distribution given the current hidden layer configuration.
4. Repeat steps 2 and 3 for a certain number of iterations or until \mathbf{s} fulfils some convergence criteria.

The Gibbs sampling method is, in fact, a special case of MH sampling where we have $\alpha = 1$ and thus always accept the new candidate sample. The key benefit of Gibbs sampling on an RBM is that the visible units are conditionally independent given the hidden layer, and the hidden units are similarly conditionally independent given the visible layer. Thus, Gibbs sampling can be easily parallelised and accelerated using GPUs.

Chapter 3

Methods for solving QUBO problems

This chapter reviews the different approaches used to solve QUBO problems, along with their advantages and limitations. There are a variety of possible methods for solving QUBO problems and Ising models, which can be broadly categorised as Classical, Quantum Annealing, Neural Network Quantum States, and Hybrid Quantum-Classical Algorithms.

3.1 Classical

Classical approaches search for solutions without exploiting quantum properties such as the superposition of states. A typical classical approach to solving large QUBO problems is by exact diagonalisation of the corresponding Ising Hamiltonian (Zen, 2021). Exact diagonalisation solves for all the eigenvalues and eigenvectors by diagonalising the corresponding Ising Hamiltonian matrix. Exact diagonalisation, also known as eigendecomposition, is possible for quantum Hamiltonians as they are represented by Hermitian matrices (Horn & Johnson, 1990). However, the runtime of exact diagonalisation scales exponentially with input problem size and becomes computationally infeasible once the matrix grows large (Zen, 2021). Since we are only interested in finding the smallest eigenvalue and the corre-

sponding eigenvector, we can use iterative methods such as the Lanczos algorithm or the implicitly restarted Arnoldi method to find the smallest eigenvalue (Lanczos, 1950Lehoucq, Sorensen, & Yang, 1998). However, such methods also often run into stability issues. The rapidly increasing search space for QUBO problems has inspired classical methods to find approximate good solutions to QUBO problems instead.

One class of such methods relies on "heuristics" to search for optimal solutions (Dunning, Gupta, & Silberholz, 2018). For example, variants of Tabu search—a local search algorithm that allows for moves that are not improvements and discourages visiting already visited states—are highly competitive heuristic algorithms to find good solutions to QUBO problems(Kochenberger et al., 2013Palubeckis, 2006). Dunning et al.(2018) systematically reviewed and evaluated published heuristics for QUBO problems.

Another classical approximate method for solving QUBO problems is simulated annealing (SA) (Katayama & Narihisa, 2001). Simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983) is a probabilistic method that starts with an initial trial state and a temperature T , which decreases slowly during the search. In each iteration, the algorithm samples neighbouring states of the current state and probabilistically accepts the new sample based on the difference in energy, ΔE , between the current state and the new state. If $\Delta E < 0$, then the new sample is always accepted, and if $\Delta E \geq 0$, the new sample is accepted with a probability $\propto e^{-\frac{\Delta E}{T}}$. The algorithm to sample new states is adapted from the Metropolis-Hastings sampling method (Hastings, 1970), and the chance to explore poorer solutions allows the algorithm to escape local minima.

There are many more classical approaches to solving QUBO problems, as they have been studied extensively. For a detailed survey of classical methods for solving QUBO problems, refer to (Punnen, 2022).

3.2 Quantum Annealing

Quantum annealing, first formulated by Kadowaki and Nishimori (1998), can solve for the ground state configuration of Ising models by utilising the *adiabatic theorem*. The *adiabatic theorem* states that ”a quantum system in its ground state will remain in the ground state, provided that the Hamiltonian governing the dynamics changes sufficiently slowly” (Yarkoni, Raponi, Back, & Schmitt, 2022Born & Fock, 1928). Suppose we can manipulate the final Hamiltonian of a quantum system to one that describes the Ising model of interest. In that case, we can find the ground state of the Ising model simply by measuring the final configuration of the quantum system.

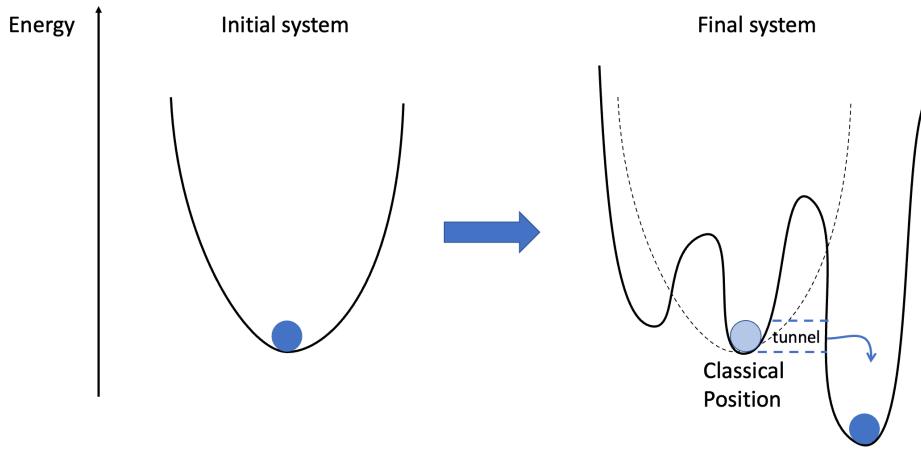


Figure 3.1: Energy landscape before (left) and after (right) quantum annealing

Quantum annealers first prepare a system in the ground state with a simple initial Hamiltonian H_0 (called the mixer or tunnelling Hamiltonian), shown with a simple energy landscape in the left of Figure 3.1. Then, the system Hamiltonian is slowly changed to a more complex form H_c (Lucas, 2014), shown with a complex energy landscape in the right of Figure 3.1. The Hamiltonian at any point $H(s)$ can be written as

$$H(s) = A(s)H_0 + B(s)H_c \quad (3.1)$$

where $s \in [0, 1]$ is the normalised anneal fraction and $A(s)$ and $B(S)$ are decreasing and increasing functions, respectively, and are determined by the specific quantum annealing controls. At the start of the annealing, we should have $A(s) \gg B(s)$, and at the end of the annealing, we should have $B(s) \gg A(s)$. If the transition time is sufficiently large, the adiabatic theorem ensures that the system will remain in the ground state, which can then be measured to yield the desired ground state configuration of H_c (Yarkoni et al., 2022).

As H_0 usually consists of a transverse magnetic field and does not commute with the target Hamiltonian H_c , it allows for quantum tunnelling through energy barriers between classical states (Kadowaki & Nishimori, 1998). Quantum effects allow the wave function to remain at the ground state even when classical search methods may end up stuck in a local minimum. However, the adiabatic theorem requires an arbitrarily long anneal time to guarantee that the system remains in the ground state, which is not feasible for practical purposes. Current implementations of quantum annealing instead rely on a finite time approximation with repeated sampling to increase the success rate (Farhi, Goldstone, Gutmann, Lapan, Lundgren, & Preda, 2001).

Quantum annealing has been extensively studied and applied in multiple fields, such as Scheduling Problems (Rieffel, Venturelli, O’Gorman, do, Prystay, & Smelyanskiy, 2014), Portfolio Optimization (Rosenberg, Haghnegahdar, Goddard, Carr, Wu, & Lopez de Prado, 2016), and Quantum simulations (Harris, Sato, Berkley, Reis, Altomare, Amin, Boothby, Bunyk, Deng, Enderud, Huang, Hoskinson, Johnson, Ladizinsky, Ladizinsky, Lanting, Li, Medina, Molavi, & Yao, 2018). Even though there are significant roadblocks in scaling the currently limited hardware capabilities (Yarkoni et al., 2022), and there is debate on whether Quantum Annealing will eventually run faster than classical search methods (Lucas, 2014), there is hope that these challenges can be tackled soon with the rapid progress made in quantum computing research. The leading commercial provider of

quantum annealing hardware is D-wave, a Canadian quantum computing company (D-Wave Systems, 2024b).

3.3 Neural-Network Quantum States

Carleo and Troyer (Carleo & Troyer, 2017) introduced a neural-network-based method for modelling the wave function of a target quantum system known as *neural-network quantum states* (NNQS). The authors used the approach to find the ground state and time evolution of the Ising and Heisenberg models in Physics. The NNQS is used as an Ansatz to approximate the wave function, which can be viewed as a complex probability distribution, of a system as an artificial neural network (Zen, 2021). The more general method of using an Ansatz as a trial wave function and minimising its energy through Monte Carlo sampling is also known as variational Monte Carlo (VMC).

The theoretical foundations of the ability of NNQS to approximate a wave function rely on the Kolmogorov–Arnold representation theorem (Kolmogorov, 1957), which implies that every multivariate, smooth function can be represented by a neural network with two hidden layers. Since the wave function of a quantum system generally satisfies these requirements, we can expect that a neural network can reasonably approximate the system’s wave function (Carleo & Troyer, 2017).

The original NNQS architecture utilised a Restricted Boltzmann Machine (RBM) as its network (Carleo & Troyer, 2017). The RBM, shown in Figure 3.3, is an energy-based generative model that has two layers of nodes, a visible layer \mathbf{s} with n nodes, a hidden layer \mathbf{h} with m nodes, and a weight matrix \mathbf{W} . The number of hidden nodes is generally a multiple of the number of visible nodes, and the ratio $\alpha = \frac{m}{n}$ is a hyperparameter of the model. Each visible node s_i is connected to every hidden node h_j with a certain weight W_{ij} , but there are no connections within the visible or hidden layer. In other words, the visible and hidden nodes of the RBM form a bipartite graph.

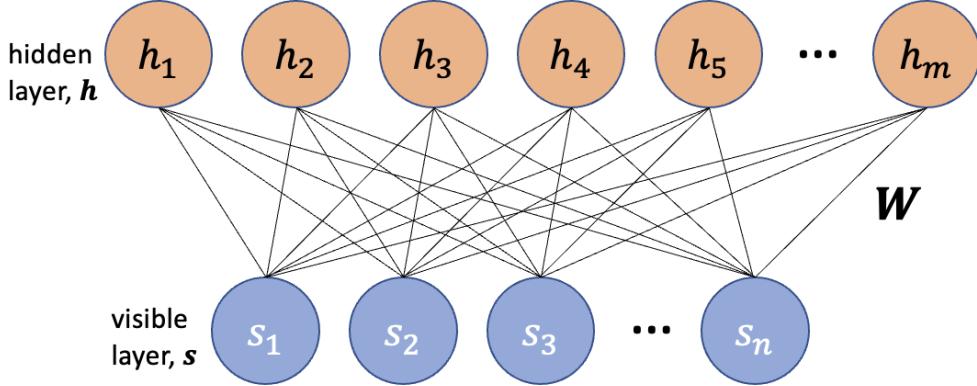


Figure 3.2: Structure of Restricted Boltzmann Machine

When approximating the wave function of an Ising model, each visible node s_i represents the spin of a particle in the Ising model and can only take on the values of 1 and -1 (Carleo & Troyer, 2017). The representation of the wave function by the RBM can be expressed as:

$$\Psi(\mathbf{s}; \boldsymbol{\theta}_{rbm}) = \sum_h e^{\sum_i a_s s_i + \sum_j b_j h_j + \sum_{i,j} W_{is} s_i h_j} \quad (3.2)$$

where $\boldsymbol{\theta}_{rbm} = \{\mathbf{a}, \mathbf{b}, \mathbf{W}\}$ are the biases and weights of the RBM (Carleo & Troyer, 2017). The RBM is usually trained with unsupervised learning. The network weights \mathbf{W} are usually updated by performing Gibbs sampling, calculating the average energy of sampled configurations, and using gradient-based optimisers or the stochastic reconfiguration method to derive the weight updates (Zen, 2021).

Other neural network architectures, such as the Multilayer Perceptron (MLP), can also be used in NNQS. An MLP model is a feedforward artificial neural network that consists of an input layer \mathbf{x} , one or more hidden layers, and an output layer. Each layer is fully connected to the next layer with certain weights, and each node has a non-linear activation function σ such as the sigmoid or ReLU function. Each input node x_i represents the spin of a particle in the Ising model, and the output nodes represent the value of the wave function. If we assume that the wave function is real and positive, then only one output node is needed to represent the real part of the wave function. (Carleo & Troyer,

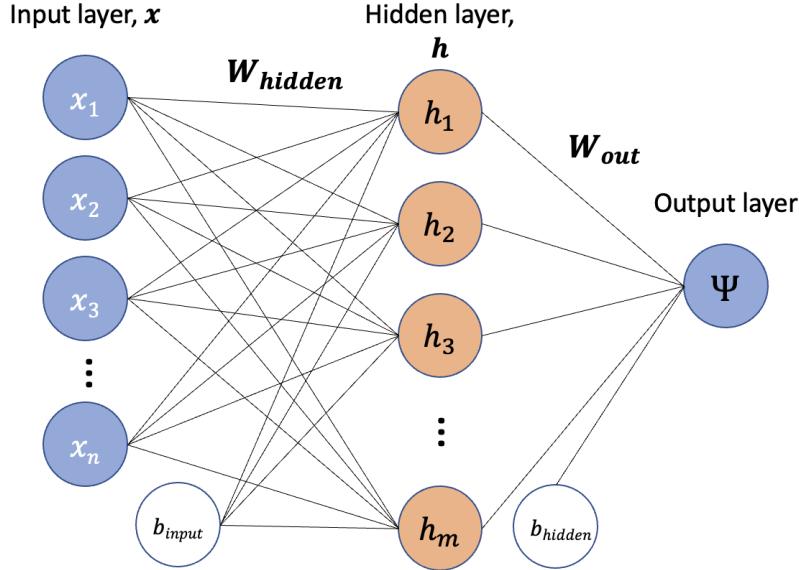


Figure 3.3: Structure of Multilayer Perceptron with one hidden layer and one real-valued output node

2017). With a complex wave function, we could employ output combinations with two nodes representing the real and imaginary components or the magnitude and phase. With one hidden layer, the MLP representation of the wave function can be expressed as:

$$\Psi(\mathbf{x}; \boldsymbol{\theta}_{mlp}) = \sigma_{out} (\mathbf{W}_{out} \sigma_{hidden} (\mathbf{W}_{hidden} \mathbf{x} + \mathbf{b}_{input}) + \mathbf{b}_{hidden}) \quad (3.3)$$

where $\boldsymbol{\theta}_{mlp} = \{\mathbf{W}_{hidden}, \mathbf{b}_{input}, \mathbf{W}_{out}, \mathbf{b}_{hidden}\}$ are the weights and bias of the MLP and $\sigma_{out}, \sigma_{hidden}$ are the non-linear activation functions of the nodes (Carleo & Troyer, 2017).

The MLP is trained in an unsupervised manner, similar to the RBM, except that a more general sampling method, the Metropolis-Hastings algorithm, is used (Zen, 2021).

3.4 Hybrid quantum-classical

The fourth class of QUBO solving methods are hybrid quantum-classical methods. They are designed to use current limited quantum computing resources by integrating them with classic methods to solve combinatorial optimisation problems (Zhou, Wang, Choi,

Pichler, & Lukin, 2020). In the noisy intermediate-scale quantum (NISQ) era that we are currently in, quantum computers are not yet stable enough to reliably run deep and complex quantum circuits.

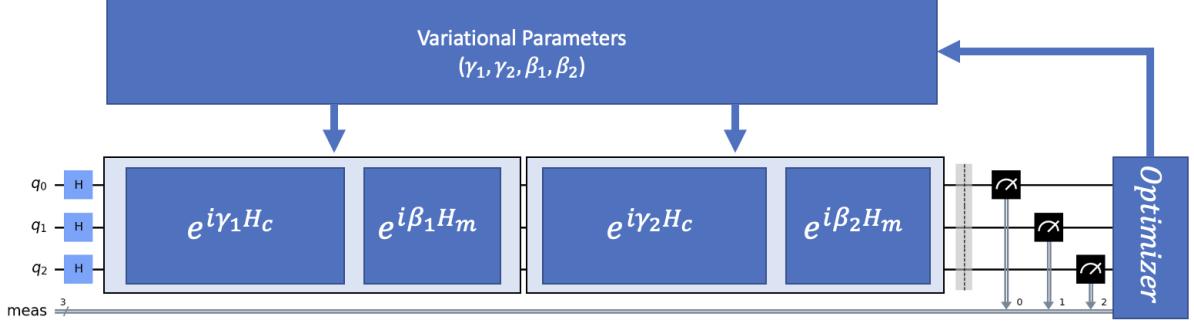


Figure 3.4: Circuit diagram of the QAOA algorithm circuit with $p = 1$

One such algorithm is the Quantum Approximate Optimization Algorithm (QAOA), used to find approximate solutions to general combinatorial optimisation problems (Farhi, Goldstone, & Gutmann, 2014). Like NNQS, QAOA aims to find an approximation of the ground state of an input Hamiltonian using gate-based quantum circuits and $2p$ parameters, which are optimised with a classical computer (Willsch, Willsch, Jin, De Raedt, & Michielsen, 2020). With a problem of size N , the algorithm first prepares a quantum state $|+\rangle^{\otimes N}$ in uniform superposition by applying the Hadamard gate to n qubits (Willsch et al., 2020). The trial wave function is then constructed by

$$\Psi(\boldsymbol{\gamma}, \boldsymbol{\beta}) = U_B(\beta_p)U_C(\gamma_p)\dots U_B(\beta_1)U_C(\gamma_1)|+\rangle^{\otimes N} \quad (3.4)$$

$$U_C(\gamma) = e^{-i\gamma H_c}$$

$$U_B(\beta) = e^{-i\beta H_0}$$

Using the time-independent Schrödinger equation, we can see that U_c and U_B are operators that evolve the state with the Hamiltonian H_c (problem Hamiltonian) and H_0 (an easy-to-implement, mixing Hamiltonian) for time intervals $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. At the same time, the

parameter p determines the number of time evolutions of the final state (Willsch et al., 2020). Each operation in the mixing Hamiltonian is implemented with a single rotation gate R_X while the two-qubit operations in the problem Hamiltonian are implemented with a 2-qubit R_{ZZ} gate (Blekos et al., 2024). The state is then measured repeatedly, and γ and β are chosen to minimise the average energy of the sample measurements. The final solution can be determined by repeatedly sampling from the trial wave function with the optimised parameters.

QAOA can be regarded as a discretised version of Quantum Annealing (Blekos et al., 2024). There is evidence that QAOA is more difficult for classical computers to simulate than Quantum Annealing, which may suggest that it could be a better way to demonstrate quantum supremacy (Farhi & Harrow, 2016). In contrast to quantum annealing, which can only be run on specialised quantum annealing devices, QAOA can be implemented on a general gate-based quantum computer (Kurowski, Pecyna, Słysz, Rozycki, Waligora, & Weglarz, 2023).

Chapter 4

Methodology

This chapter explains the methodology employed in our study, which includes the solvers, datasets, and evaluation metrics.

4.1 Solvers

We will employ five QUBO-solving-methods:

1. D-Wave Quantum Annealing
2. Neural Network Quantum States (NNQS)
3. Quantum Approximate Optimization Algorithm (QAOA)
4. GUROBI Optimizer
5. Fixstars Amplify QUBO Solver

The following sections will provide more information on the solvers used for each method.

4.1.1 D-Wave Quantum Annealing

We will use quantum annealers from D-wave Systems as the solver for Quantum Annealing. D-Wave Systems, a Canadian-based company, currently produces the most popular

commercially available quantum annealing hardware and uses superconducting qubits to represent binary variables in their annealers (Yarkoni et al., 2022). Given a target QUBO problem to solve, the process for using a D-Wave Systems Quantum Processing Unit (QPU) is as follows:

1. **Problem Definition** QUBO problem is first converted to its corresponding Ising model Hamiltonian on the local device.
2. **Minor Embedding** As the Quantum Processing Unit (QPU) used by D-Wave is not fully connected as shown in Figure 4.1, a single variable in the Ising model may need to be represented by multiple qubits called a *chain* which are forced to return the same value with large interaction terms (D-Wave Systems, 2024b). The embedding process will be done with an embedder provided by the D-wave library.
3. **Programming** The parameters of the annealing process are set, which consists of the bias for each qubit (represents magnetic field acting on each qubit) and coupler strength (represents variable interaction).
4. **Initialization** The QPU is initialised in the ground state of an easy-to-implement initial Hamiltonian. The qubits are prepared to be in a superposition of all possible states.
5. **Annealing** The system evolves with a time-varying Hamiltonian.
6. **Readout of solution** The spin values of the qubits are measured and stored as a possible solution.
7. **Resample** As the finite-time quantum annealing process does not guarantee that the system ends up in the ground state, we repeat the sampling to obtain many possible solutions likely to be the ground state of the initial Hamiltonian.

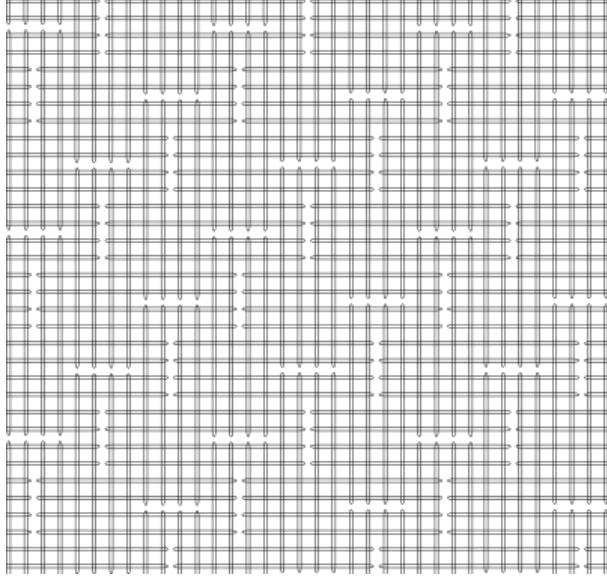


Figure 4.1: A view of the D-Wave pegasus topology where each line represents a qubit and intersections represent available couplers (McGeoch & Farré, 2021)

The experiments will be conducted with the D-Wave Advantage 4.1 QPU with up to 5640 qubits and 40484 couplers (McGeoch & Farré, 2021). The QPU operates at a temperature of approximately 12mK and is built with "a network of tunably coupled rf superconducting quantum-interference device (rf-SQUID) qubits" arranged with the Pegasus graph topology that allows for up to 15 couplers per qubit, shown in Figure 4.1.

The system Hamiltonian as a function of s , the normalised anneal fraction, is:

$$H(s) = -\frac{A(s)}{2} \left(\sum_i \hat{\sigma}_x^{(i)} \right) + \frac{B(s)}{2} \left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j} \hat{\sigma}_z^{(i)} \hat{\sigma}_z^{(j)} \right) \quad (4.1)$$

where functions $A(s)$ and $B(s)$ used in the Advantage 4.1 QPU are shown in Figure 4.2 with $A(s) \gg B(s)$ when $s = 0$ and $B(s) \gg A(s)$ when $s = 1$ and have an energy scale of around $10^{-24}J$. We will use the default annealing time of $20\mu s$ with 1000 repeated samplings (number of reads). The candidate solution will be the best among the 1000 sampled configurations.

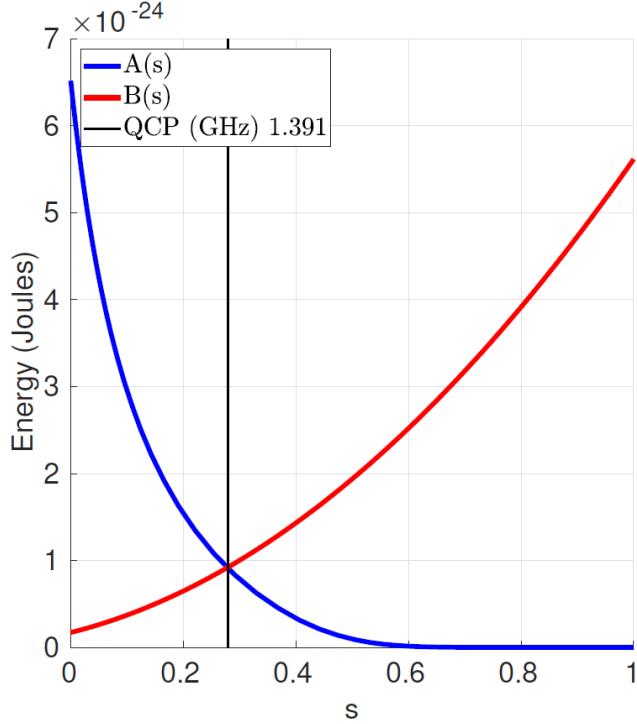


Figure 4.2: Annealing functions $A(s)$ and $B(s)$ as a function of the normalized anneal fraction s (McGeoch & Farré, 2021)

4.1.2 Neural-Network Quantum States

We will adopt the Python library MAPALUS for implementing the Neural-Network Quantum States (Zen, 2021). MAPALUS uses the Tensorflow library and enables parallel execution on general-purpose graphics processing units (GPUs) for quicker sampling. In this section, we will use the Restricted Boltzmann Machine (with $5n$ hidden nodes and the sigmoid activation function) as the underlying architecture for the NNQS.

To solve a given input QUBO problem, the NNQS will simulate a quantum annealing process with a time-dependent Hamiltonian that follows Equation 3.1. Since the equations for $A(s)$ and $B(s)$ are unavailable, we employ a curve-fitting process to obtain analytical functions for use in the NNQS. The fitted functions used are $A(s) = 1.11e^{-7.06s} + -0.00569$ and $B(s) = 0.680s^2 + 0.288s + 0.0305$, details on the curve fitting process are available in Appendix B.

Algorithm 1 NNQS Progressive Training

Require: Problem Hamiltonian \hat{H}_c

Ensure: Trained NNQS

Initialize NNQS with random weights;

for $s \in [0.1, 1.0]$ step 0.1 **do**

 Set $H(s) \leftarrow A(s)\hat{H}_0 + B(s)\hat{H}_c$;

 Train NNQS on $H(s)$ until convergence or until epoch limit of 100 is reached;

The NNQS will be trained with a progressive training schedule that mimics the quantum annealing process and follows Algorithm 1. \hat{H}_c is the problem Hamiltonian and \hat{H}_0 is a Hamiltonian with linear biases as 1 and no quadratic terms. The normalised anneal fraction s is increased in small steps while the NNQS is trained to converge, miming the system Hamiltonian's slow change in quantum annealing. The NNQS also remains in the ground state throughout the training.

The parameters of the NNQS, θ , are updated with a Variational Monte Carlo approach to minimise the energy expectation value. The general training algorithm is described as follows:

1. Initialize the NNQS parameters θ randomly.
2. Sample a set of m configurations \mathbf{s} from the probability distribution defined by the NNQS, $\mathbb{P}(s) = |\Psi(s; \theta)|^2$, with a Markov Chain Monte Carlo technique such as Gibbs sampling or the Metropolis-Hastings algorithms section 2.5.
3. Calculate the energy expectation value as the average energy of the sampled configurations with the Hamiltonian, which will form an unbiased estimate of the true energy expectation value.
4. Compute the gradients of the energy expectation value with respect to the NNQS parameters using backpropagation.

5. Update the parameters with gradient-based optimisers such as Root Mean Squared Propagation (RMSprop) (Tieleman, 2012).
6. Repeat steps 2-5 for some iterations or until convergence is reached.

The appendix in (Carleo & Troyer, 2017) fully describes the NNQS training algorithm. Each training process runs for at most 1000 epochs, and the model weights are updated with the RMSprop optimiser with a learning rate of 0.001. The candidate solution will be the best among the final 1000 sampled configurations. All NNQS experiments were run on a 32 Core AMD 7543P Processor and an NVIDIA A100 40GB GPU with 500GB of RAM.

4.1.3 Quantum Approximate Optimization Algorithm

We will implement QAOA in Qiskit, an open-source software development kit for quantum computing, with $p = 1$ using a backend hosted on the IBM Quantum Platform (IBMQ) (Qiskit contributors, 2023). Even though IBMQ allows for access to quantum computers with up to 127 qubits, it is limited by long wait times (> 5 hours for each problem) for a benchmarking experiment. Instead, we will utilise the IBMQ simulator, which can handle quantum circuits with up to 32 qubits. With an input Hamiltonian \hat{H}_c , we will use the default mixing Hamiltonian of $\hat{H}_0 = \sum_i \hat{\sigma}_x^{(i)}$ and a set of Hadamard gates applied to all qubits as the initial state. The Qiskit transpiler is used to optimise the decomposition of the operators into their individual parametrised quantum gates as shown in Figure 4.3 (IBM, 2024).

The parameters of QAOA, (γ, β) are updated to minimise the energy expectation value. The general training algorithm is described as follows:

1. Initialise the circuit in the initial state $|+\rangle^{\otimes n}$ with each qubit is in a superposition state. and random initial γ, β .

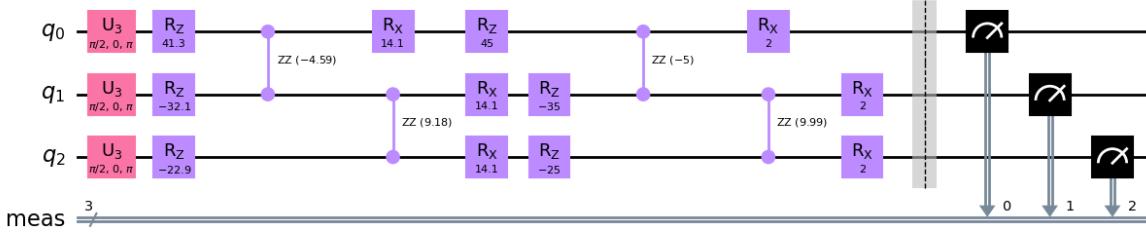


Figure 4.3: Circuit diagram with the Hadamard gates, problem Hamiltonian operator, and mixing Hamiltonian operator and (left to right)

2. Construct the operators $U_C(\gamma), U_B(\beta)$ with γ, β .
3. Calculate the energy expectation value as the average energy of the sampled configurations by repeatedly measuring the final states of the qubits.
4. Use a classical derivative-free optimiser such as Constrained Optimisation by Linear Approximation (COBYLA) to optimise the parameters γ, β to minimise the energy expectation value.
5. Repeat steps 2-4 for some iterations or until convergence is reached.

The experiments were run on the IBMQ cloud with the `ibmq_qasm_simulator`, a general-purpose simulator for quantum circuits, and the COBYLA optimiser for updating the circuit parameters. To measure the optimal performance of the QAOA algorithm, we will assume an ideal circuit without a quantum noise model. We will repeatedly sample the final circuit with optimised parameters 1000 times, and the candidate solution will be chosen as the best solution among the sampled configurations.

4.1.4 GUROBI Optimizer

The GUROBI optimiser is used as a state-of-the-art classical commercial solver, which is free for academic use (Gurobi Optimization, LLC, 2023). As the GUROBI optimiser

supports QUBO problems, we construct the QUBO matrix directly using the Gurobi Python library and run the optimiser locally for 10 minutes for each input problem. If the optimiser finishes before the cutoff, the candidate solution is guaranteed to be the optimal configuration. Otherwise, we would use the best solution within the cutoff time as a candidate solution. All experiments were run on a 32 Core AMD 7543P Processor and an NVIDIA A100 40GB GPU with 500GB of RAM using Gurobi Optimizer version 10.0.3.

4.1.5 Fixstars Amplify QUBO Solver

The Fixstar Amplify QUBO solver is a commercial simulated annealing-based QUBO solver that has limited free access (Dunning et al., 2018). As the Fixstar solver supports QUBO problems, we submit the QUBO matrix directly using the Fixstar API and run the solver with the highest allowed time limit of 100 seconds for each input problem. The solver is implemented on GPUs and runs on Fixstar remote servers, accessed with an API token.

4.2 Benchmark datasets

We use three randomly generated problem sets to benchmark our QUBO solvers. These problems were chosen as they are commonly used to represent NP-hard problems to measure the performance of QUBO solvers and are relatively straightforward to encode into QUBO form. They are the not-all-equal 3-satisfiability (NAE3SAT), max-cut, and the Sherrington-Kirkpatrick (SK) model. The NAE3SAT and max-cut problem sets are macro benchmarks (application-based) to represent practical combinatorial optimisation problems. In contrast, the SK model problem set is a microbenchmark designed to test the performance of the solvers. Each problem set comprises problems of 13 sizes, ranging

from 10 to 300¹. 20 different random problems are generated² for each problem size n for a total of 260 problems per problem set. Each problem is also first formulated in either the QUBO or Ising form, and conversion between them follows subsection 2.2.2 and subsection 2.2.3.

Not-all-equal 3-satisfiability (NAE3SAT)

The NAE3SAT problem is a variant of the boolean satisfiability problem where each problem instance consists of n boolean variables (x_1, x_2, \dots, x_n) and m clauses that each combine three literals, which can be a variable or its negation. The objective is to find an assignment of boolean values such that the three values in each clause are not all the same, i.e., each clause has at least one true and one false value. The ratio of clauses to variables $\rho = \frac{m}{n}$ is a problem parameter, and NAE3SAT problems are known to transition from being satisfiable to unsatisfiable at around $\rho = 2.1$ (Achlioptas, Chtcherba, Istrate, & Moore, 2001). We generate random NAE3SAT problems with $\rho = 2.1$ using the `random_nae3sat` generator from the `dimod` D-wave Python library, which "randomly samples clauses from the space of 3-variable clauses with replacement" (D-Wave Systems, 2024d).

To convert a NAE3SAT problem into an Ising problem, we represent each boolean variable as a spin variable and turn each clause into a Hamiltonian term. For example, for the clause $(x_1, x_2, \neg x_3)$, we use the Hamiltonian term $H(s_1, s_2, s_3) = s_1 \cdot s_2 + s_2 \cdot (-s_3) + s_1 \cdot (-s_3)$ which has a value of 3 when $x_1 = x_2 = \neg x_3$ and -1 otherwise. The final Hamiltonian \hat{H}_c is simply a sum of the individual Hamiltonian terms for each clause.

Max-cut

The max-cut problem aims to find a partition of the vertices of a graph $G = (V, E)$ into V_0, V_1 with $V = V_0 \cup V_1$ and $V_0 \cap V_1 = \emptyset$, such that the number of edges crossing V_0 and

¹ $n \in [10, 15, 20, 25, 30, 35, 50, 75, 100, 150, 200, 250, 300]$

²random seed is chosen to be from 0 – 19

V_1 is maximised. We will use the Erdos-Renyi model to generate random graphs with n vertices and a probability $p = 0.25$ to include each edge.

To convert a max-cut problem into a QUBO problem, we represent the assignment of each vertex as a binary variable ($x_v = i$ if $x \in V_i$) and use the objective function $f(\mathbf{x}) = \sum_{e=(u,v) \in E} -x_u - x_v + 2x_u x_v$, where each term has a value of -1 when $x_u \neq x_v$ and 0 otherwise. If a problem has a max-cut value of C , the objective function would have a minimum value of $-C$.

Sherrington-Kirkpatrick (SK) model

The Sherrington Kirkpatrick (SK) model is an Ising problem with a random Hamiltonian of the form $\hat{H}_c = \frac{1}{\sqrt{n}} \sum_{1 \leq i < j \leq n} J_{ij} s_i s_j$ where $J_{ij} \sim \mathcal{N}(0, 1)$ are independent standard Gaussian variables. The energy landscape of the SK model is complex and has exponentially many local minima with a unique global minimum separated by high energy barriers (D. J. Thouless & Palmer, 1977). This many-valley structure implies that it is challenging to find the exact solution of the model. We generate random Gaussian variables with a random normal generator from the NumPy Python library.

4.3 Performance evaluation

To evaluate the performance of each solver, we will use two performance metrics:

1. The success probability, which is the probability of finding a solution with the lowest energy among all solutions found by the 5 solvers. Since we are generating 20 problems of each type and size, the success probability for each solver is:

$$\bar{p} = \frac{\text{number of lowest energy solutions found}}{20} \quad (4.2)$$

2. The average normalized energy, used in (Willsch et al., 2020), which is:

$$\bar{r}_{solver} = \frac{1}{20} \sum_{i=1}^{20} \frac{E_{max} - E_{solver}}{E_{max} - E_{min}} \quad (4.3)$$

where E_{max} and E_{min} are the energies of the worst and best solutions found by all solvers, and E_{solver} is the energy that the solver found. Note that if a solver finds the best solution, it gets a normalised energy of 1; if it finds the worst solution, it gets a normalised energy of 0. All solvers get a normalised energy of 1 if all solutions have the same energy.

The average normalised energy measures the quality of the solutions that the solvers produce, while the success probability measures the solvers's ability to find the best solution.

Chapter 5

Benchmarking QUBO solvers

This chapter first introduces related benchmarking work for QUBO problems and QUBO solvers. Then, we present our results for the solvers and datasets used for this study.

5.1 Related benchmarking work

One of the first benchmarking works for quantum annealing was conducted by Denchev et al. (2016), who measured the performance of D-Wave quantum annealing on the older D-Wave 2X machine using specially crafted problems that have tall and narrow energy barriers separating local minima. Quantum annealing is expected to be 1.8×10^8 times faster than simulated annealing, which tends to fail with problems with such an energy landscape.

Willsch et al. (2020) evaluated the performance of QAOA on the IBMQ backend and the D-Wave solver using instances of MaxCut and 2-satisfiability problems with up to 18 variables. The performance of the QAOA algorithm is inconsistent and underperforms quantum annealing in their problem set. More recently, Pelofske, Bärtschi, and Eidenbenz (2023) also compared the performance of QAOA on the IBMQ backend and D-Wave quantum annealing on randomly generated Ising problems with cubic interaction terms

and found that quantum annealing had superior performance over QAOA for all problem sizes.

Gomes, McKiernan, Eastman, and Pande (2019) showed that the NNQS solving method with an RBM architecture produces good quality solutions for the max-cut problem with graph sizes of up to 256. Khandoker, Abedin, and Hibat-Allah (2023) uses recurrent neural networks as the NNQS architecture for the max-cut and travelling salesman problem and found that it outperforms SA. However, no direct study has compared performance across quantum annealing, QAOA, and NNQS.

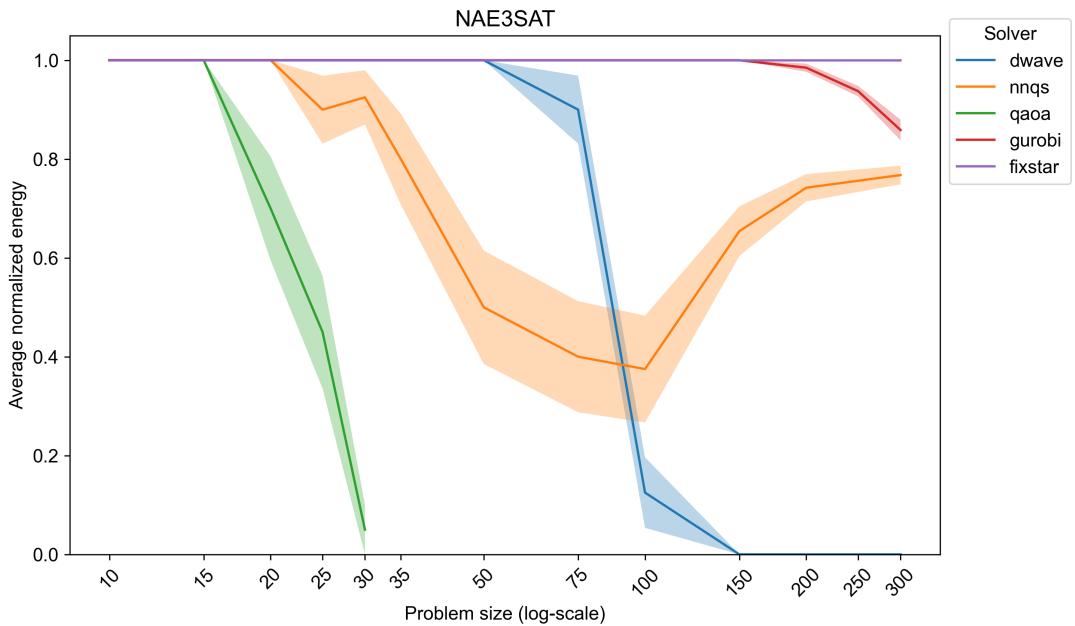
5.2 Results and Discussion

Performance is shown for each dataset, accompanied by error bars representing each data point's unbiased standard error of the mean. Graphs with problem sizes on the x-axis are plotted with a log scale.

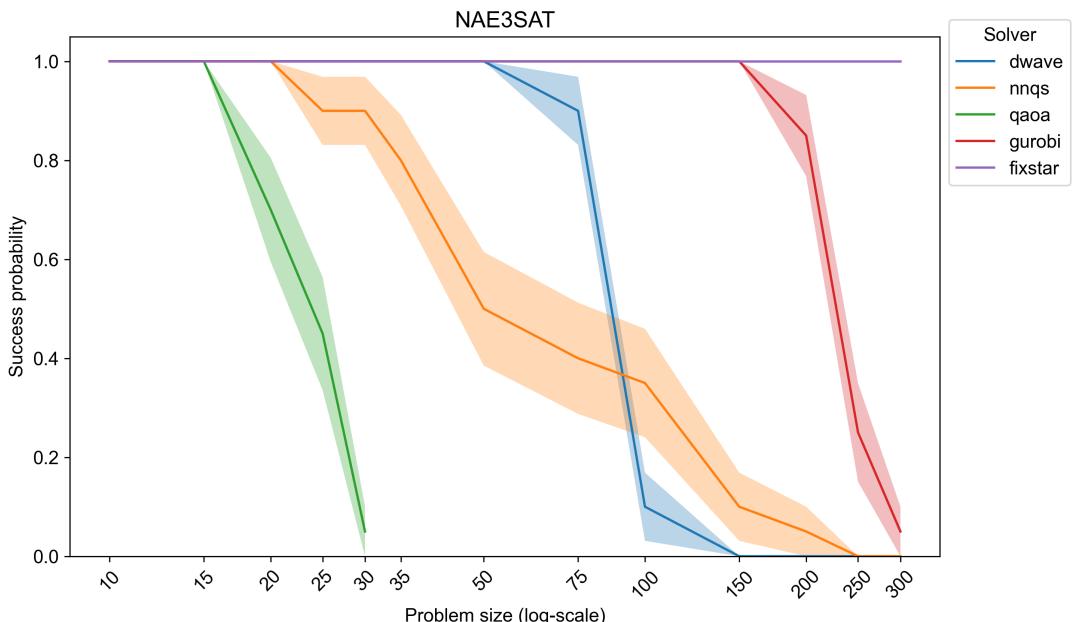
5.2.1 NAE3SAT

Performance by size for the NAE3SAT dataset in Figure 5.1 and average performance is shown in Figure 5.2. The D-wave solver and NNQS could both solve problems up to $n = 300$. However, multiple embedding requests were required for problems of size 300 for the D-wave Pegasus topology, which suggests that $n = 300$ might be near the D-wave size limit for the NAE3SAT problem. QAOA could only solve problems of up to $n = 30$ due to the limitations on the number of qubits of the simulator.

In terms of performance, the D-wave solver performs well up to $n = 50$ with a success probability of 1. For larger problem sizes, the performance of the D-wave solver drops off sharply. The NNQS performs well up to $n = 20$, with the success probability and normalised energy gradually decreasing until $n = 300$. The QAOA solver performs well up to $n = 15$, and performance decreases until $n = 30$. Between the classical solvers, the

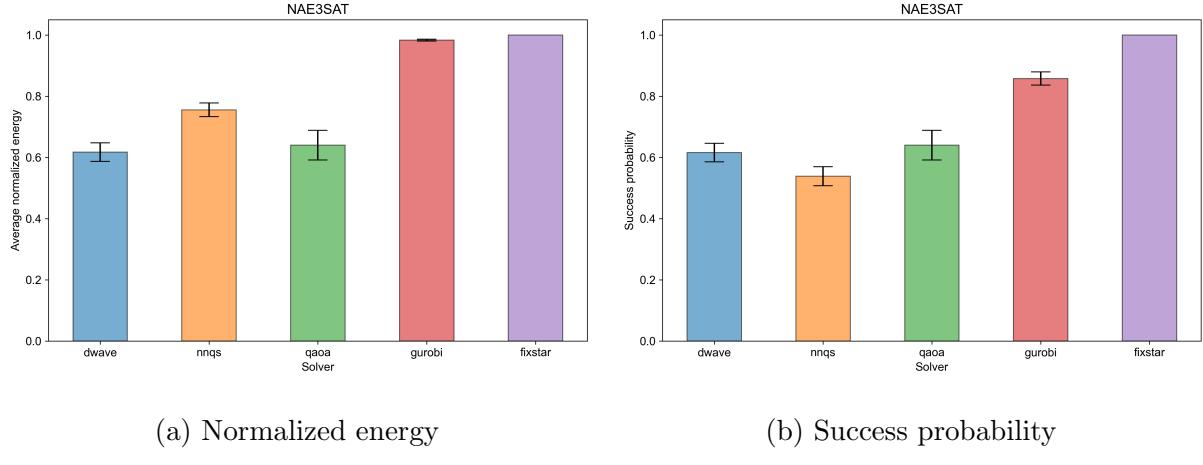


(a) Normalized energy



(b) Success probability

Figure 5.1: Performance of different solvers for NAE3SAT by problem size



(a) Normalized energy

(b) Success probability

Figure 5.2: Average performance of different solvers for NAE3SAT

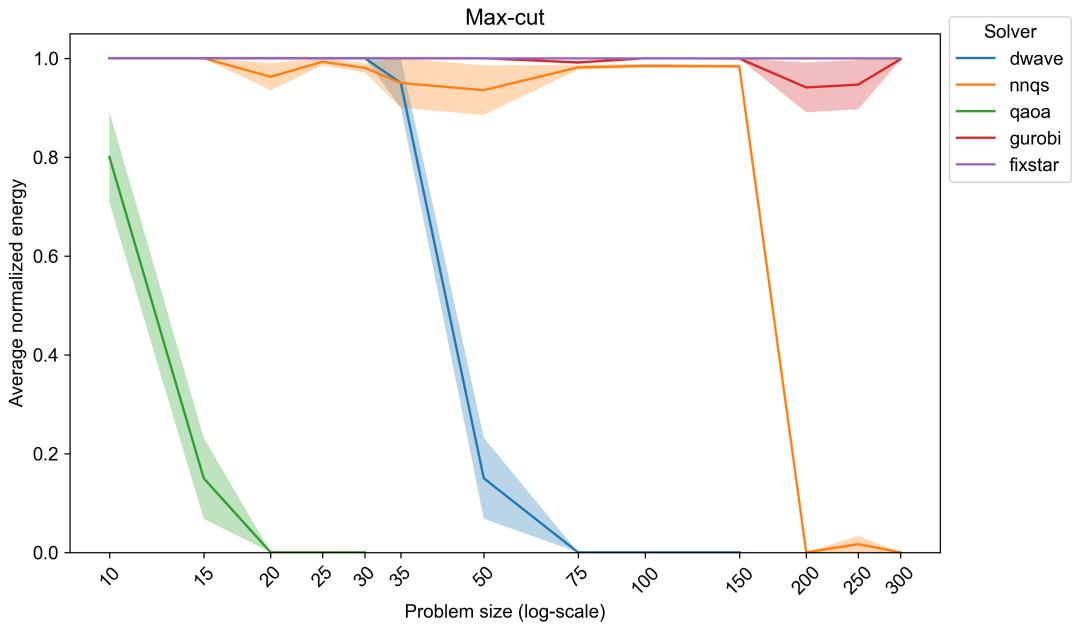
Fixstars QUBO solver performs better than the GUROBI optimiser at larger problem sizes (> 150). Both classical solvers outperform the quantum-inspired solvers.

Overall, the NNQS has the highest average normalised energy among the three quantum-inspired solvers but has the lowest success probability, which is likely due to it being able to solve problems of larger sizes than the D-wave Annealer and QAOA solver cannot handle. The QAOA solver has the highest success probability but can only handle problems up to $n = 30$.

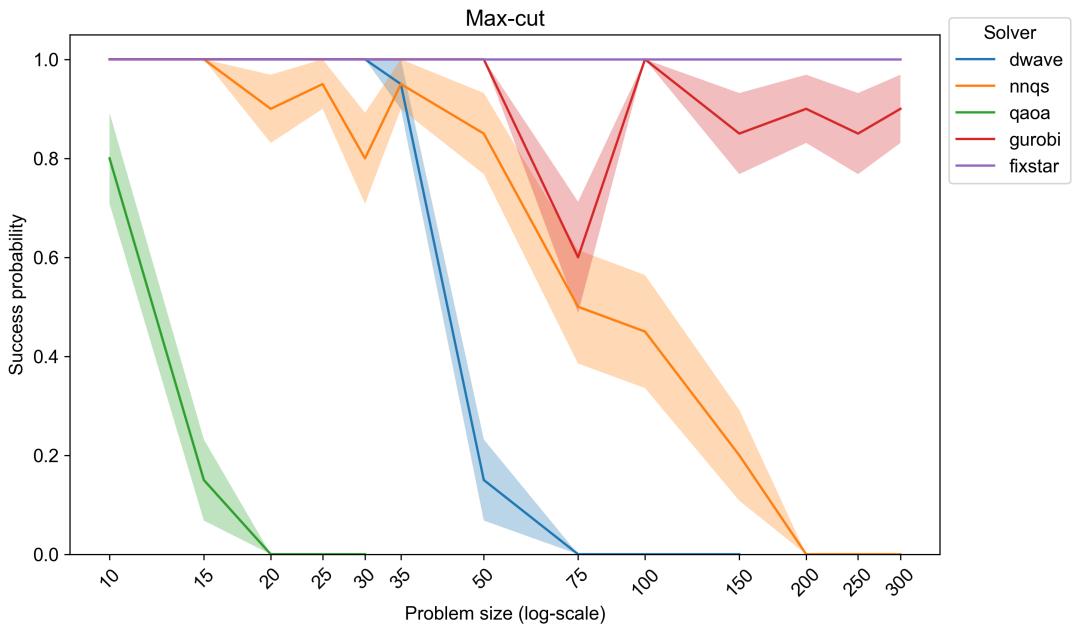
5.2.2 Max-cut

Performance by size for the max-cut dataset is shown in Figure 5.3, and average performance is shown in Figure 5.4. For the max-cut problem, the D-wave solver could only handle problem sizes up to $n = 150$ due to the need for minor embedding onto the pegasus topology. QAOA solved problems of up to $n = 30$ due to the limitations of the simulator.

In terms of performance, the D-wave solver performs well up to $n = 30$, and performance drops off sharply for larger problems. The NNQS performs well at $n = 150$, although the success probability decreases for problem sizes over 50. The QAOA solver performs well only for $n = 10$, with performance decreasing for larger problems.

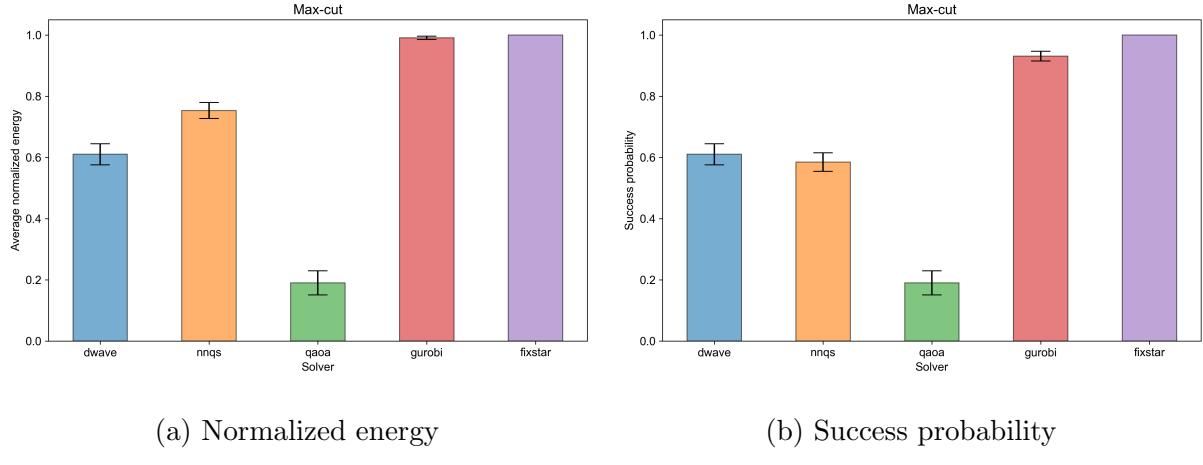


(a) Normalized energy



(b) Success probability

Figure 5.3: performance of different solvers for max-cut by problem size



(a) Normalized energy

(b) Success probability

Figure 5.4: Average performance of different solvers for max-cut

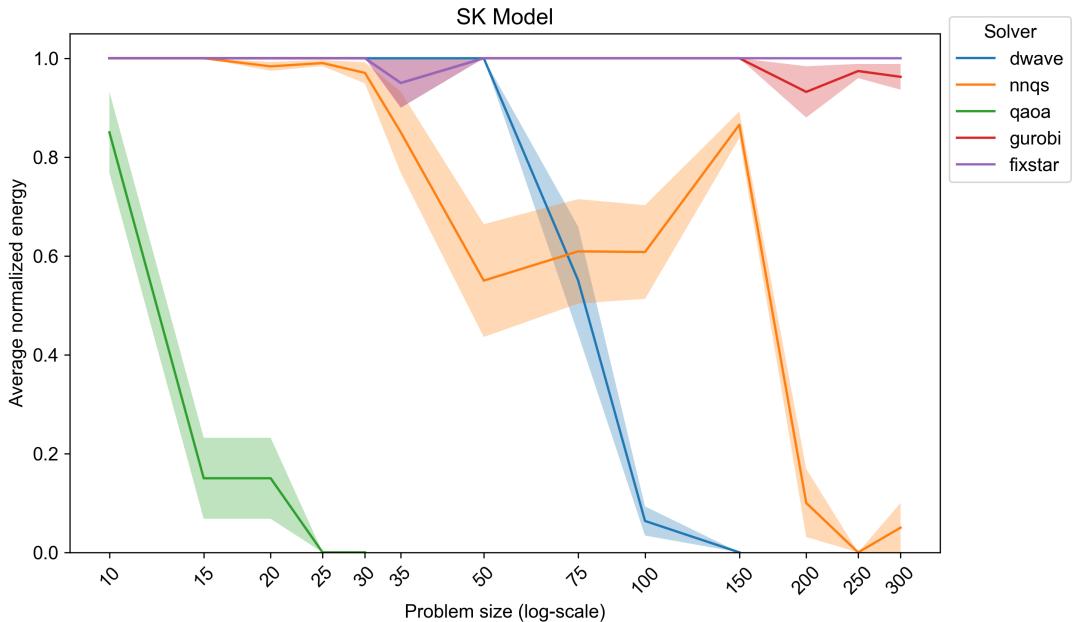
Overall, the NNQS has the highest average normalised energy among the three quantum-inspired solvers and has a slightly lower success probability than the D-wave solver. The QAOA solver performs poorly in both metrics.

5.2.3 SK model

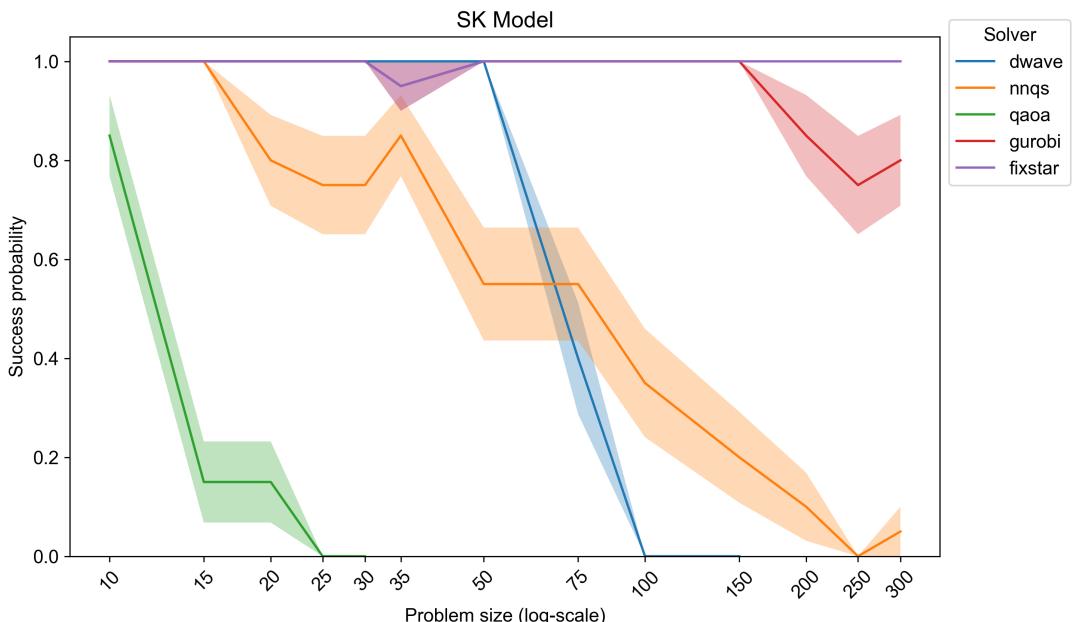
Performance by size for the SK model dataset is shown in Figure 5.5, and average performance is shown in Figure 5.6. For the SK model, the D-wave solver could only handle problem sizes up to $n = 150$ due to the need for minor embedding onto the pegasus topology. The SK model is fully connected, which makes embedding difficult for the D-wave QPU. QAOA solved problems of up to $n = 30$ due to the limitations of the simulator.

Due to its multi-valley energy landscape, the SK model presents a difficult problem for all QUBO solvers. The D-wave solver performs well up to $n = 30$, gradually decreasing performance for larger problems. The NNQS follows a similar trend, although it can solve problems of larger sizes and performs better at sizes of $= 100, 150$. The QAOA solver has consistently poor performance across problem sizes.

Overall, the D-wave annealer has the highest average normalised energy and success probability among the three quantum-inspired solvers. The NNQS is slightly worse in

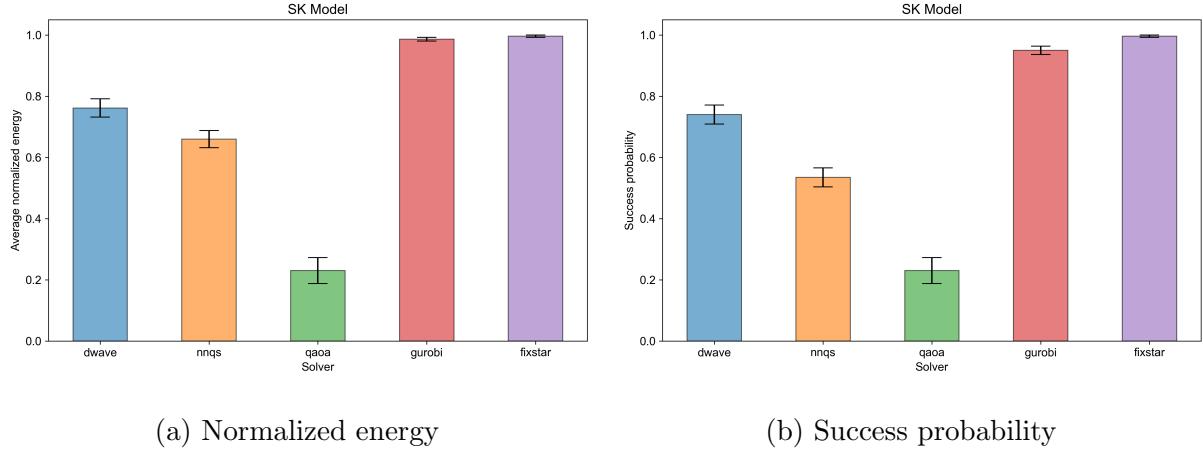


(a) Normalized energy



(b) Success probability

Figure 5.5: performance of different solvers for SK model by problem size



(a) Normalized energy

(b) Success probability

Figure 5.6: Average performance of different solvers for SK model

both metrics, while the QAOA solver performs poorly for both metrics.

5.3 Time-Constrained Solver Comparison

We also measured the runtime for each solver for each problem and the average runtime across all problems with the same size n shown in Figure 5.7. Runtimes split by problem type can be found in Appendix D

For the D-wave solver, the average runtime increases approximately linearly from 0.128s for $n = 10$ to 0.184s for $n = 50$ and 0.292s for $n = 300$.

For the NNQS solver, the runtime does not increase significantly from $n = 10$ to $n = 150$ and remains around from 250s to 350s but increases sharply for larger $n \geq 200$ which could be due to memory issues with the GPU or the sampling may have taken a longer time to converge.

The QAOA solver's runtime remains stable from $n = 10$ to $n = 20$ at around 38s. However, it increases rapidly to 6847s at $n = 30$ due to the need for more optimisation iterations and greater computational resources for the simulation.

The GUROBI optimiser was set with a maximum time limit of 600s but could finish solving before the time limit for most problems with $n \leq 35$. The Fixstar solver was

configured with a maximum time limit of 100s, the maximum possible duration, and each solve utilised the entire time limit.

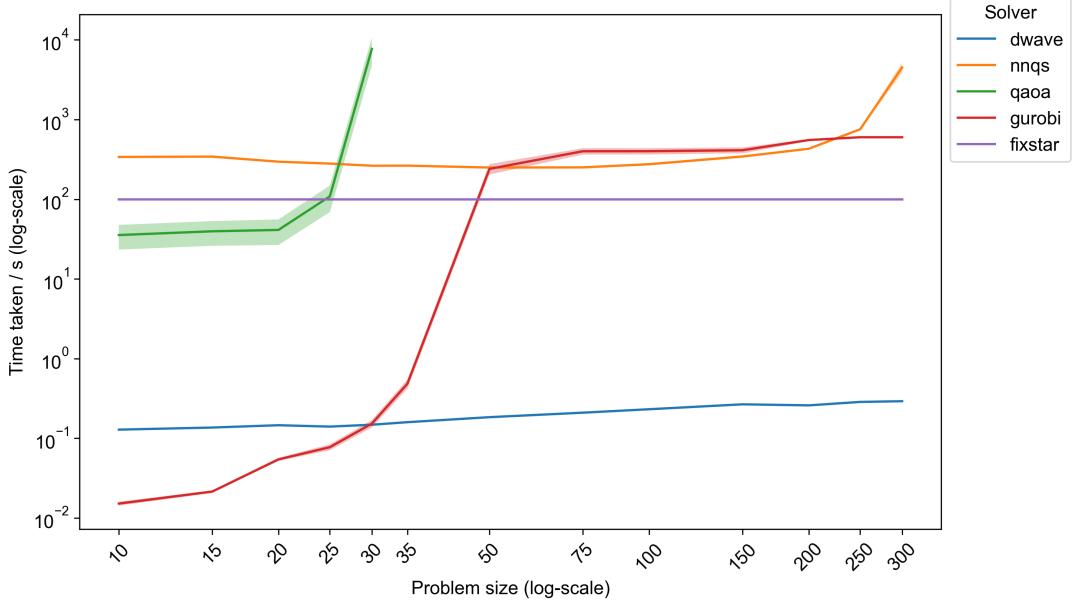


Figure 5.7: Average runtime in log scale taken by different solvers for QUBO problems by size

Using the average runtime of the D-wave solver, we conducted a second run of the benchmarking to measure the performance of the D-wave solver against the classical solvers—GUROBI and Fixstar. This experiment aimed to test if the D-wave solver could outperform the classical solvers if they were limited to the same runtime. For a problem of a specific type and size, the classical solvers were run with a maximum runtime equal to the average time required by the D-wave solver for problems of equivalent type and size shown in Table 5.1.

The results are shown in Figure 5.8. For each problem type, the performance of the GUROBI solver drops before the D-wave solver, and there are problem types and sizes where the D-wave solver outperforms the GUROBI solver when the runtime is matched. The D-wave solver outperforms GUROBI for NAE3SAT with $n = 50, 75$, max-cut with $n = 30, 35$, and SK model with $n = 20, 35$. However, when the problem sizes increase

n	10	15	20	25	30	35	50	75	100	150	200	250	300
NAE3SAT	0.133	0.135	0.141	0.149	0.138	0.156	0.173	0.184	0.201	0.243	0.259	0.286	0.292
Max-cut	0.127	0.136	0.1560	0.130	0.155	0.160	0.183	0.223	0.247	0.278	-	-	-
SK model	0.125	0.137	0.140	0.140	0.150	0.161	0.195	0.221	0.245	0.278	-	-	-

Table 5.1: Average runtime (seconds) of the D-wave solver by problem type and size. Dashes indicate that the D-wave solver could not embed problems of that size.

even further, the D-wave solver performs poorly, possibly due to the increased noise of the quantum annealer. The Fixstar solver remains the top-performing solver across all problem types and sizes, even when the runtime is matched with the D-wave solver. The results show that the D-wave solver can outperform classical solvers like GUROBI for specific problem sizes when the runtime is matched.

5.4 Solver logs

During the benchmarking experiments, supplementary metadata from the D-wave and QAOA solver was recorded. We recorded the embedded qubit count for the D-wave solver, which tends to grow differently for different problem types. For the QAOA solver, we recorded the number of quantum gates and the circuit depth, which can help quantify the complexity of the quantum circuit used. A metadata summary is available in Appendix E.

5.5 Conclusion

When comparing the three quantum-inspired solvers, NNQS has the best normalised energy for the NAE3SAT and max-cut dataset, while the D-wave solver has the best normalised energy for the SK model performance. NNQS also does the best in normalised energy when averaged across the three datasets. NNQS tends to do better in normalised

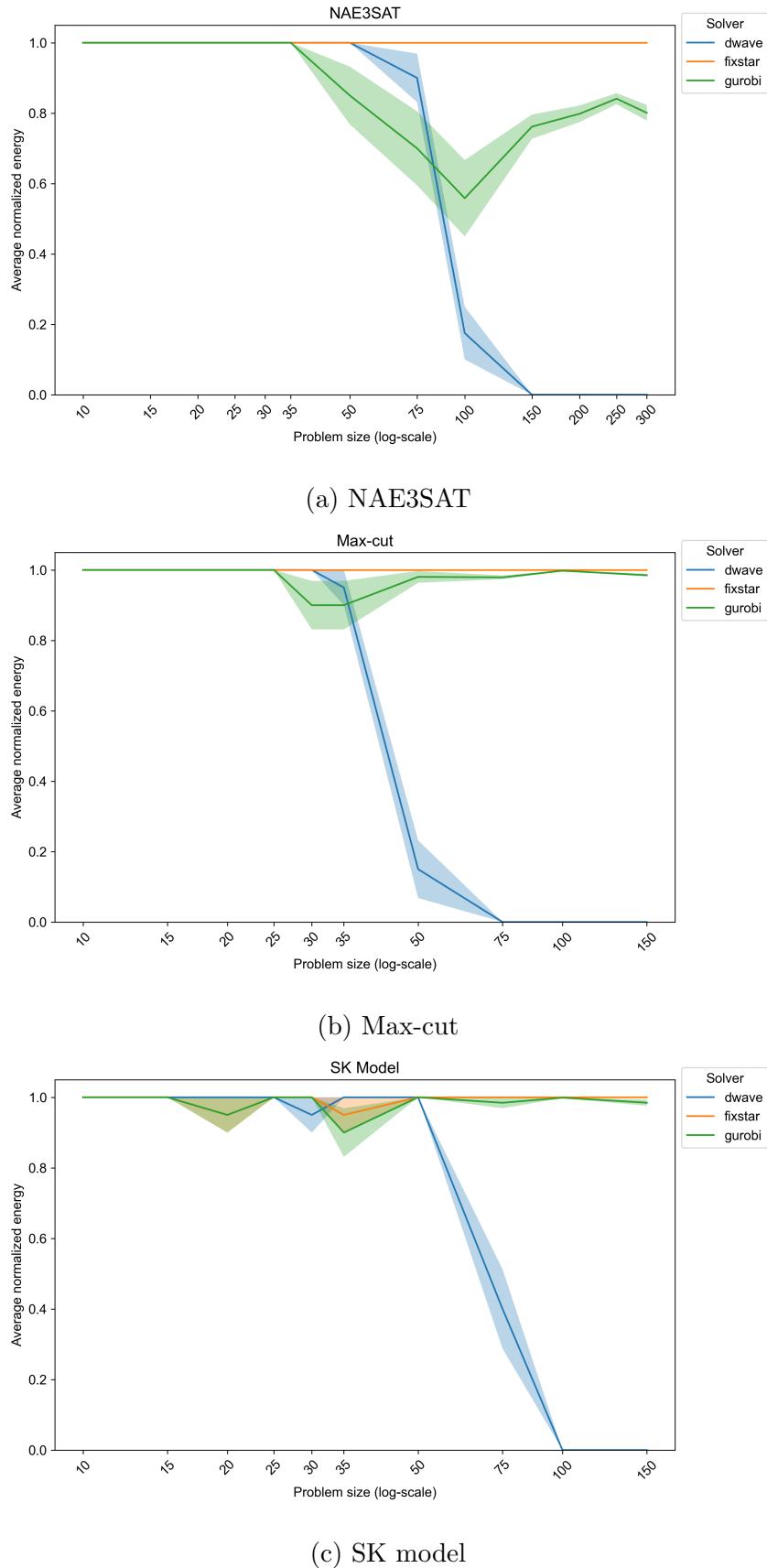


Figure 5.8: Performance of D-wave solver against GUROBI and Fixstar by problem type and size

energy since it minimises the energy expectation value, which optimises the average energy of samples but does not necessarily optimise for the highest probability of sampling the best solution.

QAOA has the highest success probability for the NAE3SAT dataset. However, it is important to note that it could only handle problems with up to 30 variables. The D-wave solver has the best success probability for the max-cut and SK model datasets and the average success probability across the three datasets.

Table 5.2 and Table 5.3 show the average normalised energy and success probability for different solvers for each dataset and the average across all datasets. Across all datasets and both metrics, the Fixstar QUBO solver has the best performance, consistently returning the best solutions out of all solvers.

	D-wave	NNQS	QAOA	GUROBI	Fixstar
NAE3SAT	0.617	0.755	0.640	0.983	1.00
Max-cut	0.610	0.753	0.190	0.991	1.00
SK model	0.761	0.664	0.230	0.990	1.00
Average	0.663	0.724	0.353	0.988	1.00

Table 5.2: average normalised energy for different solvers

	D-wave	NNQS	QAOA	GUROBI	Fixstar
NAE3SAT	0.615	0.538	0.640	0.858	1.00
Max-cut	0.610	0.585	0.190	0.931	1.00
SK model	0.740	0.538	0.230	0.954	1.00
Average	0.655	0.554	0.353	0.914	1.00

Table 5.3: Success probability for different solvers

Chapter 6

NNQS exploration

This chapter explores the effects of different architectures and training schemes that can be used to train NNQS to solve QUBO problems. All experiments in this section were run on a subset of the entire dataset with problem sizes of 10, 25, 50, 75, 200, 250 and 10 problems for each problem type and size. The problem evaluation metrics are also calculated by considering the solutions from the different types of NNQS to draw a clearer comparison between architectures and training schemes.

6.1 Architectures and training algorithms

We will utilise the Restricted Boltzmann Machine (RBM) and the Multilayer Perceptron (MLP) as the architecture for NNQS. For a given input problem with n variables, the RBM model will have n visible nodes and $5n$ hidden nodes, while the MLP will have n input nodes, 1 hidden layer of size $5n$ and 1 positive real output node. The RBM uses the sigmoid function, while the MLP uses the ReLU activation function. We use Gibbs sampling for the RBM and Metropolis-Hastings sampling for the MLP; both sampling methods are detailed in section 2.5.

We will also compare three training algorithms for NNQS—progressive, direct, and

continuous. The progressive training algorithm follows Algorithm 1. In direct training, described in Algorithm 2, the normalised anneal fraction s is held constant at 1 for all epochs. In continuous training, described in Algorithm 3, the normalised anneal fraction is increased gradually every epoch and the NNQS is not trained to convergence.

Algorithm 2 NNQS Direct Training

Require: Problem Hamiltonian \hat{H}_c

Ensure: Trained NNQS

Initialize NNQS with random weights;

Set $H \leftarrow B(1)\hat{H}_c$;

Train NNQS on H until convergence or until epoch limit of 1000 is reached;

Algorithm 3 NNQS Continuous Training

Require: Problem Hamiltonian \hat{H}_c

Ensure: Trained NNQS

Initialize NNQS with random weights;

for $s \in [0.001, 1.0]$ step 0.001 **do**

 Set $H(s) \leftarrow A(s)\hat{H}_0 + B(s)\hat{H}_c$;

 Train NNQS on $H(s)$ for 1 epoch;

Direct training serves as a baseline for directly training a neural network with the cost function as the problem Hamiltonian. Progressive training most closely resembles the quantum annealing process, where the system is kept at the ground state by training until convergence in each increment of s . Continuous training combines the other two by slowly incrementing s but never reaching convergence.

6.2 Results and Discussion

Performance is shown for each dataset, accompanied by error bars representing each data point's unbiased standard error of the mean. Graphs with problem sizes on the x-axis are plotted with a log scale. The graphs for performance by size are shown in Appendix C.

6.2.1 NAE3SAT

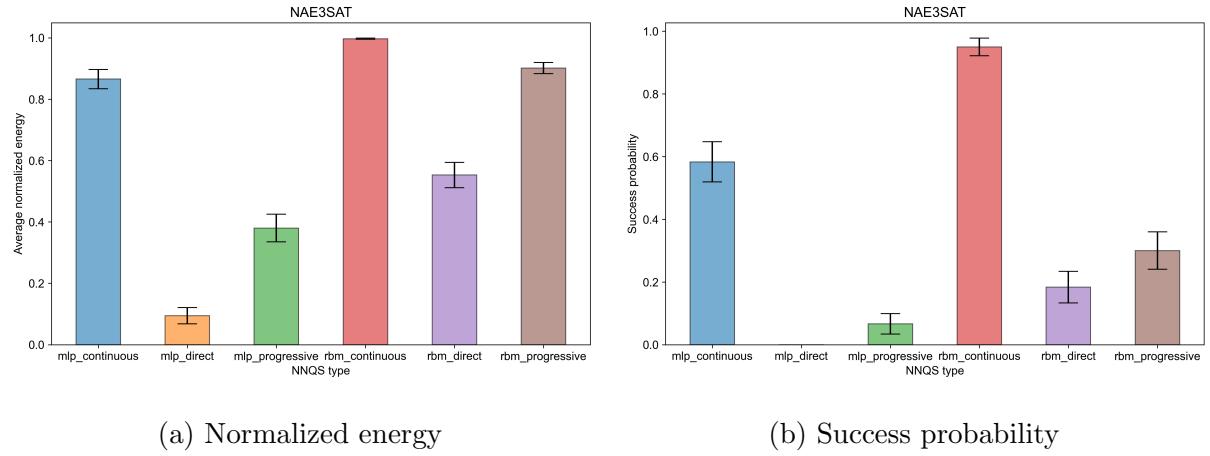


Figure 6.1: Average performance of different NNQS types for NAE3SAT

For the NAE3SAT dataset, the continuous training algorithm with the RBM performs the best in average performance and success probability when averaged across all sizes, shown in Figure 6.1. In the performance by problem size, the continuous training algorithm with the RBM performed the best in normalised energy and success probability, shown in Figure C.1, except with a problem size of 50, which is likely due to variance in the randomly generated data set.

6.2.2 Max-cut

For the maxcut dataset, the continuous training algorithm with the RBM performs the best in average performance and success probability when averaged across all sizes, shown

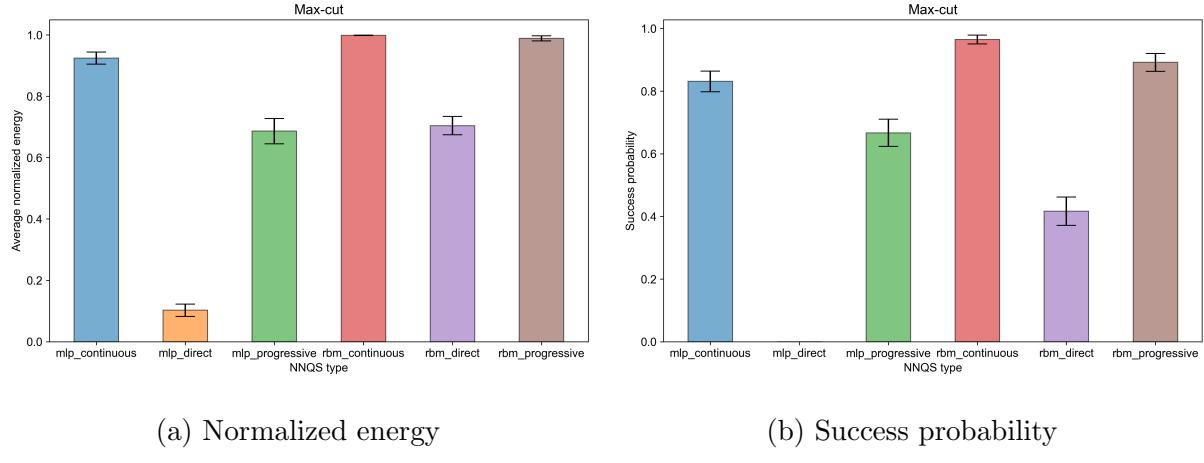


Figure 6.2: Average performance of different NNQS types for max-cut

in Figure 6.2. In the performance by problem size, the continuous training algorithm with the RBM performed the best in normalised energy and success probability, shown in Figure C.2, except with a problem size of 25, which is again likely due to variance in the randomly generated data set. However, the gap in success probability between the models is relatively small, which likely implies that the max-cut problem is easier in general than the NAE3SAT problem.

6.2.3 SK model

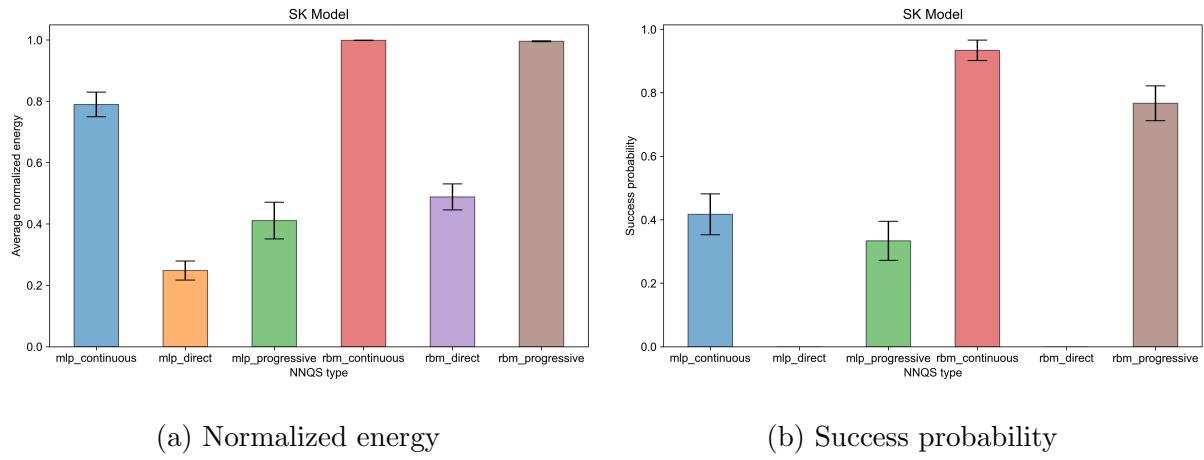


Figure 6.3: Average performance of different NNQS types for SK model

For the SK model dataset, the continuous training algorithm with the RBM performs the best in average performance and success probability when averaged across all sizes, shown in Figure 6.3. The performance averaged across all sizes, shown in Figure 6.3, also highlights that the RBM with a continuous training algorithm performs the best. However, it is interesting that the direct training schemes perform poorly with small problem sizes (≤ 50) but are relatively better at higher problem sizes (100, 250).

6.3 Conclusion

Table 6.1 and Table 6.2 show the average normalised energy and success probability for different types of NNQS for each dataset and the average across all datasets.

Across all the problem sets, the RBM has a better average performance than the MLP in average normalised energy and success probability for each training algorithm. In terms of training schemes, direct training performs the worst among the 3, and continuous training performs slightly better than progressive training. Using the RBM with a continuous training scheme gives the highest normalised energy and success probability across all datasets.

Even though progressive training more closely mimics the quantum annealing process in D-wave solvers, the sudden change of Hamiltonian may have led to large gradient changes that may have made it more difficult for the network to converge, leading to poorer training. Continuous training gradually changes the Hamiltonian, which limits the gradient magnitudes and may result in better training. Direct training is expected to have poor performance as it tends to get stuck in local minima.

	MLP			RBM		
	Continuous	Direct	Progressive	Continuous	Direct	Progressive
NAE3SAT	0.866	0.118	0.352	0.997	0.511	0.910
Max-cut	0.924	0.102	0.686	0.998	0.704	0.988
SK model	0.790	0.248	0.411	0.999	0.488	0.995
Average	0.860	0.156	0.483	0.998	0.568	0.965

Table 6.1: average normalised energy for different NNQS types

	MLP			RBM		
	Continuous	Direct	Progressive	Continuous	Direct	Progressive
NAE3SAT	0.583	0.029	0.062	0.950	0.167	0.364
Max-cut	0.831	0.000	0.667	0.965	0.417	0.892
SK model	0.417	0.000	0.333	0.933	0.000	0.767
Average	0.610	0.010	0.354	0.949	0.194	0.674

Table 6.2: Success probability for different NNQS types

Chapter 7

Conclusion

This chapter summarises our study's contributions and limitations. It also makes recommendations for further work that future projects could investigate.

7.1 Contributions

In this study, we have benchmarked 5 QUBO solvers:

1. D-Wave Quantum Annealing
2. Neural Network Quantum States (NNQS)
3. Quantum Approximate Optimization Algorithm (QAOA)
4. GUROBI Optimizer
5. Fixstars Amplify QUBO Solver

3 datasets were used which comprised of 3 types of combinatorial optimization problems:

1. Not-all-equal 3-satisfiability (NAE3SAT)
2. Max-cut

3. Sherrington-Kirkpatrick model (SK model)

Our results show that the D-wave and NNQS solvers perform best among the quantum-inspired solvers. The NNQS solver is generally better than the D-wave solver except for the SK model dataset. The QAOA solver achieves generally poor performance across datasets and is not comparable due to the lack of large-scale gate-based quantum computers. All three quantum-inspired solvers underperform the two classical solvers, with the simulated-annealing-based Fixstar Amplify QOBO solver achieving the best performance out of all solvers for all datasets.

We also investigated the NNQS solver with different architectures and training algorithms. The Restricted Boltzmann Machine (RBM) and the Multilayer Perceptron (MLP) were used as the underlying neural networks along with 3 different training algorithms—progressive, direct, and continuous. We found that using the RBM with a continuous training scheme gives the best performance across all datasets.

The primary constraint of our study came from the limitations of the QAOA solver, which was intended to be run on a gate-based quantum computer. Due to restricted availability, we could only use a simulator capable of handling only up to 30 variables. However, as the QAOA simulator does not model quantum noise and the results from the QAOA solver for small problems are not promising, QAOA on an actual quantum device would likely perform even worse for larger problems and is thus too interesting to benchmark in the current NISQ era. The other significant limitation was that the NNQS exploration dataset was relatively smaller due to runtime constraints. However, as there were still around 60 problems for each solver, the standard error for performance was small.

7.2 Future work

Future studies can explore more QUBO problem types and attempt to classify problems that are difficult for annealing-based solvers such as QA and SA but easier for other solvers such as QAOA. When gate-based quantum computers are readily available, the QAOA solver with larger p values ($p > 1$) could be benchmarked, likely leading to better performance.

For further work with NNQS, future studies could investigate if more modern machine learning models, such as Graph Neural Networks or Attention-based Neural Networks, can be used as the underlying architecture for NNQS and whether they provide better performance.

Future work could also investigate whether NNQS closely approximates the wave function of a D-wave solver in the quantum annealing process. This could be done by conducting a quench of the D-wave annealing process to take a snapshot of the intermediate state. Some initial work is detailed in Appendix F but has not been included in the main report as the results are not substantial.

References

- Achlioptas, D., Chtcherba, A., Istrate, G., & Moore, C. (2001). The phase transition in 1-in-k sat and nae 3-sat. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, , , 05, 2001.
- Alidaee, B. (2009). Minimizing absolute and squared deviation of completion times from due dates. *Production and Operations Management*, 3, , 01, 2009, 133 – 147.
- Alidaee, B., Glover, F., Kochenberger, G., & Rego, C. (2005). A new modeling and solution approach for the number partitioning problem. *JAMDS*, 9, , 01, 2005, 113–121.
- Barahona, F. (1982). On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10), , 1982, 3241.
- Baxter, R. J. (2016). *Exactly solved models in statistical mechanics*. Elsevier.
- Blekos, K., Brand, D., Ceschini, A., Chou, C.-H., Li, R.-H., Pandya, K., & Summer, A. (2024). A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, 1068, , 2024, 1–66.
- Born, M., & Fock, V. (1928). Beweis des adiabatensatzes. *Zeitschrift für Physik A Hadrons and Nuclei*, 51, , 03, 1928, 165–180.
- Carleo, G., & Troyer, M. (2017). Solving the quantum many-body problem with artificial neural networks. *Science*, 355, , 02, 2017, 602.
- Cipra, B. A. (1987). An introduction to the ising model. *The American Mathematical Monthly*, 94(10), , 1987, 937–959.
- D. J. Thouless, P. W. A., & Palmer, R. G. (1977). Solution of 'solvable model of a spin glass'. *The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics*, 35(3), , 1977, 593–601.
- D-Wave Systems (2024a). Annealing implementation and controls. https://docs.dwavesys.com/docs/latest/c_qpu_annealing.html. Accessed: 2024-01-04.
- D-Wave Systems (2024b). Minor-embedding. <https://docs.ocean.dwavesys.com/en/stable/concepts/embedding.html>. Accessed: 2024-01-04.

- D-Wave Systems (2024c). Qpu-specific characteristics. https://docs.dwavesys.com/docs/latest/doc_physical_properties.html. Accessed: 2024-01-04.
- D-Wave Systems (2024d). Solver docs. https://docs.ocean.dwavesys.com/en/stable/docs_dimod/reference/generated/dimod.generators.random_nae3sat.html. Accessed: 2024-01-04.
- Denchev, V. S., Boixo, S., Isakov, S. V., Ding, N., Babbush, R., Smelyanskiy, V., Martinis, J., & Neven, H. (2016). What is the computational value of finite-range tunneling? *Physical Review X*, 6(3), , 2016, 031015.
- Dunning, I., Gupta, S., & Silberholz, J. (2018). What works best when? a systematic evaluation of heuristics for max-cut and qubo. *INFORMS Journal on Computing*, 30, , 08, 2018, 608–624.
- Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm.
- Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., & Preda, D. (2001). A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516), , 2001, 472–475.
- Farhi, E., & Harrow, A. W. (2016). Quantum supremacy through the quantum approximate optimization algorithm.
- Geman, S., & Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, 6, , 1984, 721–741.
- Glover, F., Kochenberger, G., Hennig, R., & Du, Y. (2022). Quantum bridge analytics i: a tutorial on formulating and using qubo models. *Annals of Operations Research*, 314, , 07, 2022.
- Gomes, J., McKiernan, K. A., Eastman, P., & Pande, V. S. (2019). Classical quantum optimization with neural network quantum states. *arXiv preprint arXiv:1910.10675*, , , 2019.
- Griffiths, D. J., & Schroeter, D. F. (2018). *Introduction to quantum mechanics*. Cambridge University Press, 3 edition.
- Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual.
- Harris, R., Sato, Y., Berkley, A., Reis, M., Altomare, F., Amin, M., Boothby, K., Bunyk, P., Deng, C., Enderud, C., Huang, S., Hoskinson, E., Johnson, M., Ladizinsky, E., Ladizinsky, N., Lanting, T., Li, R., Medina, T., Molavi, R., & Yao, J. (2018). Phase transitions in a programmable quantum spin glass simulator. *Science (New York, N.Y.)*, 361, , 07, 2018, 162–165.
- Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1), , 1970, 97–109.

- Horn, R. A., & Johnson, C. R. (1990). *Matrix analysis*. Cambridge University Press.
- IBM (2024). Transpiler. <https://docs.quantum.ibm.com/api/qiskit/transpiler>. Accessed: 2024-01-04.
- Ising, E. (1925). Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1), , 1925, 253–258.
- Kadowaki, T., & Nishimori, H. (1998). Quantum annealing in the transverse ising model. *Physical Review E*, 58(5), , 1998, 5355.
- Katayama, K., & Narihisa, H. (2001). Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem. *European Journal of Operational Research*, 134(1), , 2001, 103–119.
- Khandoker, S. A., Abedin, J. M., & Hibat-Allah, M. (2023). Supplementing recurrent neural networks with annealing to solve combinatorial optimization problems. *Machine Learning: Science and Technology*, 4(1), , 2023, 015026.
- Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), , 1983, 671–680.
- Kochenberger, G., Hao, J.-K., Glover, F., Lewis, M., Lu, Z., Wang, H., & Wang, Y. (2014). The unconstrained binary quadratic programming problem: A survey. *Journal of Combinatorial Optimization*, 28, , 07, 2014.
- Kochenberger, G., Hao, J.-K., Lu, Z., Wang, H., & Glover, F. (2013). Solving large scale max cut problems via tabu search. *Journal of Heuristics*, 19, , 08, 2013.
- Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk*, Vol. 114 (pp. 953–956), Russian Academy of Sciences, , 1957.
- Kramers, H. A., & Wannier, G. H. (1941). Statistics of the two-dimensional ferromagnet. part i. *Phys. Rev.*, 60, , Aug, 1941, 252–262.
- Kurowski, K., Pecyna, T., Slysz, M., Rozycki, R., Waligora, G., & Weglarz, J. (2023). Application of quantum approximate optimization algorithm to job shop scheduling problem. *European Journal of Operational Research*, 310, , 03, 2023.
- Lanczos, C. (1950). An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand. B*, 45, , 1950, 255–282.
- Lang, J., Zielinski, S., & Feld, S. (2022). Strategic portfolio optimization using simulated, digital, and quantum annealing. *Applied Sciences*, 12(23), , 2022.
- Lehoucq, R., Sorensen, D., & Yang, C. (1998). *Arpack users' guide: Solution of large-scale eigenvalue problems with implicitly restarted arnoldi methods*. Software, Environments, and Tools. Society for Industrial and Applied Mathematics.

- Lucas, A. (2014). Ising formulations of many np problems. *Frontiers in Physics*, 2, , 02, 2014, 5.
- McGeoch, C., & Farré, P. (2021). *The advantage system: Performance update* (Technical report). D-Wave Systems.
- Onsager, L. (1944). Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Phys. Rev.*, 65, , Feb, 1944, 117–149.
- Palubeckis, G. (2006). Iterated tabu search for the unconstrained binary quadratic optimization problem. *Informatica*, 17(2), , apr, 2006, 279–296.
- Pelofske, E., Bärtschi, A., & Eidenbenz, S. (2023). Quantum annealing vs. qaoa: 127 qubit higher-order ising problems on nisq computers. In A. Bhatele, J. Hammond, M. Baboulin, & C. Kruse (Eds.), *High Performance Computing* (pp. 240–258), , 2023: Springer Nature Switzerland.
- Punnen, A. P. (2022). *The quadratic unconstrained binary optimization problem*. Springer International Publishing.
- Qiskit contributors (2023). Qiskit: An open-source framework for quantum computing.
- Quintero, R., & Zuluaga, L. (2022). Qubo formulations of combinatorial optimization problems for quantum computing devices.
- Rieffel, E., Venturelli, D., O’Gorman, B., do, M., Prystay, E., & Smelyanskiy, V. (2014). A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, 14, , 07, 2014.
- Rosenberg, G., Haghnegahdar, P., Goddard, P., Carr, P., Wu, J., & Lopez de Prado, M. (2016). Solving the optimal trading trajectory problem using a quantum annealer. *IEEE Journal of Selected Topics in Signal Processing*, 10, , 09, 2016, 1–1.
- Tavares, G. (2008). *New algorithms for quadratic unconstrained binary optimization (qubo) with applications in engineering and social sciences*. PhD thesis, Rutgers University.
- Tieleman, T. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. <https://cir.nii.ac.jp/crid/1370017282431050757>.
- Willsch, M., Willsch, D., Jin, F., De Raedt, H., & Michielsen, K. (2020). Benchmarking the quantum approximate optimization algorithm. *Quantum Information Processing*, 19(7), , Jun, 2020, 197.
- Yarkoni, S., Huck, A., Schuelldorf, H., Speitkamp, B., Tabrizi, M., Leib, M., Back, T., & Neukart, F. (2021). *Solving the shipment rerouting problem with quantum optimization techniques*, pp. 502–517. Springer, Cham.
- Yarkoni, S., Raponi, E., Back, T., & Schmitt, S. (2022). Quantum annealing for industry applications: Introduction and review. *Reports on Progress in Physics*, 85, , 08, 2022.

- Zen, R. A. M. (2021). *Transfer learning for neural-network quantum states*. PhD thesis, National University of Singapore. Available at <https://scholarbank.nus.edu.sg/handle/10635/224567>.
- Zen, R. A. M., My, L., Tan, R., Hébert, F., Gattobigio, M., Miniatura, C., Poletti, D., & Bressan, S. (2020). Finding quantum critical points with neural-network quantum states. *European Conference on Artificial Intelligence*, , 2020.
- Zhou, L., Wang, S.-T., Choi, S., Pichler, H., & Lukin, M. D. (2020). Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X*, 10, , Jun, 2020, 021067.

Appendix A

Reformulating the Knapsack problem as a QUBO problem

This appendix explains an example of how a combinatorial optimisation problem with inequality constraints can be formalised as a QUBO problem. The knapsack problem is a classic combinatorial optimisation problem with a knapsack with integer weight limit $W > 0$ and n items, each with a weight $w_i > 0$ and profit $c_i > 0$. The problem seeks to find the optimal set of items to place in the knapsack to maximise total profit while not exceeding the weight limit. Formally, we have

$$\max \sum_{i=1}^n c_i x_i \tag{A.1}$$

$$\text{st. } \sum_{i=1}^n w_i x_i \leq W \tag{A.2}$$

$$x_i \in \{0, 1\}^n$$

The quantity A.1 refers to the total profit of the chosen items, and constraint A.2 keeps the total weight below the weight limit. To convert this optimisation problem into a QUBO problem, we can include the inequality constraint in our QUBO formulation by introducing slack variables, y_j 's, which turn the inequality constraint into an equality

constraint (Quintero & Zuluaga, 2022).

$$\max \sum_{i=1}^n c_i x_i \quad (\text{A.3})$$

$$\text{st. } \sum_{i=1}^n w_i x_i = \sum_{j=1}^W j y_j \quad (\text{A.4})$$

$$\sum_{j=1}^W y_j = 1 \quad (\text{A.5})$$

$$x \in \{0, 1\}^n, y \in \{0, 1\}^W$$

The slack variables y_j track the total weight of the knapsack with $y_j = 1 \Leftrightarrow$ total weight is j .

Constraint A.5 ensures that only one of y_j equals 1. With penalty parameters $C_1 > \sum_{i=1}^n c_i$ and $C_2 > \sum_{i=0}^n c_i$, we can formulate the following QUBO problem:

$$\max f(x, y) := \sum_{i=1}^n c_i x_i - C_1 P_w^2 - C_2 P_n^2 \quad (\text{A.6})$$

$$P_w := \sum_{i=1}^n w_i x_i - \sum_{j=1}^W j y_j \quad (\text{A.7})$$

$$P_n := 1 - \sum_{j=1}^W y_j \quad (\text{A.8})$$

$$x \in \{0, 1\}^n, y \in \{0, 1\}^W$$

Since the penalty parameters are larger than $\sum_{i=1}^n c_i$, the optimal solution to the QUBO problem must have $P_w = P_n = 0$. Hence, it ensures that exactly one of the $y_i = 1$ and the total weight is below the knapsack capacity. Since the optimal solution to the original knapsack must also have a total weight below the knapsack capacity, the value of x that solves the QUBO must also solve the original knapsack problem. We can also formulate this QUBO problem in terms of Q by following similar steps as subsection 2.1.1.

Appendix B

Curve fitting for NNQS

This section will describe how we obtained an approximate analytical form of functions $A(s)$ and $B(s)$ as shown in Figure 4.2. The exact form is not provided in the D-wave documentation, but a set of 1000 points is available from the documentation (D-Wave Systems, 2024c). A snapshot of the dataset is shown here:

s	$A(s)$ (GHz)	$B(s)$ (GHz)
0	0.000000	9.839417×10^0
0.001	0.001001	9.746483×10^0
0.002	0.002002	9.655434×10^0
...
0.998	0.997998	1.344913×10^{-9}
0.999	0.998999	1.293784×10^{-9}
1.000	1.000000	1.242656×10^{-9}

Table B.1: Discrete points of annealing functions $A(s)$ and $B(s)$

To obtain an analytical form, we utilised the curve fit function from the `scipy` library. First, we normalise by the maximum value in $B(s)$ as only the ratio of A and B is important. We fit an exponential decay form of $a_A \cdot e^{-b_A \cdot s} + c_A$ to $A(s)$ and a quadratic

function $a_B \cdot s^2 + b_B \cdot s + c_B$ function to $B(s)$. The forms of the functions were chosen from the D-wave documentation (D-Wave Systems, 2024c). We obtain fitted constants of $a_A = 1.11, b_A = 7.06, c_A = -0.00569, a_B = 0.680, b_B = 0.288, c_B = 0.0305$. The fitted functions are shown in Figure B.1 along with the discrete values from the documentation and are found to be a good fit.

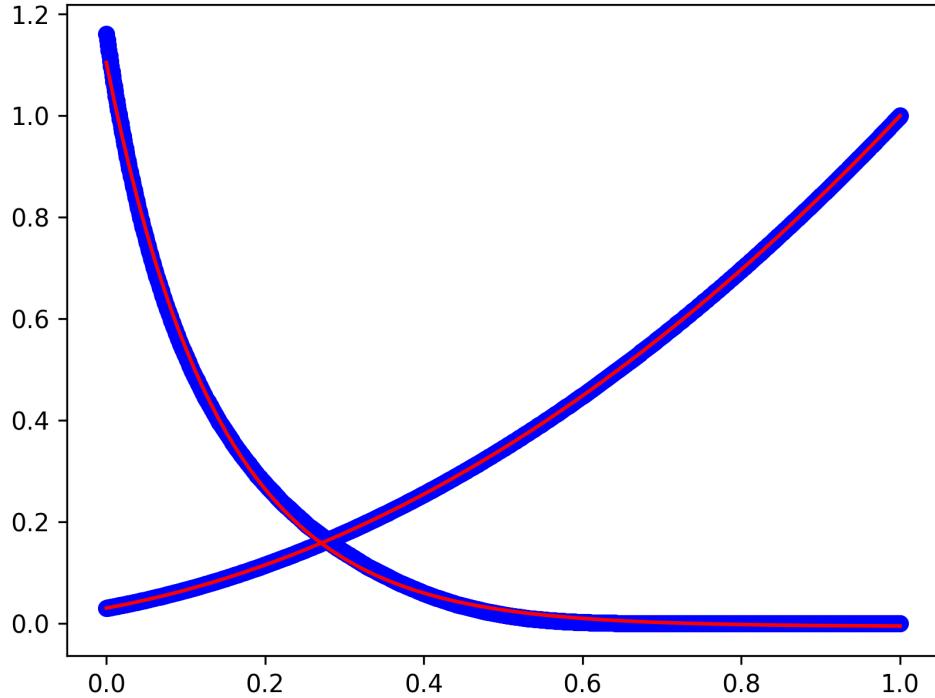


Figure B.1: Fitted $A(s)$ and $B(s)$ equations (red) with discrete values (blue) against the normalised annealing fraction s

Appendix C

NNQS exploration performance by sizes

C.1 NAE3SAT

Refer to Figure C.1.

C.2 Max-cut

Refer to Figure C.2.

C.3 SK model

Refer to Figure C.3.

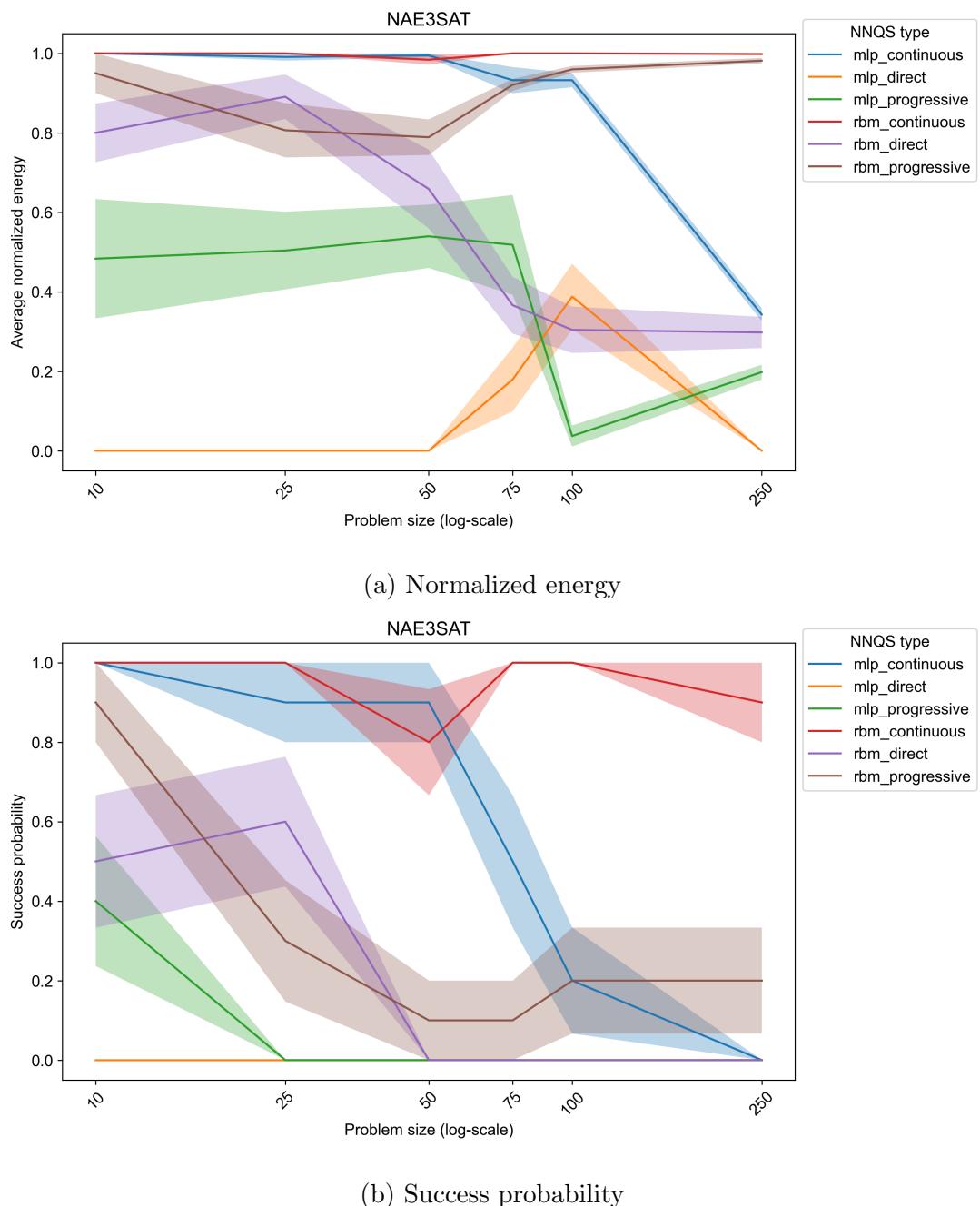
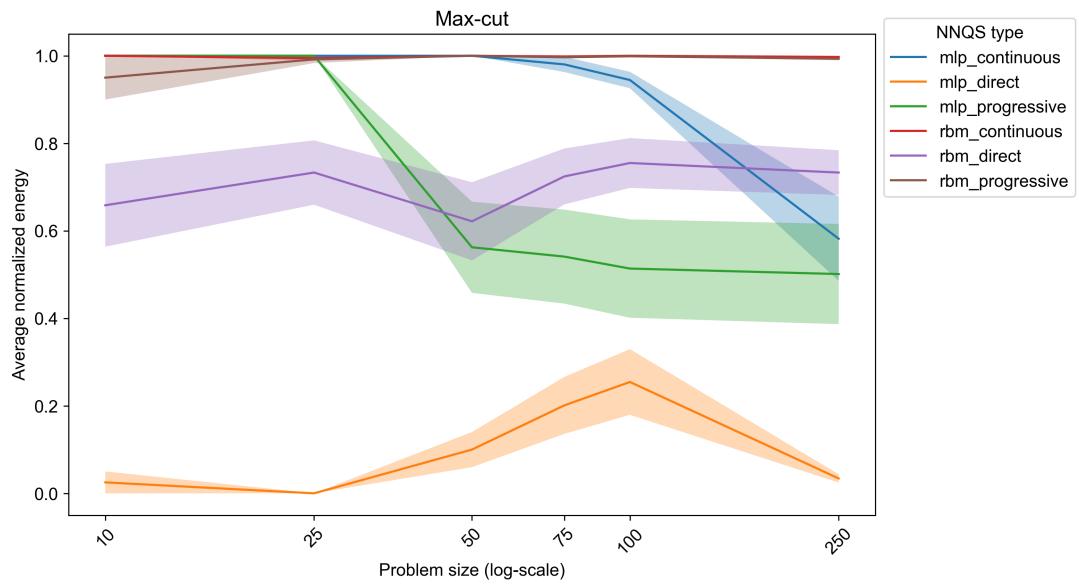
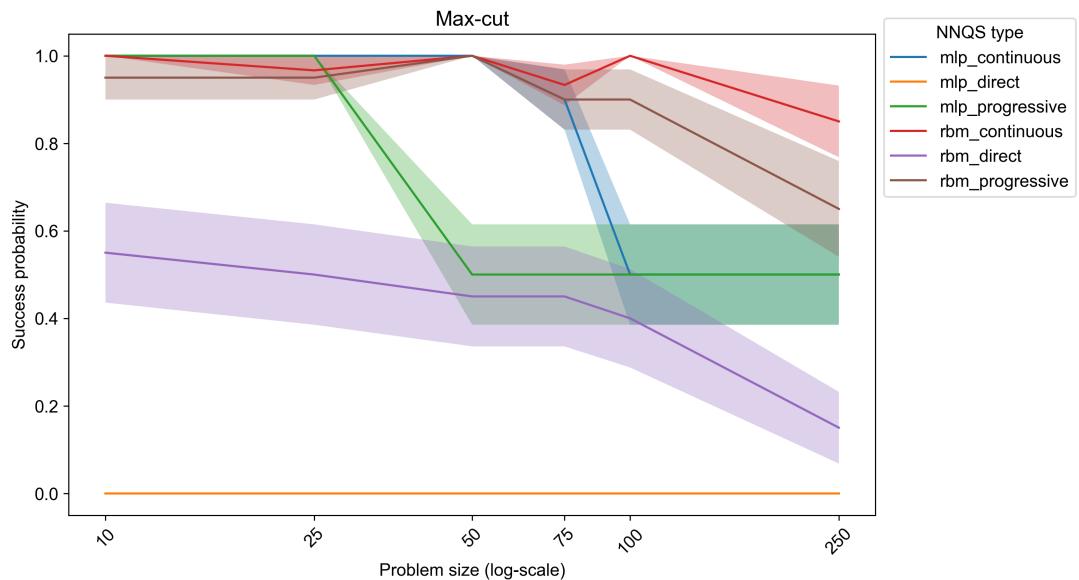


Figure C.1: Performance of different NNQS types for NAE3SAT by problem size

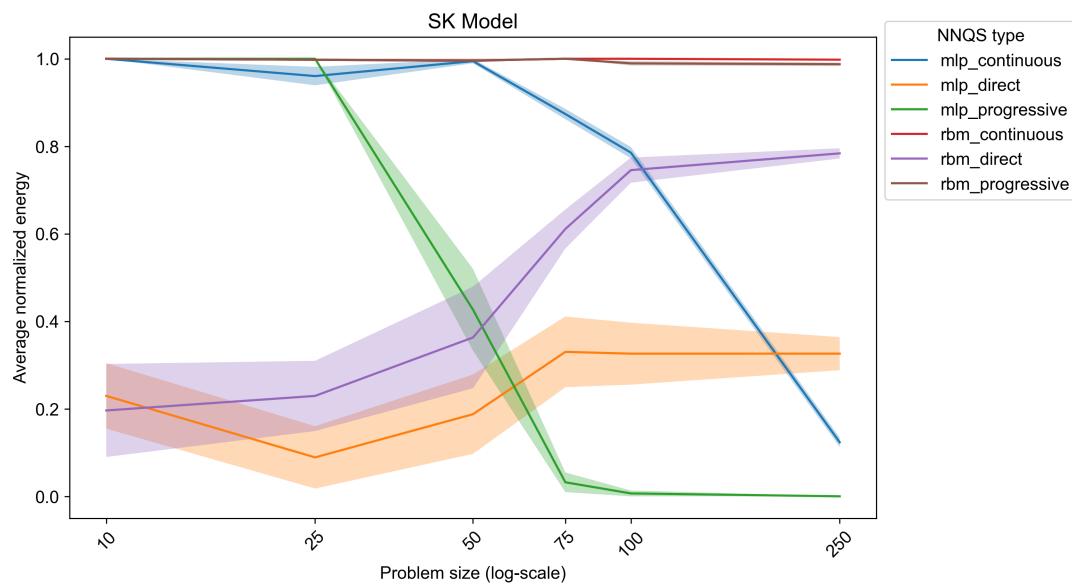


(a) Normalized energy

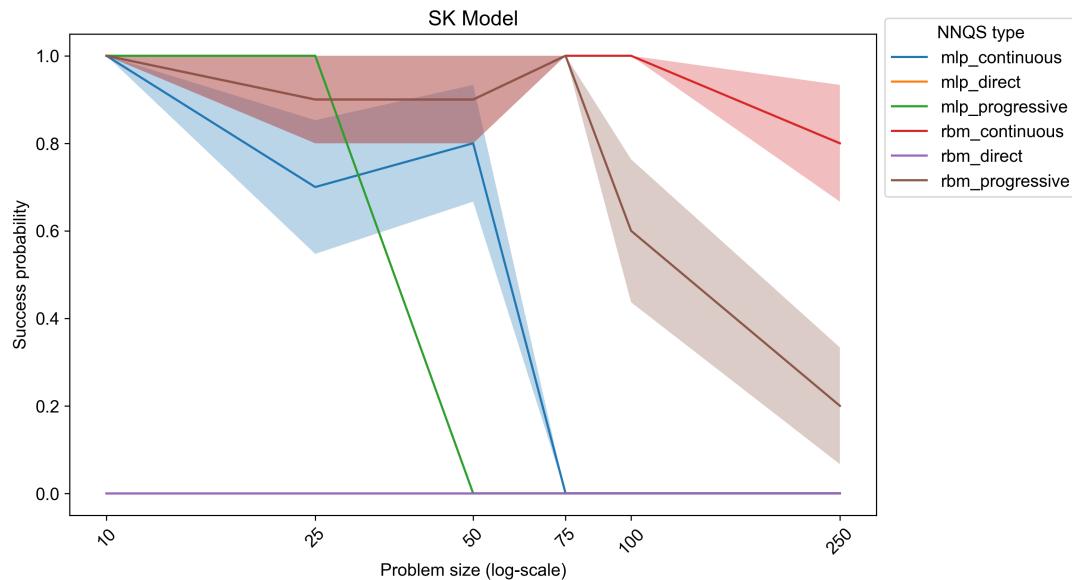


(b) Success probability

Figure C.2: Performance of different NNQS types for max-cut by problem size



(a) Normalized energy



(b) Success probability

Figure C.3: Performance of different NNQS types for SK model by problem size

Appendix D

Average runtime of solvers by problem type and sizes

D.1 NAE3SAT

Figure D.1 shows the average runtime taken for NAE3SAT problems.

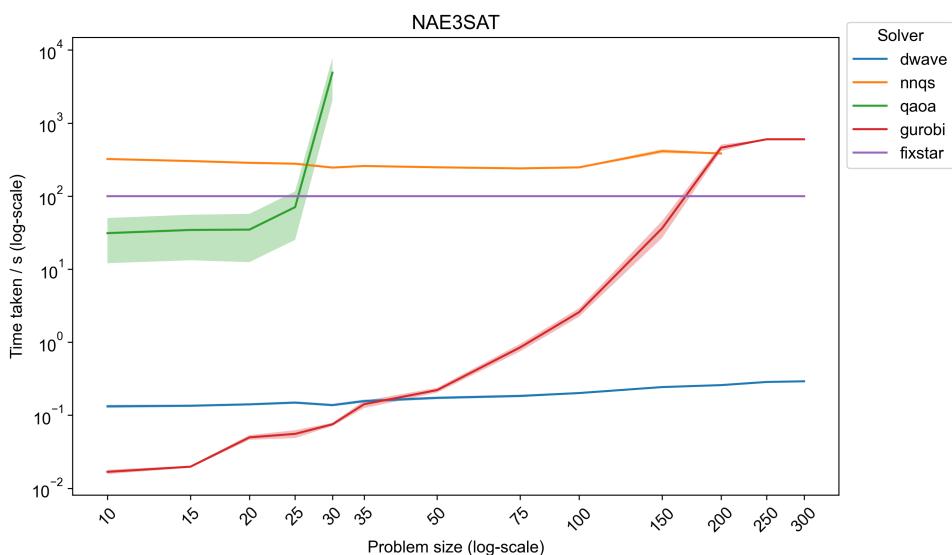


Figure D.1: Average runtime taken by different solvers for NAE3SAT by problem size

D.2 Max-cut

Figure D.2 shows the average runtime taken for max-cut problems.

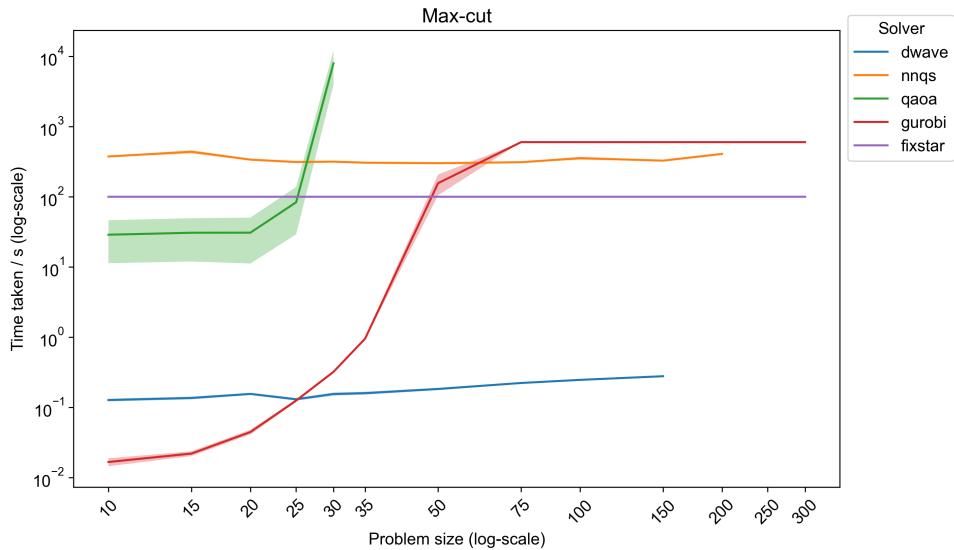


Figure D.2: Average runtime taken by different solvers for max-cut by problem size

D.3 SK model

Figure D.3 shows the average runtime taken for SK model problems.

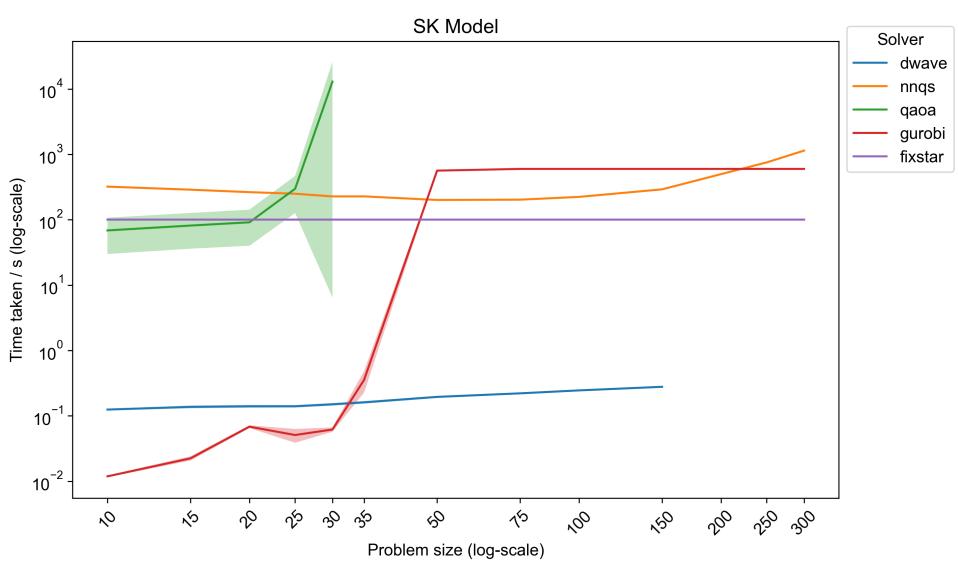


Figure D.3: Average runtime taken by different solvers for SK model by problem size

Appendix E

Metadata of solvers by problem type and sizes

E.1 D-wave

For the D-wave solver, we recorded the average number of embedded qubits, which is needed during the minor embedding phase shown in Table E.1.

n	10	15	20	25	30	35	50	75	100	150	200	250	300
NAE3SAT	15.15	27.90	42.20	58.85	80.90	102.25	184.05	379.65	622.15	1381.00	2322.65	3756.20	4493.60
Max-cut	12.70	25.95	45.45	69.90	102.60	142.50	304.05	706.05	1290.00	3087.95	-	-	-
SK model	17.05	35.80	54.50	87.65	121.15	169.80	331.60	739.45	1338.65	2995.25	-	-	-

Table E.1: Average number of embedded qubits for the D-wave solver by problem type and size

E.2 QAOA

For the D-wave solver, we recorded the average number of quantum gates and the circuit depth shown in Table E.2 and Table E.3, which help to quantify the complexity of the

quantum circuit.

n	10	15	20	25	30
NAE3SAT	77.75	127.80	181.50	235.20	292.30
Max-cut	75.00	131.00	200.00	281.00	375.00
SK model	95.00	180.00	290.00	425.00	585.00

Table E.2: Average number of quantum gates in the quantum circuit used by the QAOA solver by problem type and size

n	10	15	20	25	30
NAE3SAT	18.65	25.30	30.70	35.15	40.00
Max-cut	19.00	28.05	36.50	45.60	55.05
SK model	22.00	32.00	42.00	52.00	62.00

Table E.3: Average depth of the quantum circuit used by the QAOA solver by problem type and size

Appendix F

D-wave Quenching

A mid-anneal quench helps us identify the current state of the D-wave annealing by quickly increasing the value of s , the anneal fraction. By doing so, we change the system quickly compared to the system, which does not give the system enough time to evolve and freezes the current state. By measuring the distribution of states through repeated sampling, we can peek into the evolution of the wave function of the D-wave annealing setup and compare it to that of the NNQS.

We can quench the annealing process in a D-wave solver by specifying the anneal_schedule parameter by providing discrete points for (anneal fraction, time) (McGeoch & Farré, 2021). For example, a schedule that is $[(0.0, 0.0)(10.0, 0.5)(11.0, 1.0)]$, would mean a $10\mu s$ anneal until $s = 0.5$ then a $1\mu s$ quench until $s = 1$. For more information, refer to the D-wave documentation (D-Wave Systems, 2024a).