University of Western Australia

# A Quantum Solution to Graph Similarity Problems

*Dylan O'Donoghue*

This thesis is presented for the degree of
**Masters of Physics (Quantum Technology and Computing)**

Supervised by:
Prof. Jingbo Wang, Dr. Edric Matwiejew, Dr. Tavis Bennett
Department of Physics
School of Physics, Mathematics and Computing
The University of Western Australia

9 October 2025

# Acknowledgments

I would like to thank my supervisor, Jingbo Wang. Your support and guidance throughout the course of my studies has made this journey immensely rewarding. Thank you for your excellent management of the QUISA research group and the quantum optimisation subgroup. In taking on the role of connecting people together in meaningful pursuits of knowledge, you have created an community of collaborators which extends well beyond the usual barrier of academics and researchers into the practical world.

I also extend my thanks to Tavis, Edric, Pascal, Aidan, Andrew, and Leo for their support on the project. Tavis for sharing his valuable research, which made this whole project possible. Edric for the many insights in data analysis and QuOp. Pascal for the top-down view of the project as a quantum-HPC hybrid expert. Aiden, Andrew, and Leo, for the collaboration which guided the analysis and generation of these results.

Thank you to Benjamin Rossdeutscher. Our academic journeys have been in parallel for many years, and your personal and professional support have been a key part of not just this thesis but the path here.

This work was supported by resources provided by the Pawsey Supercomputing Research Centre with funding from the Australian Government and the Government of Western Australia.

Finally, I thank my family and friends for their continuous support throughout my degree. Without you, none of this would have been possible.

# Declaration

This is to certify that:

1. This thesis comprises of my original work.
2. Due acknowledgment has been made in the text to all other materials used.
3. This thesis consists of less than 60 pages inclusive of tables and figures but exclusive of references and appendices.

I authorise the Head of the Department of Physics to make a copy of this thesis to any person judged to have an acceptable reason for access to the information, i.e, for research, study or instruction.

Student Name: Dylan O'Donoghue    ID: 23158061

Signature

---

Date

---

Supervisor: Professor Jingbo Wang

Signature

---

Date

---

# Summary of Student Achievement

TBC.

# List of Figures

# List of Tables

# Abstract

Brief summary......

# Contents

# Chapter 1

# Introduction

Quantum computing is a rapidly evolving field that leverages the principles of quantum mechanics to perform computational tasks. By utilising the unique properties of quantum bits (qubits), such as superposition and entanglement, quantum computers can solve certain problems more efficiently than classical computers. This has led to significant interest in exploring quantum algorithms and their applications across various domains.

Network graph models have become crucial components in systems that are used in everyday life, such as search engines and social networks. Graph similarity is often important in these contexts. For example, social networks are compared to identify certain interaction patterns, traffic networks are compared to help detect abnormal changes in traffic patterns, and web networks are compared for anomaly detection. In all these cases, the similarity between two graphs with overlapping sets of nodes is assessed, and the detection of changes in the patterns of connectivity is important. For graphs that match approximately, it is useful to obtain a quantitative measure of similarity.

Graph similarity measures are quantitative calculations based on comparisons made between the structure of network graphs. Different measures of graph similarity will produce a variety of results because of differences in how the structures of the graphs are analysed. There are many measures of graph similarity, including Maximum Common Subgraph, Graph Edit Distance, Frobenius Distance, and Graph Eigendecomposition. All of these measures are successful in indicating the degree to which two graphs are similar and can also detect small differences between them.

However, many of these measures require a matching between the vertices of the two graphs, which can be computationally expensive to determine. The problem of finding an optimal vertex matching is known to be NP-hard, meaning that there is no known polynomial-time algorithm to solve it for all instances. As a result, various heuristic and approximation techniques have been developed to tackle the graph similarity problem.

A recently introduced algorithm, the non-variational QWOA was designed to solve hard combinatorial optimisation problems, generalising to non-binary and constrained problems, while simultaneously solving the challenges related to the variational approach [1]. It amplifies the probability of measuring high-quality solutions to a problem through a process of phase-separation and mixing via a continuous-time quantum walk (CTQW) on a mixing graph. In the original work, it was shown through classical simulation that non-variational QWOA found globally optimal solutions on problems such as weighted maxcut, $k$-means clustering, quadratic assignment, and the capacitated facility location problem within a small number of iterations.

We seek to determine whether the non-variational QWOA will perform equally well on the problem of calculating graph similarity as it performed on its first problems. We will do this through classical simulation of the non-variational QWOA on small instances of the graph similarity problem. We will compare its performance to that of Grover's search algorithm [2], which is a well-known quantum algorithm for unstructured search problems. Grover's algorithm provides a quadratic speedup over classical search algorithms, making it a useful benchmark for evaluating the performance of other quantum algorithms.
In this report, we will first provide a brief overview of the non-variational QWOA and Grover's search algorithm. We will then describe our methodology for simulating these algorithms on small instances of the graph similarity problem. Finally,

we will present our results and discuss their implications for the use of quantum algorithms in solving combinatorial optimisation problems.

# Chapter 2

# Literature Review

## 2.1 Quantum Computing

Quantum computing is based on the principles of quantum mechanics, which describe the behaviour of particles at the atomic and subatomic levels.

This section provides a brief overview of the fundamental concepts of quantum computing that are relevant to understanding the non-variational QWOA and its application to the graph similarity problem.

### 2.1.1 Qubits and Quantum Superposition

The fundamental unit of information in classical computing is the bit, which can exist in one of two states, typically written as 0 or 1.

In quantum computing, the basic unit of information is the quantum bit or qubit. Unlike classical bits, which can only be in one state at a time, qubits can exist in a superposition of states.

Qubits are physically realised with two-state quantum systems, such as the spin of an electron in which the two states can be taken as spin up or spin down. Qubits are mathematically described by their two-dimensional complex state vectors. Systems of qubits are described in a $2^n$ dimensional complex vector space, where $n$ is the number of qubits. A qubit's state (or the state of a collection of qubits) can be represented as the complex weighted sum of its basis states. A common choice of basis states is called the standard or computational basis, consisting of the states $|0\rangle$ and $|1\rangle$, which are the positive and negative eigenvalues of the Pauli Z operator. A general state of a single qubit can be expressed in the standard basis as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where $\alpha$ and $\beta$ are complex numbers that satisfy the normalisation condition $|\alpha|^2 + |\beta|^2 = 1$. The Born rule states that the coefficients $|\alpha|^2$ and $|\beta|^2$ represent the probabilities of measuring the qubit in the states $|0\rangle$ and $|1\rangle$, respectively.

When multiple qubits are combined, their joint state is represented by the tensor product of their individual states. For example, the state of a two-qubit system can be expressed as:

$$\begin{aligned}
|\psi_{12}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle &= (\alpha_1\,|0\rangle + \beta_1\,|1\rangle) \otimes (\alpha_2\,|0\rangle + \beta_2\,|1\rangle) \\
&= \alpha_1\alpha_2\,|0\rangle\,|0\rangle + \alpha_1\beta_2\,|0\rangle\,|1\rangle + \beta_1\alpha_2\,|1\rangle\,|0\rangle + \beta_1\beta_2\,|1\rangle\,|1\rangle \\
&= \alpha_1\alpha_2\,|00\rangle + \alpha_1\beta_2\,|01\rangle + \beta_1\alpha_2\,|10\rangle + \beta_1\beta_2\,|11\rangle
\end{aligned}$$

where $|00\rangle, |01\rangle, |10\rangle$, and $|11\rangle$ are the basis states of the two-qubit system.

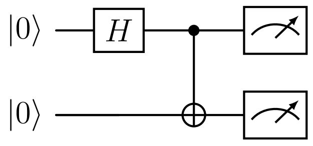
This allows a quantum computer with $n$ qubits to represent $2^n$ states simultaneously, which is a key feature that enables quantum parallelism, allowing quantum computers to perform certain computations more efficiently than classical computers.

Despite this parallelism, the computational power is limited by the fact that measuring a quantum state collapses it to one of its basis states, yielding only a single outcome. Therefore, quantum algorithms must be carefully designed to manipulate the amplitudes of the quantum states such that the desired outcomes have higher probabilities of being measured.

### 2.1.2 Quantum Gates, Entanglement

Quantum gates are the building blocks of quantum circuits, analogous to classical logic gates. They manipulate the states of qubits through unitary transformations, which are reversible operations that preserve the total probability of the quantum system. Common quantum gates include the Hadamard gate, which creates superposition, and the CNOT gate, which entangles two qubits.

We can visually represent a series of quantum gates acting on qubits using a quantum circuit diagram. For example, the following diagram shows a simple quantum circuit with two qubits and two gates:



This diagram represents a circuit where the first qubit is put into superposition by the Hadamard gate (H), and then a CNOT gate acts on the second qubit depending on whether the first qubit is in a $|1\rangle$ state or the $|0\rangle$ state. Finally, both qubits are measured.

The quantum state begins as:

$$|\psi\rangle = |00\rangle$$

Then after the Hadamard gate is applied, the quantum state is:

$$|\psi_H\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

Finally the CNOT gate is applied and the quantum state is:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

The measurements can be performed relative to any basis in the qubit's vector space. Commonly, measurements are performed in the same basis as that is used to describe the qubits.

This state is an example of entanglement, a unique quantum phenomenon where the states of two or more qubits become correlated such that the state of one qubit cannot be described independently of the state of the other qubit(s). For example, in the entangled state above, if one were to measure the first qubit to be in the state $|0\rangle$, that necessarily requires the second qubit to also be in the state $|0\rangle$.

The measurement of one of the entangled qubits has collapsed the superposition for both qubits. This surprising result was initially deemed unphysical on the grounds that it violates local realism by Einstein, Poldowsky, and Rosen, the first to describe this behaviour[3]. Nevertheless, quantum entanglement and its implications have been repeatedly verified through experimentally violating Bell's inequality, establishing that the correlations produced from quantum entanglement cannot be explained by local hidden variables.

citation here for Bell's inequality and experiments violating it

These correlations are fundamental to quantum algorithms. Without entanglement, quantum computers could be efficiently simulated by classical computers, negating any advantage.

### 2.1.3 Quantum Algorithms and Challenges

Quantum algorithms seek to leverage properties of qubits and quantum gates to solve specific problems more efficiently than classical algorithms. Notable examples include Shor's algorithm for integer factorization[4], which runs exponentially faster than the best-known classical algorithms, and Grover's algorithm for unstructured search[2], which provides a quadratic speedup over classical search algorithms.

Large, universal, fault-tolerant quantum computers that can run these algorithms at useful scales will require large numbers of qubits. While researchers have made significant progress in error correction, coherence times, qubit connectivity, and gate fidelities, fault tolerant quantum computing may take decades to achieve. In the meantime, Noisy Intermediate-Scale Quantum (NISQ) computers already exist, and can run small-scale quantum algorithms that can provide insights into quantum behavior and inform the development of larger systems.

These NISQ devices are limited by the number of qubits, gate fidelities, and coherence times. As a result, quantum algorithms designed for NISQ devices must be efficient in terms of qubit usage and circuit depth to minimize the impact of noise and errors.

Even if NISQ devices cannot solve problems faster than the best classical computers using the best classical algorithms (particularly with the emergence of exascale supercomputing), they could still provide an economic advantage by using less energy or by being lower cost to build and operate.

NISQ algorithms are designed to work within these low-resource constraints. These algorithms typically involve a hybrid approach, where a quantum computer is used to prepare and measure quantum states, while a classical computer optimises parameters based on the measurement results. Some examples of NISQ-era algorithms are described in the following sections.

## 2.2 Combinatorial Optimisation Problems

Combinatorial optimisation problems consist of a discrete set of feasible solutions, which can be represented as a set $S$, with finite order $N = |S|$. Each of the $N$ solutions $\mathbf{x} \in S$ can be represented as a vector of $n$ variables $\mathbf{x} = (x_1, x_2, ..., x_n)$, where $x_j \in \{0, 1, ..., k-1\}$ are called the *decision variables*. The solution space $S$ may contain all possible solutions of this form (of which there are $N = k^n$) or $S$ may be restricted to solutions which satisfy some constraint (such as in the case of permuation-based problems where $k = n$ and $N = n!$). Each solution has an associated objective value (or cost), given by a objective function (or cost function) $f : S \to \mathbb{R}$, which gives a measure of the quality of any solution. The goal of the optimisation problem is to find the *optimal solution* $\mathbf{x}^*$ that maximises (or minimises) the objective (or cost) function:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in S} f(\mathbf{x}) \text{ or } \mathbf{x}^* = \arg\max_{\mathbf{x}\in S} f(\mathbf{x})$$

The structure of the objective function is determined by the problem definition. The problem definition consists of a series of criteria which are defined on a subset of the variables in $\mathbf{x}$, and the objective function conveys the extent to which a solution satisfies the criteria.

Combinatorial optimisation problems are often NP-hard, meaning that no known polynomial-time algorithm can exactly solve all instances of the problem. Examples of NP-hard combinatorial optimisation problems include the travelling salesman problem and the knapsack problem.

The main challenge in solving these problems is that they often require a near-exhaustive search of the solution space, which usually grows exponentially or super-exponentially with problem size. An example of this super-exponential growth will be present in the graph similarity problem in section 2.3.

Research in combinatorial optimisation algorithms showing exponential speedup includes:

- Exact algorithms that find optimal solutions for special problem instances with certain properties.

Citations Here

- Approximate algorithms that find solutions which are near the optimal solutions for general problem instances.

Approximate algorithms are the focus of this thesis.

expand on this

To quantify how close a solution $\mathbf{x}_i$ is to the optimal solution, the approximation ratio $A$ is used:

$$A_i = \frac{f(\mathbf{x}_i)}{f(\mathbf{x}^*)}$$

This value compares the objective value of a solution against the maximum possible objective value. A successful approximate combinatorial optimisation algorithm should find solutions which give approximation ratios close to 1.

## 2.3 Graph Similarity Problem

The graph similarity problem involves determining how similar two graphs are to each other. This can be useful in various applications, such as social network analysis, bioinformatics, and pattern recognition.

There is a need to distinguish between the graph similarity problem and the graph isomorphism problem.

A pair of graphs $G_1$ and $G_2$ are considered isomorphic ($G_1 \cong G_2$) if and only if there exists a bijection between their vertex sets such that any two vertices in $G_1$ are adjacent if and only if their corresponding vertices in $G_2$ are also adjacent. In this definition, the graphs are understood to be undirected, non-labelled, non-weighted graphs. However, isomorphism may apply to all other notions of graph by adding requirements to preserve any additional structures.

The graph isomorphism problem is then the task of correctly creating that bijection (or equivalently, proving that one exists). Algorithms that solve this problem are trivially verifiable, as checking the correctness of the solution only involves checking that the vertex bijection correctly preserves the edges. Since the graph isomorphism problem can be posed as a 'yes-no' question on a set of inputs values, it is called a decision problem.

In contrast, the graph similarity problem is the problem of calculating how close a pair of graphs are to being isomorphic. This cannot be posed as a 'yes-no' problem, as many magnitudes of similarity must meaningfully captured. There are multiple measures of graph similarity. Some examples include:

- *Graph Edit Distance (GED):* Counts how many 'edits' are required to transform one graph into another, where 'edits' include inserting/deleting a vertex or edge, or substiting one vertex or edge with another. The lower the distance, the more similar the two graphs are.

citations for each of these

- *Maximum Common Subgraph (MCS):* Finds the largest induced subgraph that is common to the two given graphs. The larger the subgraph, the more similar the two graphs are.
- *Frobenius Distance:* Calculates the distance between two graphs using the difference between their adjacency matrices. The lower the distance, the more similar the two graphs are.
- *Edge Overlap:* Finds the proportion of matching entries in the adjacency matrices of the graphs. The greater the proportion, the more similar the two

graphs are.[5]

The choice of measure often depends on the specific application and the properties of the graphs being compared. Considerations such as cost, time, and interpretability inform which measure is best for a practical problem.

Koutra et al. [6] formalised a set of axioms that graph similarity measures should conform to:

probably bring up mapping the outputs to [0,1]

1. Identity: $sim(G_1, G_2) = 1 \iff G_1 \cong G_2$
2. Symmetric: $sim(G_1, G_2) = sim(G_2, G_1)$
3. Zero: For a clique graph $G_1$ and a empty graph $G_2$, $\lim_{n\to\infty} sim(G_1, G_2) = 0$

The measures listed above can all be minimally adapted to satisfy these axioms.

Most graph similarity measures assume a given mapping between the vertices of the graphs. Unlike in graph isomorphism, this vertex mapping doesn't need to be a bijection, since there exists a notion of similarity between graphs with a different number of vertices. For graph similarity methods such as those listed previously, the choice of vertex mapping can change the final result. However, Koutra's axiom 1 requires that graph similarity measures can identify isomorphic graphs by giving them a similarity of 1.

So to satisfy Koutra's axioms, we must find a mapping between the graphs that maximises their similarity.

To find the best mapping between labelled graphs, one can simply 'pair up' the vertices in the graphs so that each pair consists of one vertex from each graph that share the same label. This significantly reduces the number of possible mappings (often to just 1), so finding the mapping with the maximum similarity is simple.

To find the best mapping between unlabelled graphs is very challenging. In the general case, the similarity associated with every possible permutation of mappings must be checked until either a mapping that shows isomorphism (or another verifiable global maximum) is found, or the entire space of mappings has been searched.

As such, the unlabelled graph similarity problem can be defined in terms of a combinatorial optimisation problem.

### 2.3.1 Problem Definition

A formal definition for the unlabelled graph similarity problem is as follows:

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two finite, simple graphs and considered *unlabelled* in the sense that vertex identifiers carry no intrinsic meaning.

Let $\Pi(V_1, V_2)$ denote the set of all maps $\pi : V_1 \to V_2$.

Given a base similarity function $f : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ that compares labelled graphs from the class $\mathcal{G}$ of all finite simple graphs, the *unlabelled similarity* between $G_1$ and $G_2$ is defined as

$$\mathrm{Sim}(G_1, G_2) = \max_{\pi \in \Pi(V_1, V_2)} f(G_1, \pi(G_2))$$

where $\pi(G_2)$ denotes the relabelling of $G_2$ induced by $\pi$.

The unlabelled graph similarity problem is the problem of computing $\mathrm{Sim}(G_1, G_2)$.

This problem is known to be NP-hard, meaning that there is no known polynomial-time algorithm to solve it for all instances. As a result, various heuristic and approximation techniques have been developed to tackle the graph similarity problem.

## 2.4 Variational Quantum Algorithms

### 2.4.1 VQAs in General

Variational Quantum Algorithms (VQAs) are a class of hybrid quantum-classical algorithms that leverage the strengths of both quantum and classical computing to solve optimisation problems. They are particularly well-suited for NISQ devices due to their relatively low resource requirements and robustness to noise.

VQAs implement a parametrised quantum circuit (PQC) whose unitary evolution is governed by a set of $d$ classically tunable parameters $\theta \in \mathbb{R}^d$. The quantum device prepares the state

$$|\psi(\theta)\rangle = U(\theta)\, |0\rangle^{\otimes n}$$

where $U(\theta)$ is a depth-restricted circuit composed of parameterised single- and multi-qubit gates, and $n$ is the number of qubits in the input to the quantum computer. Commonly, $d = \mathcal{O}(\text{poly}(n))$, since we seek efficient circuits that are polynomial in resources.

citation here.

The objective function of the algorithm is typically expressed as the expectation value of a Hermitian operator $H$ with repect to the variational state:

$$C(\theta) = \langle\psi(\theta)|\, H\, |\psi(\theta)\rangle$$

This cost function is estimated on the quantum device via repeated projective measurements in appropriate bases. A classical optimiser updates $\theta$ with the aim of minimising (or maximising) $C(\theta)$, thus driving the quantum circuit towards a state that approximates the optimal configuration with repect to the target problem. The algorithm repeatedly prepares this variational quantum state, measures its expectation value, and optimises the parameters until a certain number of iterations is complete or until measurements of the quantum state have an approximation ratio sufficiently close to 1.

To summarise, VQAs consist of a two-step loop: a quantum computer evolves the initial state to a final state via a PQC, then a classical computer measures the expectation value of the final state and uses this data to update the PQC. At the end of this loop, the quantum computer's output is now a good approximation of the optimal configuration.

### 2.4.2 QAOA

finish this section

The Quantum Approximate Optimisation Algorithm (QAOA) is a subclass of VQAs designed to solve combinatorial optimsiation problems, introduced by Farhi et al. [7] In this formulation, the PQC is defined by alternating applications of unitaries. One is a quality mixer associated with the problem instance and the other is a mixing operation that induces interference.

The quantum state begins in the equal superposition state over the computational basis states:

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

where $n$ is the number of qubits.

Define a unitary operator $U(\gamma)$ which depends on: an angle $\gamma$, which lies between 0 and $2\pi$; and a quality operator $Q$, which is a diagonal operator that contains the objective function value for each solution.

$$U(\gamma) = e^{i\gamma Q}$$

Define operator $U(\beta)$ which depends on the parameter $\beta$, which lies between 0 and $\pi$.

$$U(\beta) = \Pi_{j=1}^{n} e^{-i\beta\sigma_j^x}$$

Here, $\sigma_j^x$ denotes the $j$th Pauli X operator.

that doesn't seem right

Now, for any integer $p \geq 1$, and $2p$ angles $\gamma = (\gamma_1, ..., \gamma_p)$ and $\beta = (\beta_1, ..., \beta_p)$ we define the final quantum state:

$$|\gamma, \beta\rangle = U(\beta_p)U(\gamma_p)\ldots U(\beta_1)U(\gamma_1)\,|s\rangle$$

We can now do the usual VQA method of measuring the expectation value, optimising the parameters $\gamma$ and $\beta$, and repeating to obtain good approximations of the optimal solution.

### 2.4.3 Challenges

Although VQAs are a promising platform for NISQ computing, their effective performance relies on the computationally expensive tuning of an often prohibitively large number of variational parameters. Additionally, the performance of these variational algorithms is highly dependent on the initialisation of the variational parameters.

rewrite in own words

Therefore, it can be argued that VQAs replace one difficult optimisation problem with another difficult optimisation problem.

## 2.5 Non-Variational Quantum Walk-based Optimisation Algorithm

### 2.5.1 Overview

The NVQWOA is a novel algorithm introduced by Bennett et al. that builds upon and addresses the variational challenges of the QAOA[1, 8]. Like the QAOA, it alternates between applying two unitaries: a phase-separation unitary that encodes the objective function into the complex phase of the states and a mixing unitary that induces quantum interference between the solutions. Unlike QAOA or other VQAs, the parameters defining the unitaries are not variationally tuned by a classical optimiser. Instead, they are predetermined by the choice of only three hyperparameters $\gamma, t, \beta$.

Simulations have demonstrated that this non-variational approach yields rapid convergence to global optima across diverse problem types, including MaxCut, maximum independent set, k-means clustering, capacitated facility location, and the quadratic assignment problem, using only modest circuit depths and a handful of iterations [8].

### 2.5.2 Algorithm Description

Using the terms and definitions of combinatorial optimisation problems from Section 2.2:

It begins with an initial superposition over the possible solution states:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{\mathbf{x} \in S} |\mathbf{x}\rangle$$

Each iteration of the NVQWOA applies two unitaries:

$$U_Q(\gamma) = e^{-i\gamma Q}, \quad U_M(t) = e^{-itA}$$

where $Q$ is a diagonal operator such that $Q|\mathbf{x}\rangle = f(\mathbf{x})|\mathbf{x}\rangle$ encodes the objective function values, and $A$ is the adjacency matrix of a mixing graph whose vertices correspond with feasible solutions.

The phase-separation unitary applies a phase proportional to the objective function $f(\mathbf{x})$, while the mixing unitary $U_M(t)$ implements a continuous time quantum walk (CTQW) on the mixing graph, redistributing complex amplitudes across neighbouring solutions. Repeated alternation of these unitaries produces a designed interference process in which amplitudes associated with optimal or near-optimal solutions interfere constructively, amplifying their measurement probabilities.

The final quantum state after $p$ iterations of each unitary is

$$|\gamma, t, \beta\rangle = \left[ \prod_{i=0}^{p-1} U_M(t_i) U_Q\left( \frac{\pm\gamma_i}{\sigma} \right) \right] |s\rangle$$

where the $\pm$ accounts for whether the goal is minimisation $(-)$ or maximisation $(+)$, $\sigma$ is the standard deviation of the objective function values (which can be efficiently approximated through random sampling), $t_i$ and $\gamma_i$ are obtained from specific formulae in terms of $\gamma, t$, and $\beta$ and where $\gamma > 0, t > 0$, and $0 < \beta < 1$:

$$\gamma_i = \left( \beta + (1-\beta)\frac{i}{p-1} \right) \gamma, \quad t_i = \left( 1 - (1-\beta)\frac{i}{p-1} \right) t$$

So the angles $\gamma_i$ increase linearly with $i$ and the mixing times $t_i$ decrease linearly with $i$.

### 2.5.3 Mixing Unitary

A key conceptual advance of the NVQWOA lies in the explicit construction of a problem-structured mixing graph (also referred to as mixing unitary or mixer). The vertices of the graph represent feasible solutions, and edges connect nearest neighbour configurations, typically those separated by a minimal Hamming distance or by a single elementary move. The resulting adjacency matrix $A$ defines a sparse, vertex-transitive graph whose degree $d$ is polynomial in problem size.

#### 2.5.3.1 Mixer Examples

In his work, Bennett provides some examples of mixing graphs with known efficient implementations on a quantum computer [1, 8]:

*Binary Mixer*: For problems with $n$ binary decision variables $x_j \in \{0, 1\}$, the mixing graph is a $n$-dimensional hypercube, which is identical to the transverse-field Hamiltonian from the QAOA. Solutions are connected to the $n$ other solutions which differ by one bit-flip.

*Integer Mixer*: For problems with $n$ integer decision variables $x_j \in \{0, 1, ...k - 1\}$. Solutions are connected to the $n(k - 1)$ other solutions which differ in the choice of one decision variable.

*Permutation Mixer*: For problems with $n$ integer decision variables $x_j \in \{0, 1, ...k - 1\}$ where the solutions are constrained to not have repeated decision variables. Solutions are connected to the $\frac{1}{2}n(n - 1)$ other solutions which are transpositions of each other.

The choice of mixing graph directly determines how probability amplitudes are propagated, making it possible to exploit problem structure, unlike Grover's diffusion operator or the complete-graph mixer, which treat all states as equally connected. citations

#### 2.5.3.2 Mixer Subshells

An important part of the structure of mixing graphs are the stabiliser orbits, or subshells.

The distance $d(u, v)$ between two vertices $u, v$ on a graph is the minimum length of the paths connecting them. A subshell is a subset of the vertices of the mixing graph, defined in relation to a reference vertex. The subshell $h_k$ contains all vertices that are a distance $k$ away from the reference vertex. The set of all subshells for a graph form equivalence classes on the set of the vertices of the graph.

# Chapter 3

# Methods

## 3.1 Overview

We classically simulated the application of the NVQWOA to the graph similarity problem. The measure of graph similarity chosen was Edge Overlap, as it is a low-cost measure that still serves as a interpretable measure of similarity. The aim was to maximise this objective function through matrix exponentiation and multiplication,analytically computing the manipulation of state vector such that the states which corresponded with the optimal solutions (those with the greatest similarity) had an amplified probability of being measured.

## 3.2 Classical Simulation

To investigate the performance of the NVQWOA on graph similarity problems, we chose to perform classical simulations of a quantum computer, rather than directly performing the algorithm on a quantum computer. Classical simulation allows for computation with negligible stochastic or decoherence errors and complete analysis of the evolving quantum state, something which is inaccessible on real quantum hardware but useful for characterising performance and benchmarking.

Classical simulation of quantum algorithms and computers can be approached in many different ways. Tensor networks simplify circuits into lower-entanglement systems but has exponentially degrading performance as circuit depth and entanglement entropy. Stabiliser-based approaches can efficiently simulate Clifford-only circuits and certain restricted non-Clifford extensions but cannot capture the arbitrary parameterised unitaries used in the NVQWOA. State vector simulation computes the full evolution of the state vector, but can be costly in computational resources.

citations

State vector simulation was chosen as the strategy for its accuracy. The Qiskit Aer simulator and related packages were not suitable as the permutation mixer required is a sparse operator rather than a tensor-product of standard gates. Instead, we directly computed the state vector using QuOp, a Python framework for parallel simulation of quantum variational algorithms[9, 10]. QuOp provides efficient handling of large, sparse unitaries and supports MPI-based parallelisation, allowing simulation of high-dimensional state vectors across multiple compute nodes without loss of precision.

Classical simulations of the non-variational QWOA would not be possible without high-performance computing (HPC) resources, which were provided by the Pawsey Supercomputing Centre.

## 3.3 QuOp

QuOp is a Python 3 module designed for parallel, distributed-memory simulation of QVAs with arbitrary phase-shift and mixing operators.

To apply it to the NVQWOA, a parameter mapping was used to translate the hyperparameters of the NVQWOA to the individual parameters of a QVA. The quantum circuit was defined to be the two unitaries of the NVQWOA, the phase separator $U_Q$ and the mixer $U_M$. With these modifications, QuOp could simulate the NVQWOA.

## 3.4 Graph Similarity Measure

As described in section 2.3, there are many options for the choice of which graph similarity measure to use.

Graph edit distance is the most common choice, but is costly to compute, as it requires the equivalent of a pathfinding search, such as an $A^*$ search algorithm. cite Since computational resources were a significant pressure, we chose one of the least computationally expensive similarity measures to compute, the edge overlap.

To calculate the edge overlap, the number of matching entries in the adjacency matrices of the graphs are counted and the sum is divided by $n^2$, which corresponds to the size of the adjacency matrices and the maximum possible overlap. This easy calculation also makes the edge overlap highly interpretable. One can imagine the vertex mapping as translating the vertices of one graph to the corresponding vertices in the other graph, and the edge overlap represents how much the edges line up correctly. The edge overlap also works on directed graphs, but directed graphs were not investigated in this thesis.

When the best vertex mapping is known between the two graphs, the edge overlap satisfies Koutra's similarity measure axioms, making it a 'good' measure of graph similarity.

A drawback of using edge overlap is that it is not well-defined for graphs with differing numbers of nodes, $|V_1| \neq |V_2|$. This restricts our investigation to graphs with an equal number of nodes.

The choice of graph similarity measure is not expected to have a strong effect on the observed results. The vertex mappings that give strong similarity with any particular measure usually give strong similarity with most measures. For the choice of a specific graph similarity measure to affect the results, it would have to give a different problem structure which significantly alters the structure of qualities.

## 3.5 Problem Instance Construction

There does not exist a widely used library of graph similarity problem instances. To test the algorithm on a wide group of fairly-chosen problems, we randomly generated the graphs using the Erdős-Renyi random graph model. The number of nodes $n$, was varied to investigate how changing problem sizes changed the performance of the algorithm. The probability of edge inclusion was fixed at 0.5 for simplicity and repeatability.

cite erdos renyi

Differences in graphs were induced by copying a graph and removing $k$ edges. This allowed the upper limit of similarity to be easily known to be $\frac{n^2-2k}{n^2}$.

## 3.6 Phase Separator and Mixer Construction

The phase separating unitary was constructed in QuOp. For each vertex mapping, the quality (given as edge overlap) was calculated and stored as an array. This array was converted into a diagonal unitary operator in QuOp.

The mixer was constructed as a column oriented array in Numpy and converted to a compressed sparse row format for storage. This sparse matrix could then be loaded into QuOp using HDF5 parallel I/O operations.

cite HDF5?

Since we are considering graphs with an equal number of nodes $n$, vertex mappings match the arbitrary vertex indices of each graph to one another with some ordering, $\pi : \{1, ..., n\} \rightarrow \{1, ..., n\}$. (In the unlabelled graph problem, the vertices in each graph can still have indices, just not information that can inform how to match those indices together.) Each $\pi \in \Pi$ corresponds to a distinct reordering of the vertex indices. Therefore, vertex mappings can be equivalently written as permutations of $n$ distinct decision variables.

From this, we can conclude that the correct choice of mixer is the permutation mixer, which connects solutions that are one swap of decision variables away from one another. In terms of vertex mappings, the permutation mixer connects solutions corresponding to mappings which differ only by two vertices that have had their targets swapped. Since these are the nearest neighbours to one another, we can also conclude that the minimum hamming distance from one solution to another is 2.

## 3.7 Hyperparameter Optimisation

The NVQWOA has three hyperparameters: $\gamma, t, \beta$. Although sub-optimal parameters can produce quantum circuits that significantly amplify optimal and near-optimal solutions, it is expected that finding optimal values for these hyperparameters through three-dimensional numerical optimisation will give the best possible results.[8]

We used three different objective functions as inputs to the optimiser:

1. *Expected Value*: The expected value of the state vector after amplification. This is the most practical to measure on real quantum hardware.
2. *Optimal Solution Probability (OSP)*: The probability of measuring an optimal solution. This is impractical to measure on quantum hardware, as it requires foreknowledge of the optimal solution and an unfeasible number of shots. It is included in our investigation here to provide an upper bound by finding the best possible parameters.
3. *Conditional Value at Risk (CVaR)*: The expected value of the $\alpha$-tail of the state vector. This takes more shots to measure on quantum hardware than the expected value, but has been shown to give faster convergence to better solutions.[11]

Optimisation of the NVQWOA hyperparameters was done in QuOp during the NVQWOA simulation, using the Nelder-Mead method of numerical optimisation [12]. This method was chosen after experimentation with various numerical optimisation methods showed that Nelder-Mead converged the most quickly to the optimal parameters.

## 3.8 Procedure

### 3.8.1 Amplification Analysis

We started the analysis of the NVQWOA by considering one pair of graphs at a time, and verifying that amplification is correctly applied to the optimal and near-optimal solutions.

For each of $n = 8, 9, 10$, the probability distribution for the different solutions was compared before and after the algorithm was applied. We expected that solutions with high similarity would be more likely to be measured after the algorithm was applied. Additionally, the probability to measure the optimal solution was calculated.

We later expanded this analysis to problem sets with 30 distinct problem instances.

### 3.8.2 Solution Distance Analysis

To supplement the verification of amplification, it is useful to obtain statistics on the distance between solutions. This is because we expect that near-optimal solutions, which will have a low distance to the optimal solutions, should be amplified more than other solutions, which have a greater distance to the optimal solutions.

There are two useful notions of distance between solutions: subshell distance and Hamming distance, which are defined and explained below.

#### 3.8.2.1 Subshell Distance

In this thesis, we use the term subshell distance to refer to the minimum distance on the mixing graph between a solution and an optimal solution. The term subshell in this notion of distance comes from the description in section 2.5.3.2, and is used to distinguish this distance measure from other approaches. The mixer we are using is the permutation mixer, so the subshell distance is defined as:

$$\mathrm{SD}(\mathbf{x}^1, \mathbf{x}^2) = \min\{m \in \mathbb{N}_0 : \exists \tau_1, ..., \tau_m \in \mathcal{T} \text{ with } \tau_m ... \tau_1 \mathbf{x}^2 = \mathbf{x}^1\}$$

where $\mathcal{T} = \{(ij) : 1 \leq i \leq j \leq n\}$ is the set of all transpositions in the symmetric group $S_n$ and transpositions act by reindexing coordinates. For this analysis $\mathbf{x}^1$ is always an optimal solution. If there were multiple optimal solutions, the subshell distance from each optimal solution was found and the minimum value was used.

Subshell distance is a useful metric in the analysis because it directly represents how close the solutions are to each other on the mixing graph. By analysing how amplification is applied to solutions depending on their subshell distance, we can observe how probability is being moved by the mixing process. Additionally, if one were to perform a local search after finding a near-optimal solution, it would likely be a search of solutions with low subshell distance to the found solution, to limit the search space.

#### 3.8.2.2 Hamming Distance

Hamming distance is a simple yet powerful and highly interpretable measure for comparing objects. It measures the number of different decision variables for a solution. The Hamming distance for two solutions is defined as

$$\mathrm{HD}(\mathbf{x}^1, \mathbf{x}^2) = \sum_{i=0}^{n} \delta(x_i^1, x_i^2)$$

where $\delta$ is 1 if $x_i^1 \neq x_i^2$ and 0 otherwise. For this analysis $\mathbf{x}^1$ is always an optimal solution. If there were multiple optimal solutions, the Hamming distance from each optimal solution was found and the minimum value was used.

Hamming distance is a useful metric in the analysis because it gives a measure of distance that is not related to the structure of the algorithm and can be easily compared to other combinatorial optimisation problems and algorithms.

### 3.8.3 Mixer Analysis

Using the notion of subshell distance, we can characterise the suitability of this mixer to the problem of graph similarity.

### 3.8.4 Hyperparameter Analysis

The hyperparameters were selected by the optimisation procedure outlined in section 3.7, beginning with the initial parameters $\gamma = 1, t = 0.2, \beta = 0.8$. These were chosen as the initial parameters by rounding an initial run of numerical optimised parameters to one significant figure.

We stored the hyperparameters obtained for each problem instance with each optimisation objective, producing a set of nine hyperparameters for each problem instance.

# Chapter 4

# Results

Trials were successfully conducted for the NVQWOA on graph similarity problems of size $n = 8, 9, 10$. Each set of graph sizes had 30 distinct problem instances, and each problem instance was simulated with $p = 10$ iterations for hyperparameter optimisation of the expected value, the optimal solution probability (OSP), and the conditional value at risk (CVaR). The results of these trials are presented in this chapter.

## 4.1 Amplification Analysis

Figure 4.1 is a grid of plots showing the probability to measure solutions according to their similarity scores. Figure 4.2 depicts the same values but with a log-scale y axis.

Figure 4.3 is a boxplot showing the probability of measuring the optimal solution (OSP) after amplification.

## 4.2 Hamming Distance Analysis

Figure 4.4 shows the mean Hamming distance for each problem instance.

Figure 4.5 shows the difference in Hamming distance between the initial distribu-

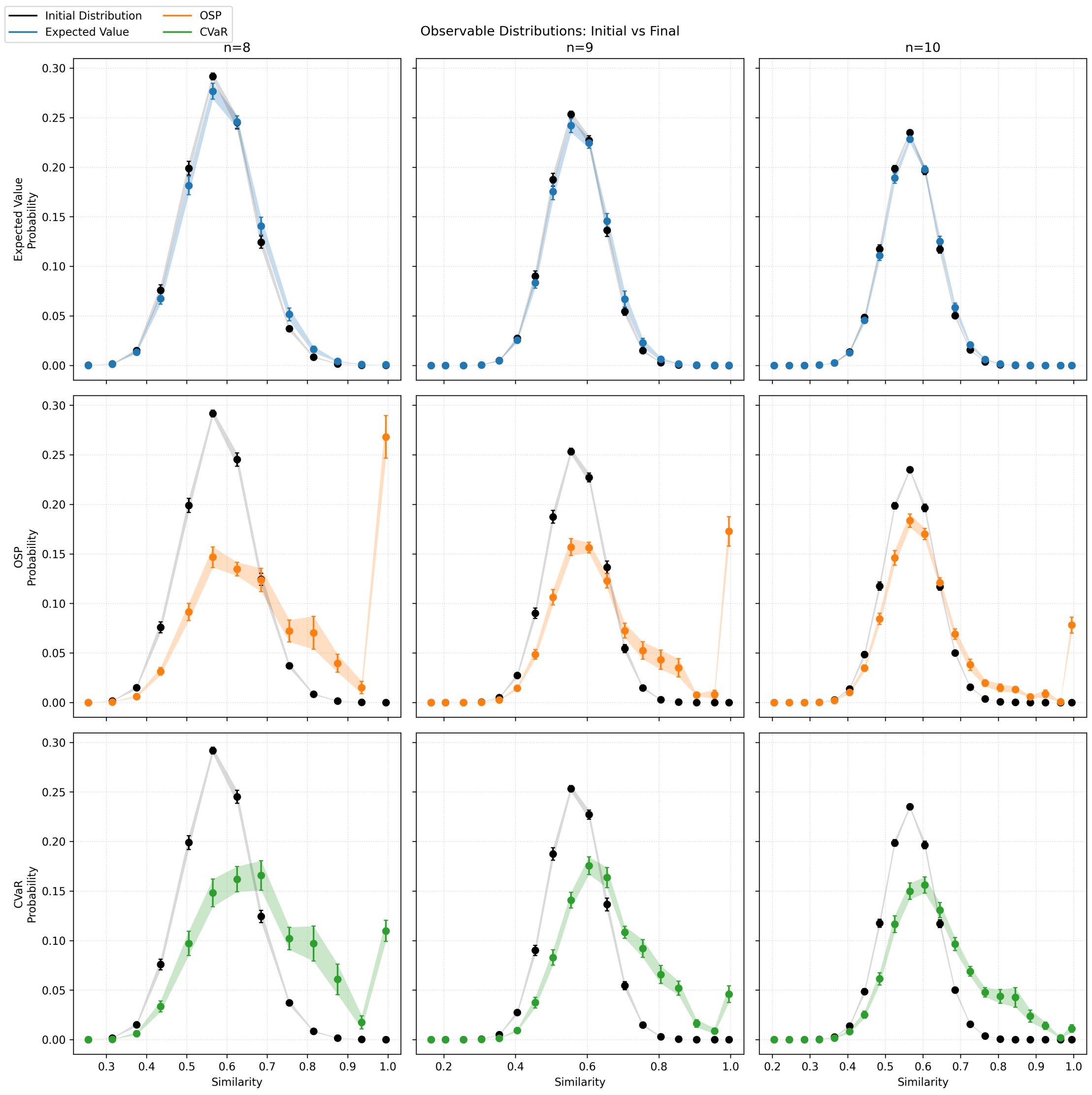


Figure 4.1: Probability distributions for different objectives and sizes

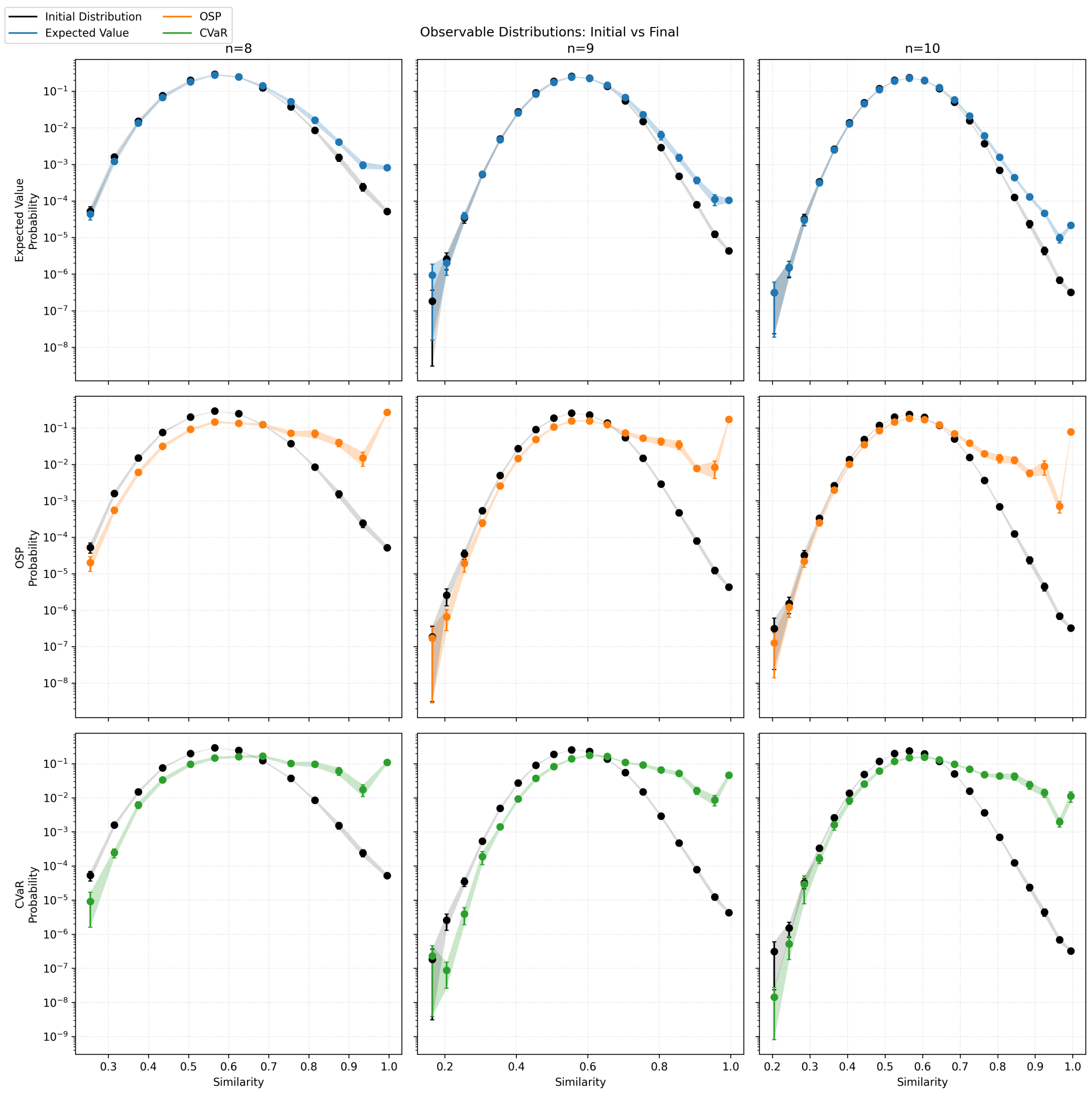


Figure 4.2: Log probability distributions for different objectives and sizes

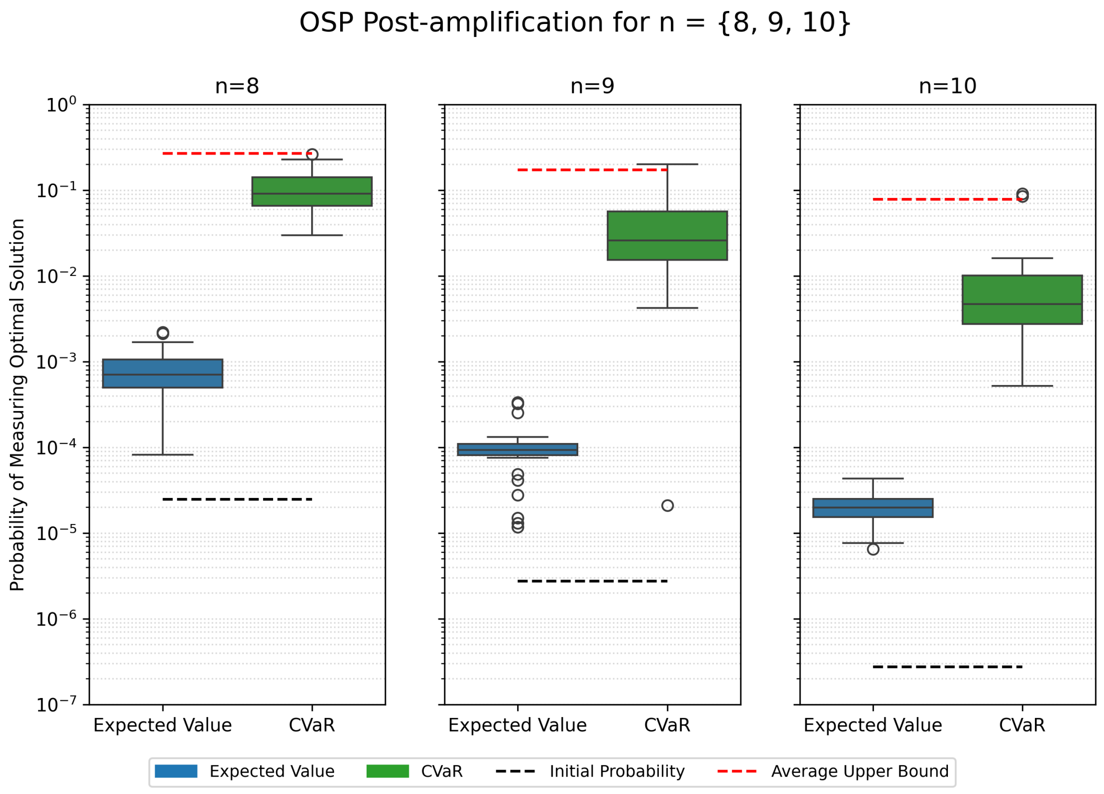
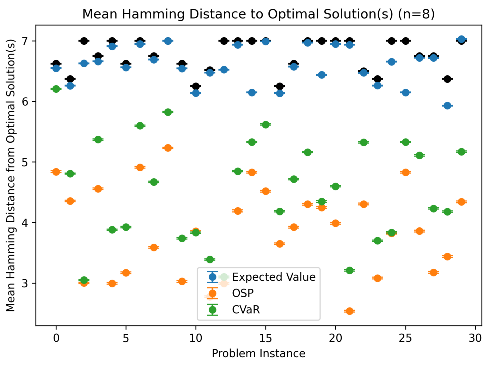
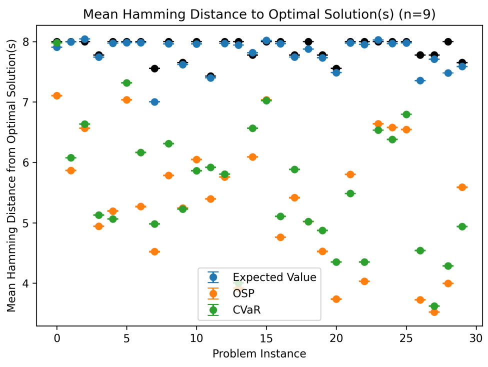


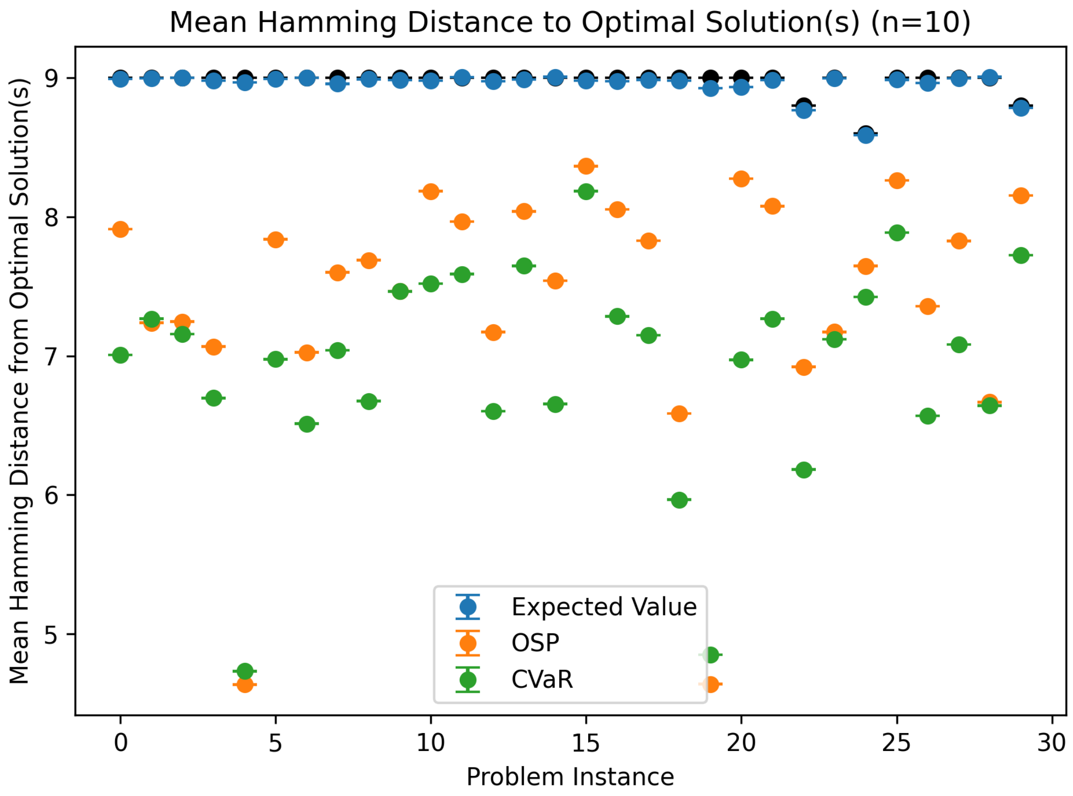
Figure 4.3: Optimal solution probability (OSP) distributions



(a) $n = 8$



(b) $n = 9$



(c) $n = 10$

Figure 4.4: Mean hamming distance for each problem instance.

tion and after amplification.

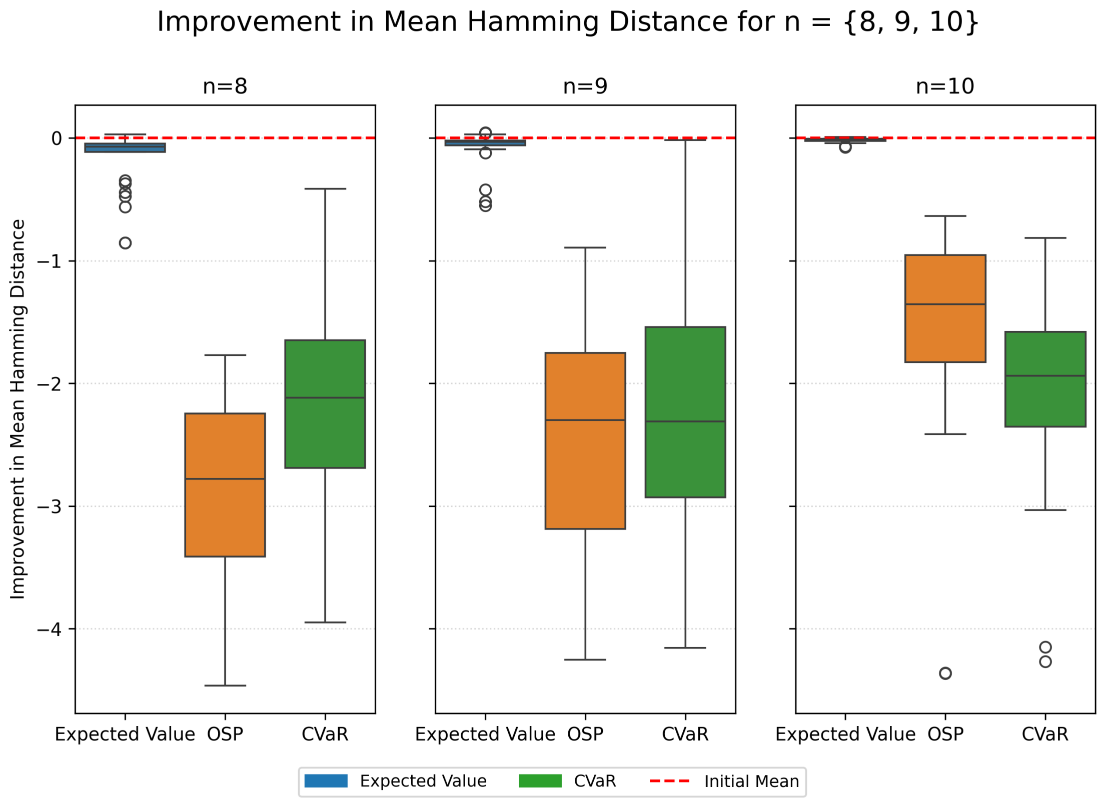


Figure 4.5: Improvement in Hamming distance boxplot.

Figure 4.6 shows how much amplification was applied to each solution based on their Hamming distance.

## 4.3 Subshell Distance Analysis

Figure 4.7 depicts the average subshell distance before and after amplification for each problem instance.

Figure 4.8 shows the difference in mean subshell distance after amplification.

Figure 4.9 shows how much amplification was applied to each solution based on their subshell distance.

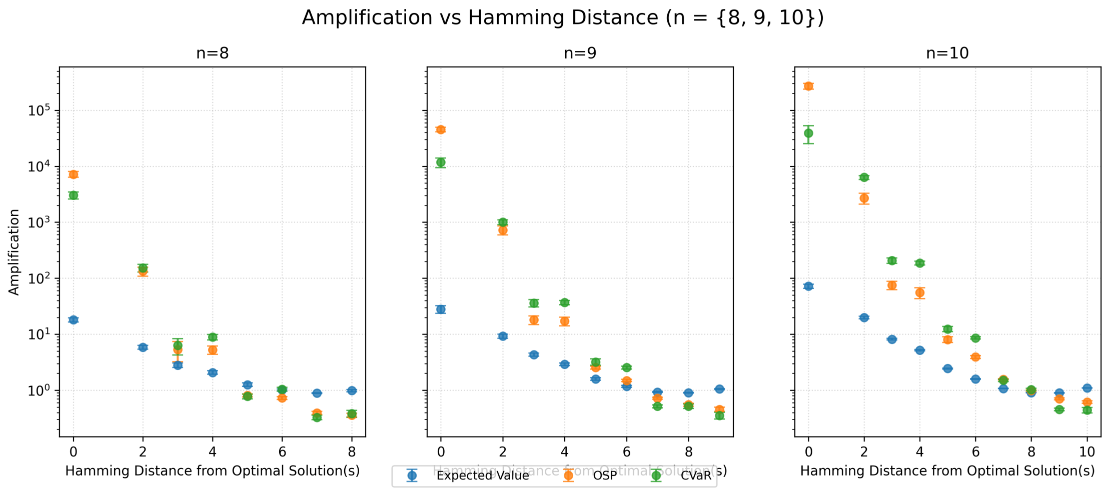


Figure 4.6: Mean amplification applied to each solution grouped by their Hamming distance from the optimal solution.

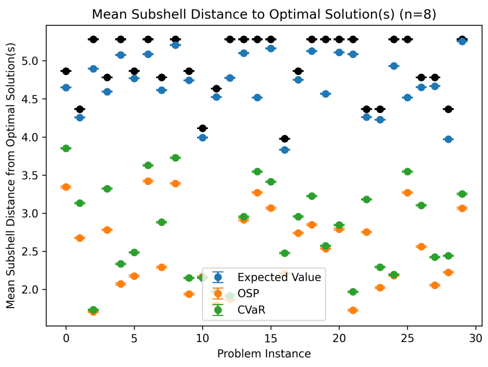
Figure 4.10 shows the mean quality gap between the optimal solutions and the subshells. Mention accounting for cases with $n - 2$ possible shell distances. And the fact that being monotonically increasing is a good thing.

Figure 4.11 shows the amount of variance in the objective values for each subshell. Lower variance indicates more phase-coherence within the shell, and therefore more potential for constructive interference with the target solution under optimisation of $t$ and $\gamma$.

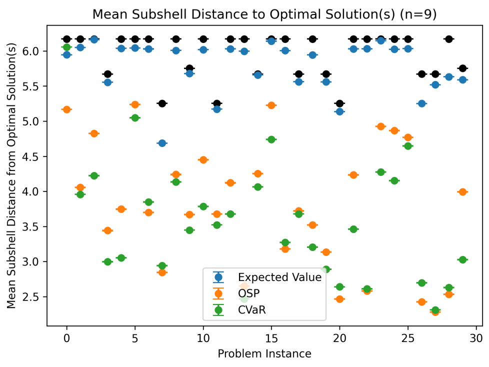
## 4.4 Hyperparameter Analysis

Table of mean parameters:

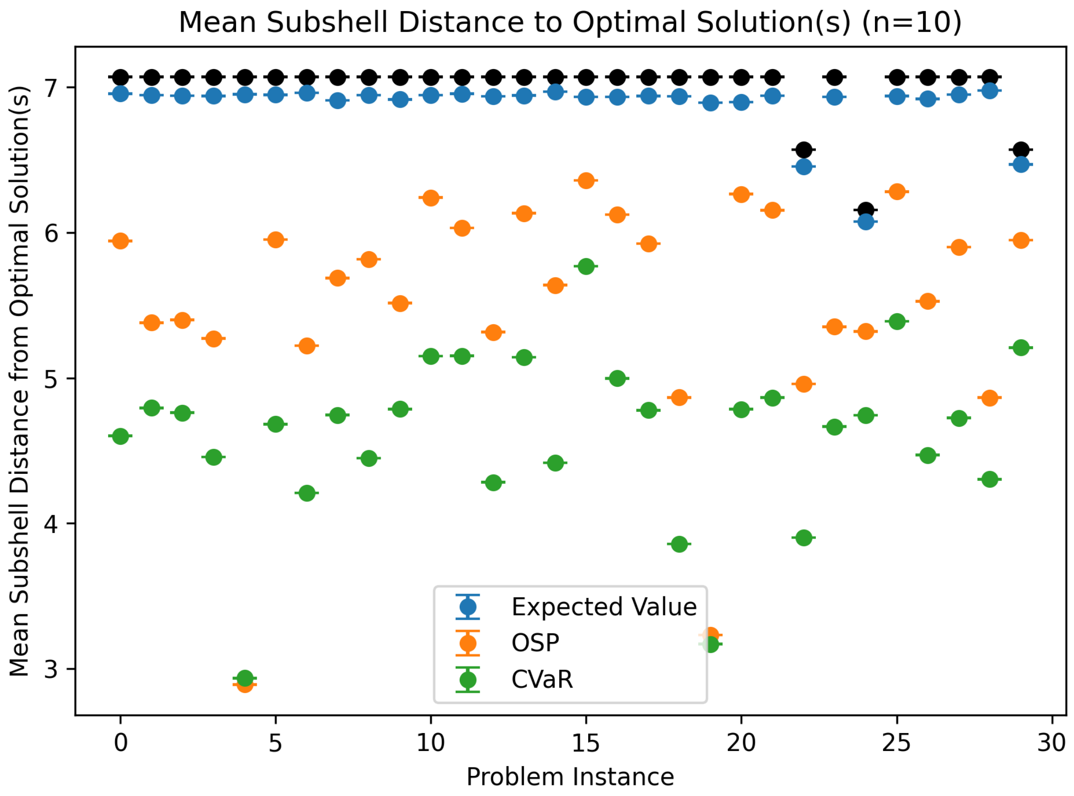
| $n = 8$ | $\beta$ | $\gamma$ | $t$ |
|---|---|---|---|
| Expected Value | 1.4(6) | 0.20(4) | 0.47(6) |
| OSP | 0.79(5) | 1.00(4) | 0.180(7) |
| CVaR | 0.54(5) | 1.2(1) | 0.17(1) |



(a) $n = 8$



(b) $n = 9$



(c) $n = 10$

Figure 4.7: Mean subshell distance for each problem instance.

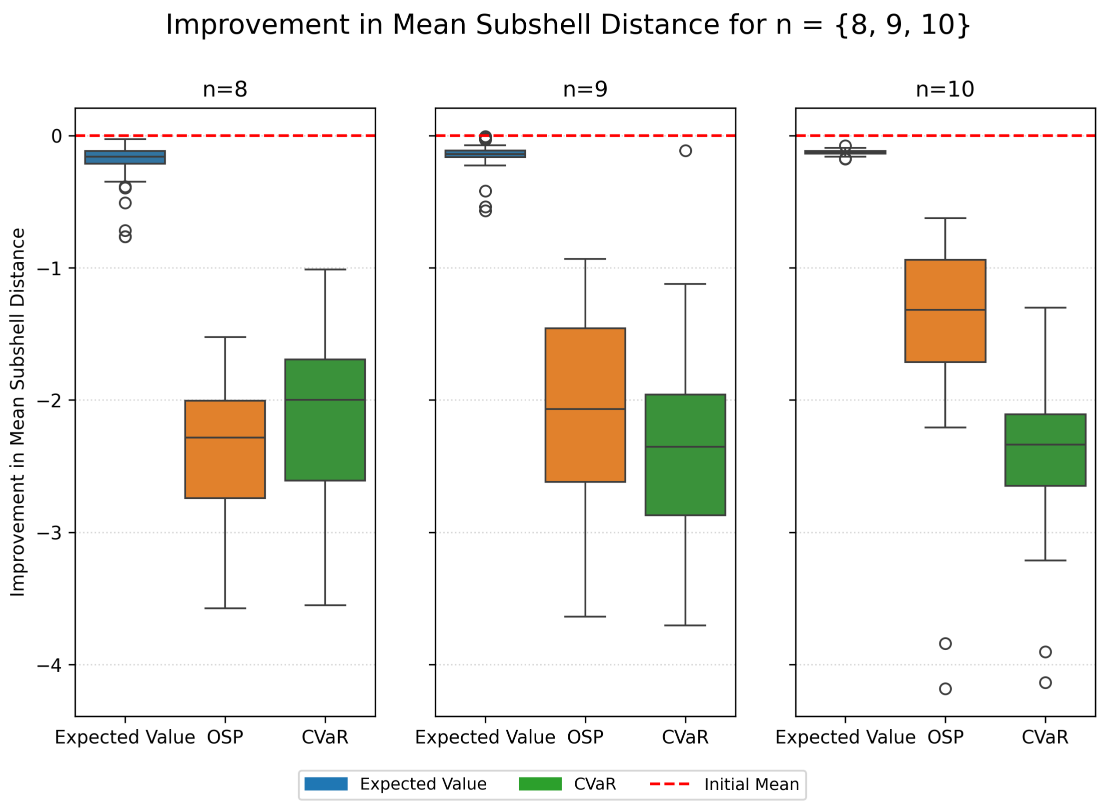


Figure 4.8: Improvement in subshell distance boxplot.

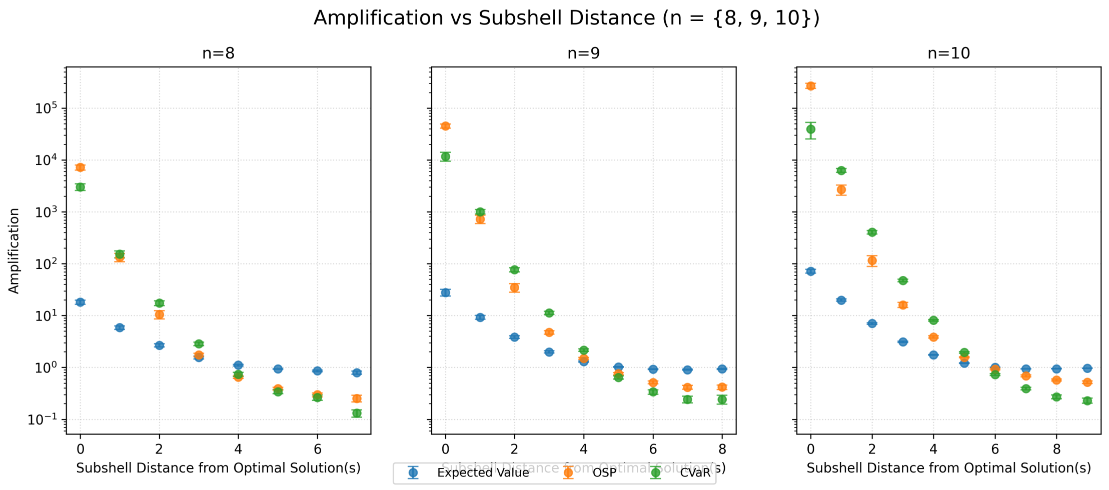


Figure 4.9: Mean amplification applied to each solution grouped by their subshell distance from the optimal solution.

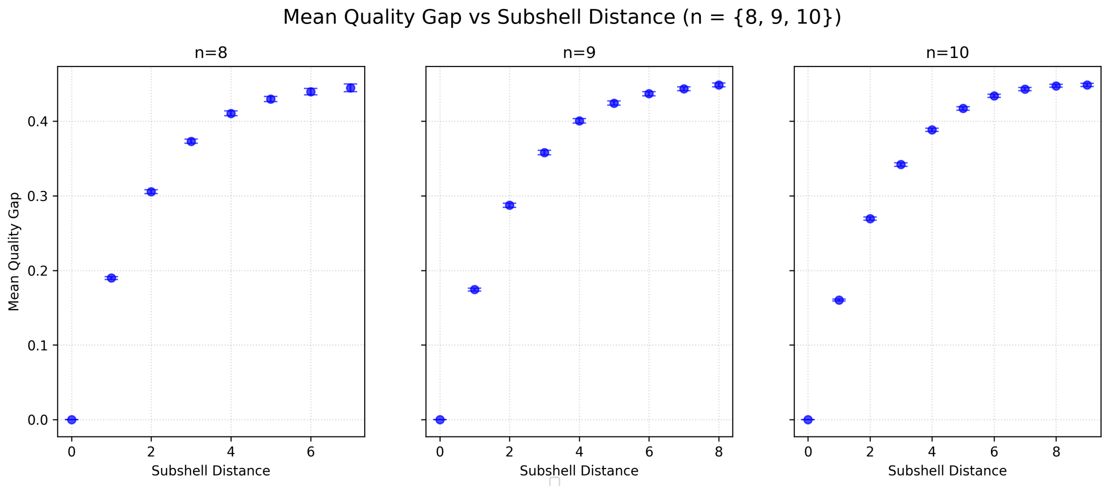


Figure 4.10: Mean quality gap vs subshell distance

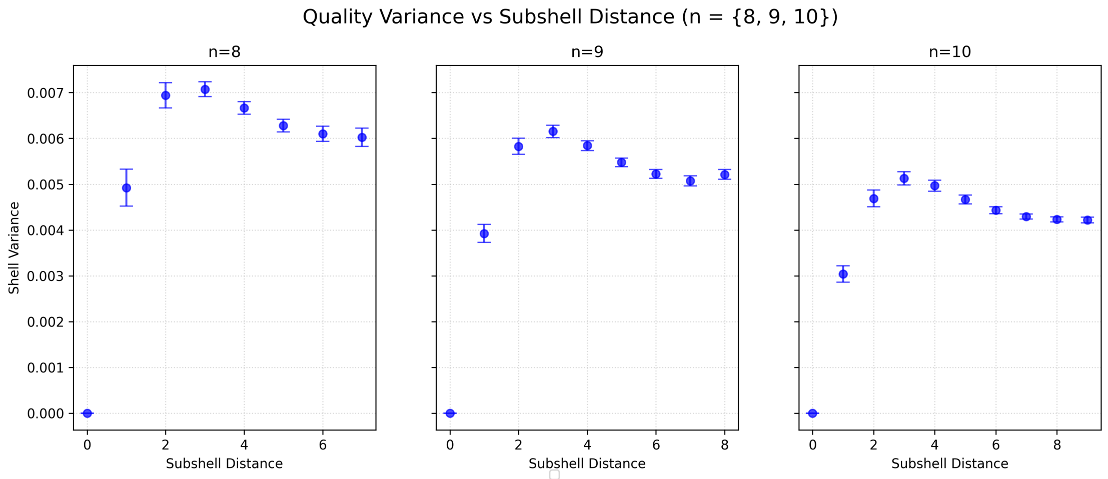


Figure 4.11: Shell variance vs subshell distance

| $n = 9$ | $\beta$ | $\gamma$ | $t$ |
|---|---|---|---|
| Expected Value | 1.08(3) | 0.20(3) | 0.40(6) |
| OSP | 0.84(3) | 1.01(4) | 0.164(7) |
| CVaR | 0.60(3) | 1.16(9) | 0.16(5) |

| $n = 10$ | $\beta$ | $\gamma$ | $t$ |
|---|---|---|---|
| Expected Value | 1.10(3) | 0.21(2) | 0.336(4) |
| OSP | 0.85(7) | 1.02(5) | 0.16(4) |
| CVaR | 0.58(2) | 1.28(5) | 0.124(6) |

# Chapter 5

# Discussion

The results of each area of investigation were largely expected and particularly show the effectiveness of using CVaR as a hyperparameter optimisation objective function.

## 5.1 Amplification Verification

The probability to measure the optimal and near-optimal solutions was consistently amplified by the NVQWOA, as shown in Figures 4.1, 4.2, and 4.3.

## 5.2 Hyperparameters

The choice of hyperparameter optimisation objective function had a large impact on the amount of amplification applied to the optimal and near-optimal solutions.

## 5.3 Solution Distance

Hamming distance is less smooth than subshell distance because there is degeneracy in the Hamming distance even for solutions with differing solution distance. For example, the permutation $[1, 2, 3, 4]$ has a Hamming distance of 4 from both $[2, 1, 4, 3]$ and $[2, 3, 4, 1]$, but its subshell distances would be 2 and 3 respectively.

Solutions with different subshell distances will be distributed differently on the mixing graph, so the probability will not be as concentrated.

Mean quality gap

Subshell quality variance

# Chapter 6

# Future Work

## 6.1 Graph Similarity Measures

## 6.2 Random Graph Generation

Exploring directed or weighted graphs

Exploring Watt-Strogatz, Barabasi-Albert graphs

## 6.3 Performance on Quantum Hardware

# Chapter 7

# Conclusions

# Bibliography

[1] Tavis Bennett, Lyle Noakes, and Jingbo Wang. Non-variational quantum combinatorial optimisation, 2024.

[2] Lov K. Grover. A fast quantum mechanical algorithm for database search. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 212–219, 1996.

[3] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, May 1935.

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

[5] Nicholas J. Pritchard. Quantum approximate optimisation applied to graph similarity, 2024.

[6] Danai Koutra, Joshua T. Vogelstein, and Christos Faloutsos. Deltacon: A principled massive-graph similarity function, 2013.

[7] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[8] Tavis Bennett, Lyle Noakes, and Jingbo B. Wang. Analysis of the non-variational quantum walk-based optimisation algorithm, 2024.

[9] Edric Matwiejew and Jingbo B. Wang. Quop_mpi: a framework for parallel simulation of quantum variational algorithms, 2022.

[10] Edric Matwiejew and Nicholas Slate. Edric-matwiejew/quop_mpi: 1.0.0 - 2021-09-30 - feature release, 9 2021.

[11] Panagiotis Kl. Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving variational quantum optimization using cvar. *Quantum*, 4:256, April 2020.

[12] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, January 1965.

# Chapter 8

# Appendices