Dissertation: Quantum Approximate Optimisation Applied to Graph Similarity

Nicholas Pritchard 21726929
Supervisors: Prof. Jingbo Wang and Prof. Amitava Datta

This report is submitted as partial fulfilment
of the requirements for the Honours Programme of the
School of Computer Science and Software Engineering,
The University of Western Australia,
2018

Abstract

Quantum computing promises solutions to classically difficult and new-found problems through controlling the subtleties of quantum computing. The Quantum Approximate Optimisation Algorithm (QAOA) is a recently proposed quantum algorithm designed to tackle difficult combinatorial optimisation problems utilising both quantum and classical computation[31]. The hybrid nature, generality and typically low gate-depth make it a strong candidate for near-term implementation in quantum computing. Finding the practical limits of the algorithm is currently an open problem. Until now, no tools to facilitate the design and validation of probabilistic quantum optimisation algorithms such as the QAOA on a non-trivial scale exist.

Graph similarity is a long standing classically difficult problem withstanding decades of research from academia and industry. Determining the maximal edge overlap between all possible node label permutations is an NP-Complete task which has faced little research from classical computer science and provides an apt measure of graph similarity. We introduce a novel quantum optimisation simulation package facilitating investigation of all constituent components of the QAOA from desktop to cluster scale using graph similarity as an example.

Our simulation provides class-leading flexibility and performance. We investigate eight classical optimisation methods each at six levels of decomposition; the most exhaustive study to date. Moreover a novel encoding for permutation based problems such as graph similarity through edge overlap to the QAOA allows for significant quantum memory savings at the cost of additional operations. This compromise extends into the classical portion of the algorithm as the inclusion of infeasible solutions creates a particularly difficult cost-function landscape.

We present performance analysis of our simulation and of the quantum algorithm itself setting a precedent for investigating and validating numerous other difficult problems to the QAOA as we move towards realising practical quantum computation.

Acknowledgements

Firstly, I would like to thank my supervisors, Prof. Jingbo Wang and Prof. Amitava Datta. Not only was their support invaluable but the faith they placed in me to learn in an area of study completely new to me was inspiring. Thank you for giving me the chance to learn some Physics.
I would also like to thank Matthew, Mitchell, Kooper, Gareth, Sam, Amit, Leigh, Edric, Ja-Jet and Lyle from the Quantum Dynamics Research Group who made the office arguably the most vibrant destination on campus and for their feedback, assistance and encouragement of my work.
Finally, I thank my parents for their unending encouragement and support throughout the year.

Contents

CHAPTER 1

Introduction

Seeking use from digital computers is an invariable goal in computer science. The computers we currently know stem from mathematics recruiting the laws of physics to realise our definition of computing. Graphs are widely used to represent real-world phenomenon in manner suitable for computation. Determining whether one graph is identical or similar to another when node correspondence is unknown is a long standing difficulty. Approximation of such measures prove useful in a myriad of real-world contexts. Quantum computing defines computation as a physical process linking computation to the underpinning physics stronger than previously encountered. Historically intractable problems can be explored in new ways as increasingly sophisticated quantum algorithms are formulated. Combinatorial optimisation is amongst the most general and practically applicable computational paradigms. Based on the formulation of quantum annealing the recently formulated Quantum Approximate Optimisation Algorithm (QAOA) [31] approaches NP-Complete combinatorial optimisation problems on discrete gate-based quantum computers. We explore the long-standing difficult problem of graph similarity via the QAOA using a problem encoding scheme novel to this algorithm. Furthermore, empirical validation is the only powerful method known to evaluate heuristic based or very probabilistic algorithms such as the QAOA. We present Qolab (Quantum Optimisation Laboratory), a software package designed to provide scalable efficient simulation of the QAOA from desktop to cluster scale for a generalised problem encodings and variations on the QAOA itself. A familiarity with the fundamentals of quantum computing is essential to understanding quantum algorithms. This familiarity is difficult to acquire without a background in quantum-physics. To aid with this we additional present a brief introduction to the field providing the minimal required knowledge to understand the QAOA. As physical implementations of gate-based quantum computers grow in scale, focus is shifting towards practical quantum computation with tangible real-world applications.

1.1 Computer Science Thus Far

Arguably, computing begins with the abacus. Over the past millennium humanity has discovered increasingly sophisticated methods to implement the model of computing. Such developments are governed by physics. The most radical discoveries in the field invariably drive innovation in the era following; Galileo formulated simple machines establishing a relationship between mathematics and theoretical and experimental physics. Newton's laws of motion gave birth to the machine age, similarly the discoveries of electromagnetism drove the development of the information age. We are yet to realise a fitting use for quantum mechanics. Quantum computing is the strongest candidate thus far and has garnered ferocious support from private industry and public institutions across the globe. Quantum computing represents a natural development that radically diverges from classical computer science.

1.2 Combinatorial Optimisation

A combinatorial optimisation problem consists of finding some optimal object from a finite set of possible choices. Specifically we define combinatorial optimisation problems with respect to the following components.

Definition 1.2.1. A combinatorial optimisation is specified precisely by the following components

- A specific problem type I. We must be able to efficiently determine if an arbitrary problem belongs to the set we are concerned with.
- For each valid problem instance $p \in I$ a solution validation function $S : p \to \mathcal{P}(\mathcal{U})$ which determines if a given input x is a feasible solution to p. The computational resources required to store both x and p must be bounded by some polynomial. This is required to efficiently verify an arbitrary solution y as a valid solution to p.
- An objective function $C(x) : I \times \mathcal{U} \to \mathbb{Z}$ which maps a feasible solution to a non-negative integer value indicating the quality of the solution. The highest value of $C(x)$ in the set of feasible solutions indicates the optimal solution for a given problem instance

Given a problem instance p of type I we aim to find an x such that

$$C(x) = max(S(p)) \tag{1.2.1}$$

This leads to a clean definition of *NP-optimisation* problems.

Definition 1.2.2. An NP optimisation problem NP is a combinatorial optimisation problem where the set of feasible solutions $S(x, NP)$ cannot be exhausted in polynomial time.

We further define a *bounded optimisation problem NPO-PB*

Definition 1.2.3. A bounded NP optimisation problem $NPO - PB$ is an NP optimisation problem with the additional constraint that the resources required to represent $C(x) \forall x \in S(NPO - PB)$ must be bounded by some polynomial.

Given the intractability of NP and $NPO - PB$ optimisation problems approximation algorithms are tolerated relaxing the problem to find an optimal approximate solution. This general problem description captures a number of useful problems in computer science such as planning, scheduling, protein folding, vertex colouring and the travelling salesman problem [34]. Finding approximate solutions of superior accuracy for such problems is an open area of research in classical computer science.

1.3 Graph Similarity

Graphs are well generalised mathematical structure. A graph encodes relations between entities and as such graphs can represent a vast number of real-world problems. The features of faces, topology of the Internet, road-networks, decision-flow, computer programs, cosmological bodies and any other number of natural and unnatural phenomenon can be expressed through this marvellous data-structure. For completeness we define the graph.

Definition 1.3.1. A graph $G(v, e)$ is a collections of vertices v and a collection of (v, v') pairs termed edges e which may be directed or un-directed [25].

The rich history of graph theory provides many metrics on graphs such as shortest path-length. While many of these measures are trivial or at least tractable to compute they typically concern themselves with the internal structure of a particular graph. Finding information about the overall structure of a graph is a much more laborious task. Graph isomorphism is a quintessential structural problem when comparing graphs; are two given graphs alike? This problem has no P algorithm and showing NP-completeness is a long-standing open problem [34] and so it occupies a unique complexity class often termed graph isomorphism

complete [86]. Difficulty arises from the factorial number of mappings between the features of two unlabelled graphs with unknown node correspondence. Relaxing the isomorphism problem provides numerous measures of similarity between graphs which are known to be NP-Complete [34] and have been given a large amount of academic and industrial attention.
We specifically investigate a measure of *whole graph similarity* which we defined as

Definition 1.3.2. Whole Graph Similarity: Given two graphs $G_1(v_1, e_1)$ and $G_2(v_2, e_2)$ with possibly different numbers of vertices and edges, find an algorithm which returns a measure of similarity $S|S \in [0, 1]$. Furthermore:

1. $S(G_1, G_1) = 1$
2. $S(G_1, G_2) = S(G_2, G_1)$

This method is simple to understand and allows for concise error determinations when comparing a brute-force optimal and computed approximate solution. Further, we define and present the measure of edge overlap (EO) as a whole graph similarity measure adhering to Definition 1.3.2 Consider two directed, un-weighted graphs G_1 and G_2 with no known vertex labelling. For each permutation σ of potential node labels between G_1 and G_2 we provide a penalty of one for every edge (or non-edge) which differs between the two graphs. This penalty score is then normalised by the maximum number of edges possible. The maximum possible score is bound by the number of edges possible which is v^2 for a directed graph, $v(v-1)$ for an un-directed graph with self-edges and $\frac{v(v-1)}{2}$ for an un-directed graph without self-edges. Normalisation yields an edge-difference value e_d which we take as our similarity value. Importantly, graphs of differing numbers of vertex can be compared by adding degenerate, unconnected nodes to the smaller graph. The computational difficulty of finding this value arises from the factorial number of possible node-labelling permutations.
For example, consider the two graphs depicted in figure 1.1. The maximum number of identical edges is 14 counting the identical edges present and identical edges which are missing (they only differ in two edges). Thus the graph similarity for these two is $1 - \frac{2}{4^2} = 1 - \frac{1}{8} = 0.875$.

1.3.1 Graph Similarity as Combinatorial Optimisation

We map graph-similarity to a bounded NP-optimisation problem as follows.

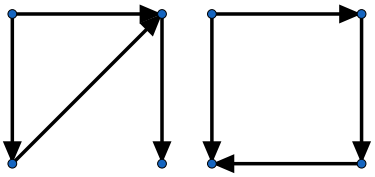


Figure 1.1: Two example unlabelled graphs (G_1 and G_2)

- The set of problem instances I is the set of all un-weighted graphs.
- Given a pair of graphs G_1, G_2 the set of feasible solutions is the set of mappings between the vertices of G_2 to G_1. The number of candidate mappings grows $\mathcal{O}(V^2)$ hence growing non-polynomially. A solution can be validated by checking that each vertex is mapped and that the mapping is bijective.
- The objective function is the edge-overlap between graphs G_1, G_2 under a candidate mapping which is bound by V^2.

Algorithm 1 Classical brute-force algorithm to find Maximal Edge Overlap

```
1: G_1 ←< E_1 >
2: G_2 ←< E_2 >
3: Best← 0
4: for each Permutation X of 1, 2...V do
5:     C(x) ← V^2
6:     for i ← 0 to V do
7:         for j ← 0 to V do
8:             if G_1[i, j]! = G_2[X[i], X[j]] then C(x) ← C(x) − 1
9:             end if
10:        end for
11:    end for
12:    if C(x) >Best then
13:        Best ← C(x)
14:    end if
15: end for
16: return Best
```

Many classically difficult problems can be simple to define but remain difficult due to the explosive growth of the solution space. In the case of graph-similarity the number of candidate solutions grow $\mathcal{O}(v!)$.

This exact formulation of graph-similarity demands a brute-force algorithm to find an exact solution presented in Algorithm 1.3.1 as it is feasible for only a single candidate to be optimal (consider two isomorphic graphs for example).

1.4 The Quantum Approximate Optimisation Algorithm

The Quantum Approximate Optimisation Algorithm (QAOA) is a hybrid quantum-classical algorithm designed to approximately solve NP-Complete combinatorial optimisation problems [29]. The QAOA is unique among other notable gate-based quantum computing algorithms for two. The gate depth required for useful computation is low and the approximate nature of the algorithm make it suitable for near-term, noisy quantum computers. Furthermore, the general formulation of the algorithm inspires other similar algorithms which allow for more sophisticated problem encoding. The QAOA and derivative algorithms are likely candidates for near-term practical use. Finding the practical limits of this algorithm remains an open problem.

1.4.1 The Quantum Adiabatic Algorithm (QAA)

Farhi et al. [32] note the possibility of exploiting the exponential number of items a qubit register can represent and formulate the quantum adiabatic algorithm (QAA). The QAA is founded in by the *quantum adiabatic theorem* where a quantum systems evolves according to the Schröidgner equation

$$i\frac{d}{dt}\left|\psi(t)\right\rangle = \hat{H}(t)\left|\psi(t)\right\rangle \tag{1.4.1}$$

The QAA utilises a quantum register of n qubits. Two system Hamiltonians are prepared $\hat{B}$ and $\hat{C}$. $\hat{B}$ often termed the driver Hamiltonian is an easily prepared maximal energy state. $\hat{C}$ often termed the problem Hamiltonian encodes the values of all possible solutions. Importantly, $\hat{C}$ encodes the problem solution as the highest energy-state of our quantum system. The Hamiltonian governs the evolution path of our system according to

$$\hat{H}(t) = (1 - \frac{t}{T})\hat{B} + \frac{t}{T}(\hat{C})) \tag{1.4.2}$$

such that $\hat{H}(0) = \hat{B}$ and $\hat{H}(T) = \hat{C}$. Adiabatic evolution ensures that if $\hat{B}$ begins in a maximum-energy state, as $t \to T$ the system will remain in a maximal energy state. Measurement at time T should yield a near-optimal solution with high-probability [32].

1.4.2 Moving to the Quantum Approximate Optimisation Algorithm

The Quantum Approximate Optimisation Algorithm (QAOA) [31] builds on the foundation of the QAA by noting that adiabatic evolution is impossible to implement exactly. A Suzuki-Trotter decomposition of the evolution into discrete increments is much simpler however and can be implemented in a gate-based quantum computer [85].
Farhi et al. define combinatorial optimisation problems with regards to *maximum satisfiability*. The corresponding cost function is defined as

$$c(x) = \sum_{i=1}^{m} c_i(x) \tag{1.4.3}$$

$c_i(x)$ check if the i-th clause is satisfied by the input bit-string. Since this problem is NP-Complete, any other NP-complete problem can be mapped to this formulation in polynomial time. The solution can be encoded as a diagonal Hamiltonian $\hat{C}$ where the i-th eigenvalue contains the cost-function value of i as a n-length bit-string. This diagonal operator is defined by the action of $\hat{C}$ on the computational basis states, just as any other quantum-computing gate.

$$\hat{U}_C(\gamma) = e^{-i\gamma\hat{C}} \tag{1.4.4}$$

where γ is a real-valued parameter which is restricted to the interval $[0, 2\pi)$. Importantly, the QAOA operates on *all* bit-strings of length n without regard for feasibility. This is not an issue for problems like MAX-SAT where all bit-strings are valid but cause issue for problem with more nuanced encodings.
An operator $\hat{B}$ defines how candidate bit-strings are considered. The canonical formulation is given by

$$\hat{B} = \sum_{i=1}^{n} \sigma_i^x \tag{1.4.5}$$

Where σ_x^i is the Pauli-X operator, the quantum equivalent of the NOT gate. This Hamiltonian allows all candidate bit-strings to be considered without restriction. Similar to $\hat{U}_C$ we define $\hat{U}_B$ as

$$\hat{U}_B(\beta) = e^{-i\beta\hat{B}} \tag{1.4.6}$$

The resulting quantum state for a given set of parameters is by

$$\left|\vec{\gamma}, \vec{\beta}\right\rangle = \hat{U}_B(\beta_p)\hat{U}_C(\gamma_p)...\hat{U}_B(\beta_1)\hat{U}_C(\gamma_1)\left|\psi\right\rangle \tag{1.4.7}$$

Where $|s\rangle$ is a trivial equal-super-position of n qubits. In practice repeated sampling of a single QAOA-iteration functions as the output of our system. This output is defined analytically as the expectation value of $\left|\vec{\gamma},\vec{\beta}\right\rangle$, $F_p(\vec{\gamma},\vec{\beta})$ and is defined by the equation

$$F_p(\vec{\gamma},\vec{\beta}) = \left\langle \vec{\gamma},\vec{\beta}\right| \hat{C} \left|\vec{\gamma},\vec{\beta}\right\rangle \tag{1.4.8}$$

For reference the expectation value of a quantum state $|\psi\rangle$ is computed as a dot-product of $|\psi\rangle^\dagger$, $\hat{C}$ and $|\psi\rangle$ and is effectively a weighted sum of the probability to measure a given bit-string multiplied by its corresponding cost-function value. Typically, the amount of decomposition is small and is expressed as the number p (≈ 2). Finding a solution to the original combinatorial optimisation problem is now accomplished by a parameter search on the $2p$ transformation parameters, which can be performed classically.
The QAOA is therefore a generalised frame-work application to a wide range of combinatorial optimisation problems and a strong candidate for near-term practical use. We present a high-level circuit representation of the QAOA in Figure 1.2.

1.4.3 Simulating the QAOA

Classical simulation of an arbitrary quantum process is a well-established difficulty [33]. Additionally, the exponential and anti-intuitive behaviour of qubits make designing quantum algorithms a generally difficult task. The current demand for exacting bit-level design for quantum algorithms makes experimentation with general quantum algorithms such as the QAOA exceedingly difficult. Current quantum computing simulations either lose performance by offering generality or lose generality by offering extreme performance for a single task. This motivates the development of a simulation package targeted at the general QAOA frame-work intending to provide a compromise between the two extremes. By providing an exacting simulation of the QAOA we make the best use of *high-performance computing*(HPC) tools allowing for powerful empirical validation. Experimental analysis is essential to understand the QAOA since closed-form correctness analysis is infeasible for all but the most trivial, well-conditioned cases [38].

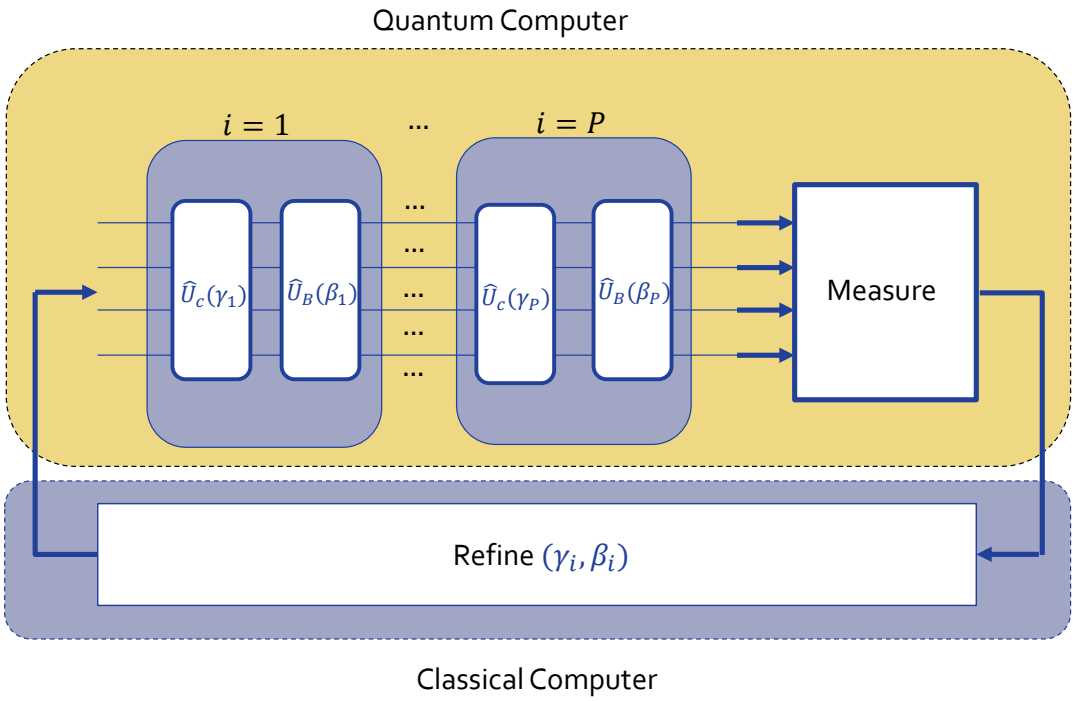


Figure 1.2: Circuit diagram of the QAOA framework

1.5 Unifying Hypothesis

Given the long-standing combinatorial difficulty in comparing whole-graph structure, the advent of near-term quantum computing hardware and a ferocious appetite for exploring the use of such machines it seems appropriate to investigate quantum combinatorial optimisation with regards to graph similarity.
We propose that for a fixed number of samples the QAOA will generate more correct solutions to naïve random-sampling for determining the maximal edge-overlap between two directed, un-weighted graphs. We expect improved performance for undirected graphs and that the number of quantum evaluations required will scale favourably establishing the QAOA as a valid approach to graph similarity.

CHAPTER 2

Literature Review

We present a discussion of literature surrounding graph similarity measures, a rapid introduction to gate-based quantum computing, a survey of current investigation into the QAOA and finally a discussion on computing the matrix exponential; a cornerstone of simulating the QAOA.

2.1 Graph Similarity

2.1.1 Applications of Graph Similarity Measures

The generality of graph-theory generates many important real-world contexts where a measure of similarity between graphs is valuable. Mapping an individual's social network has remained of interest since the original publication of the 'small-world' phenomenon which in itself provides a good model for understanding the relationship between information sources [50]. Computing the structural neighbourhoods of vertices and edges provides methods for indexing the world wide web allowing a level of access to information unparalleled in all of human history [14]. Strug [82] formulates a machine learning method to evaluate the quality of a design through graph similarity between components. In the field of data-mining classifying graphs based on similar features is applicable to many problem settings such as social network mining, drug design and anomaly detection in program-execution [40]. Facial recognition and object tracking is an obvious application for graph similarity if features can be mapped onto graph structures [88]. Heymans and Singh [44] frame the determination of evolutionary pathways in terms of graph similarity computed between the metabolic pathways of varying organisms. The problem of chemical compound matching has been a common benchmark problem prompting strong commercial and academic backing for a number of years [40], [71], [41]. Just as graphs are able to express a vast number of phenomenon the measures constructed to compare them are just as varied, finding structural similarities between graphs is a long-standing intensely valuable endeavour.

2.1.2 Classical Graph Similarity

The precise methods used to approximate graph similarity are as varied as the measures defined. However there are two general types of similarity measures commonly employed; whole-graph similarity and vertex-wise similarity [56].

Vertex-Wise Similarity

Measures of similarity formulated per vertex maintain a multitude of specific formulations. However, they all follow the same basic paradigm; vertices are considered similar if their neighbourhoods are similar. Such measures are far removed from the definition of graph isomorphism and seldom penalise differences between graphs. Lades et al. [57] presents an object recognition scheme extending classical artificial neural-networks to a more dynamic architecture testing their method by formulating facial-recognition as an elastic graph matching problem. Fortunately in the field of machine vision a degree of error is tolerated as the main design criterion for such systems is efficient calculation of average-case approximations [57]. Nevertheless, the search for more accurate and robust systems yields the extremely fruitful field of machine learning and machine vision we see today.
Analysis of the network structure of the Internet provides a large example of graph similarity measures. Kleinberg [55] proposes the extraction of improving Internet search queries based on the structure of hyper-links between pages. The neighbourhood of vertices are examined to determine high-quality pages to return as results to user queries. The ranking of pages is defined with regards to 'hub' and 'authority' pages. A 'hub' is a page which references many high-quality authorities and an 'authority' pages is one which is referenced by many high-quality hubs [55]. Clearly, there exists a large overlap in links between any pair of hub and authority pages sharing a similar neighbourhood; in this sense ranking such pages is a graph similarity problem. Kleinberg [55] further formulates this computation as an eigenvalue decomposition on the quality values of candidate pages. This approach was later extended by Brin and Page [14] in the design of the Google search engine. Restriction of scale to a local neighbourhood at any given vertex provides tractability. The success of the vertex-wise description of graph similarity is testament to the power of structural information present in graphs driving Google to forefront fields in computer science such as machine learning [80] and quantum computing [11], [12], [54]. Zager and Verghese [89] use vertex-wise similarity to compute whole graph matching. While results are promising and significantly better than random assignment the Hungarian algorithm employed to perform matching produces imperfect results. This preliminary work highlights the difficultly of the more general problem of graph matching. Later, Kolias et al. [56]

extend vertex-similarity calculations to a graph-global problem presenting scaleable parallel algorithms applicable to graphs two orders of magnitude larger than previously feasible on the order of millions of vertices.

Whole Graph Similarity

Whole graph similarity aims to compute how similar to given graphs are in their entirety as a relaxation of the graph-isomorphism problem. The maximum common sub-graph problem [34] is a very general metric which captures a natural sense of global similarity. The goal is to find the largest collection of matching vertices and edges present in two graphs. Determining the maximum common sub-graph is intuitively applicable to the analysis of chemical compounds. Hattori et al [41] show this method to be effective to not only determine compound similarity in of itself but to discover and classify systemic aspects of biology. Atoms make a natural analogy to vertices as bonds between them do as edges. The maximum common sub-graph is a particularly applicable measure to biological and chemical sciences as many larger compounds are built from well known constituent cliques. As such the techniques employed by Hattori et al. [41] make heavy use of heuristics to overcome the computational complexity involved with determining common structure. The computation of the maximum common sub-graph is in general NP-hard [34] but in the context of chemical compounds this is not entirely accurate [41]. The natural limits on atomic bonding limit complexity somewhat; carbon atoms are only permitted to maintain a maximum of four bonds for instance [41]. The use of heuristics significantly increases computation efficiency but degrades solution quality appropriately.

Strug [82] constructs a similar measure of maximum sub-graph comparison but in a generalised context of computer aided design. The problem is framed with regards to hierarchical graphs creating a natural set of useful sub-graphs to be compared thus reducing the overall number of graph comparisons required. Strug [82] employs machine learning inspired methods of kernel matching to approximate solutions degrading solution quality in a similar trade-off for computational performance as other methods.

Papadimitriou et al. [68] apply maximum sub-graph matching to identify graph-dissimilarity. By tracking snapshots of the Internet described as a graph one can identify anomalies through the differences between them. Three types of anomalies are searched for: missing connected sub-graphs; missing random vertices, and connectivity changes. Five methods to compute whole graph similarity are considered.

Vertex ranking computes the correlation between two sets of vertices given a pre-computed quality score for each and is the least successful method evaluated [68].

Vertex similarity, is related to methods discussed previously in section 2.1.2. This method performs admirably but is not superior to many other methods.
Vertex and edge-overlap computes similarity between two general graphs. Simply stated, this method considers two graphs similar if they share a large number of edges and vertices. This method utilises edit distance as a measure of similarity capturing a natural intuition for similarity [68]. A similar edit-distance measure is employed by Zheng et al. [90] for searching graph databases. Various filters are employed to reduce the computational complexity of chemical database searches by returning graphs with an edit distance below a carefully chosen bound. This method outperforms contemporary solutions by a significant margin [90]. With regards to anomaly detection however, vertex overlap fails to detect large missing connected sub-graphs. [68].
Sequence similarity considers graphs to be similar if they share a large number of smaller sub-graphs. Papadimitriou et al. [68] utilise their own method of shingling which converts a graph to a linear sequence of tokens. This method is poorly suited to detecting anomalies between graphs but is better suited to structures which are inherently linear. The final method considered by Papadimitriou et al. [68] computes a signature of each graph and uses the hamming distance between the two as a measure of similarity. Signature based similarity shows the best performance in detecting anomalies.
Yongkoo et al. [40] classify graphs by sub-graph matching based on selective applications of exact graph isomorphism tests. Computational efficiency is derived from the intelligent choice of features to be tested. Features are chosen by building a topology where frequent sub-graphs are given an identification tag which can then be queried quickly. Results show significant improvements in accuracy and speed in classifying anticancer behaviour over leading implementations [40]. However, solutions are still no more than 86% accurate.
Computing measures of graph-wide similarity remains to be a difficult problem approximated by many algorithmic methods. The widespread use of graph similarity measures in a broad-spectrum of problem contexts promote consistent attention from academic and industrial institutions. Despite decades of research no tractable exact algorithm has been found and all known solutions make significant sacrifice to solution quality or scope. This long accepted difficulty makes graph similarity a prime candidate for investigation by quantum computing.

2.2 Quantum Computing

Quantum mechanics stands among the most notable scientific achievements of the 20^{th} century describing phenomena unlike anything previously encountered. Rem-

iniscent of how electromagnetism spawned the realisation of the digital computer, practical applications of quantum physics promise exotic new technologies. Quantum computing is a major frontier in this regard. We present an introduction to fundamental concepts and historic developments in the field suitable for readers without a strong background in quantum physics.

2.2.1 Bits vs. Qubits

The fundamental difference between classical and quantum computing lies in the physical paradigm used to represent atomic data. In classical computing all data is represented in a register of bits each maintaining the value of zero or one. The exact construction of such a register varies, the fundamental principle however, remains constant. Feynman [33] makes the elegant observation that the simulation of quantum phenomenon and quantum systems is extremely difficult to perform classically. The natural extension to this observation is whether we can use these inherently complicated quantum phenomenon to perform useful computation. Deutsch [23] later defines the quantum Turing machine by refining the fundamental Church-Turing hypothesis from an abstract construction to the physically related Church-Turing principle. Re-framing the definition of computation with explicit regard to the laws of nature results in a whole new paradigm of computing machines which base their principles on quantum phenomenon. This discovery has driven a quest extending over three decades to construct such machines and to realise the upper echelons of computation permitted by the physical world.

The fundamental building block of quantum computing is the logical abstraction of any two-state quantum system known as a qubit [66]. Mathematically we use *dirac* [26] notation to represent the state of a qubit; as an example analogous to classical bits a qubit can exist in the $|0\rangle$ or $|1\rangle$ state. One drastic difference is that a qubit can exist in any linear combination of these states called *superposition*

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.2.1}$$

where α and β are any complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. The states $|0\rangle$ and $|1\rangle$ specify a *computational basis* but we could just as easily specify any other pair of basis states as long as they are orthonormal.

To quickly grasp the potential for computational power qubits afford first consider the unit sphere. In classical computing, each bit is permitted to only occupy one of the poles, each corresponding to either a zero or one. In contrast a qubit can occupy any combination of the two states $|0\rangle$ and $|1\rangle$ and can thus occupy any point on surface of the sphere. This representation is known as the *bloch sphere* and is a very useful tool to represent the state of a qubit [66]. One might believe that

this allows a single qubit to store infinite information as there are infinitely many unique points on a sphere however this is not exactly the case. This superposition of basis states is only present during a quantum computation, when a measurement is made a qubit will collapse onto one of the two basis states. More specifically, a measurement will collapse a qubit into either basis state with probabilities $|\alpha|^2$ or $|\beta|^2$ respectively. This fact captures the core difficulty in the design of quantum over classical algorithms; we can only measure a qubit once. Classical computing is built upon setting and examining the state of bits explicitly and freely. The techniques demanded in quantum computing are more nuanced.
While a measured qubit will only yield a single bit of information, nature is excellent at keeping track of all the quantum information stored by a qubit in superposition. The goal of quantum computing then is to extract as much of this information as possible.

2.2.2 Logic Gates and Quantum Circuits

A qubit register is simply a collection of multiple qubits just as a classical register is simply a collection of bits. If we have n classical bits then together there are 2^n possible values that register can represent but only one is represented at any given time. This is already powerful but the quantum mechanical nature of qubits allow for vastly more power; a register of n qubits in superposition can represent 2^n states *simultaneously*.
Analogous to classical circuits, a quantum circuit is a series of quantum logic gates operating on some initial quantum register state. Typically, gates are defined with regards to their actions on qubits which can be described in matrix form. Incredibly, the only restriction required for a quantum logic gate is that it is a unitary operator.

Definition 2.2.1. A unitary operator ($\hat{U}$) when multiplied by its conjugate transpose ($\hat{U}^\dagger$) results in the identity matrix ($\mathbb{1}$) [66].

For example, the single-qubit NOT gate swaps the amplitudes of a qubit. One can describe the effects of this gate using the following matrix

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2.2.2}$$

We see the effect of this operation on the qubit $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}. \tag{2.2.3}$$

There are a number of possible sets of quantum logic gates which define universal computation, we present one such set.

Phase Shift

The phase shift gate realises an arbitrary rotation in the computational basis. This gate has no real classical analogy as it operates exclusively on superposition states. Importantly, this gate leaves the probability amplitudes of a qubit untouched. It is described by

$$R_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}. \tag{2.2.4}$$

Hadamard Gate

The Hadamard gate acts on a single qubit to produce an equal-superposition of two basis states. One may intuitively think of the Hadamard gate as transforming a qubit 'halfway' between two basis states [66]. Successive applications of the Hadamard gate on each in a register of n-qubits $|0..0\rangle$ will result in an equal superposition between the entire computational basis represented by n-qubits. It is described by the matrix

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.2.5}$$

Control Not (CNOT)

The control-NOT (CNOT) gate acts on two qubits known as the control $|x\rangle$ and target $|y\rangle$ qubits. The CNOT gate performs a logical NOT on the target qubit if and only if the control qubit is in the state $|1\rangle$ but leaves the control qubit unchanged. One can see that this is a quantum analogue to the classical XOR-gate. One can express the effect of this gate on two states as $|\psi, \phi\rangle \rightarrow |\phi, \phi \oplus \psi\rangle$ where $\oplus$ is addition-modulo 2 (the definition of the classical XOR gate). The CNOT gate is described by the following matrix

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.2.6}$$

Measurement

Measurement is not strictly a quantum-logic-gate as it permanently alters the state of a qubit (it is not a unitary operation). However, it is included in quantum circuit diagrams as it is necessary for useful computation. One may choose to think

of qubit-measurement as a gate with a single qubit input and single classical bit output. Furthermore, if qubits are entangled with each other, the measurement of a single qubit will reveal the state of other qubits simultaneously [66].

2.2.3 Notable Algorithms

In the three decades following the original definition of a gate-based quantum computer a vast number of surprising discoveries have been made and the physical implementation of useful universal quantum computers impend on the near future. While we do not present a summary of the entire field here, we summarise a selection of the most well-known quantum algorithms.

The Deutsch-Jozsa Algorithm

Deutsch and Jozsa [24] define the first quantum algorithm to find a solution more efficiently than any classical computer. While not directly practical this algorithm provides inspiration for more sophisticated algorithms, serving an excellent introduction to quantum algorithms. Suppose we have a function f which accepts a single n-bit number from the range $[0, 2^n - 1]$ as input and produces either zero or one as an output. Further, this function is guaranteed to either be *constant* meaning it returns the same value for all inputs or *balanced* where exactly half of all inputs produce zero and the other half produce one. Using classical computers a deterministic algorithm requires $\frac{2^n}{2} + 1$ queries to reach an answer. The original Deutsch-Jozsa algorithm [24] requires only two function evaluations to compute an answer and is deterministic. Later, Cleve et al. [21] improve the Deutsche-Jozsa algorithm to only require a single query yet remain deterministic. This is clearly a vast improvement over a deterministic classical computer and still a sizeable improvement over stochastic classical algorithms [66]. Nielsen and Chuang [66] provide an excellent summary of this algorithm and its physical implementation.

Shor's Algorithm for Integer Factorisation

We begin by defining the integer factorisation problem which is equivalent to the discrete-logarithm and order-finding problems.

Definition 2.2.2. Given a positive composite integer N, what prime numbers when multiplied together produce N? [66]

Integer factorisation underpins public-key encryption systems widely used today. The exponential growth in classical complexity involved with factorising large

primes have provided security for decades [15]. Peter Shor [78] proposes a quantum algorithm to solve this problem with exponential speed-up over classical computers. The asymptotic run-time of Shor's algorithm grows polynomially with the length of the integer to be factored. Computing the quantum Fourier transform, a quantum analogue to the well established discrete Fourier transform, is core to this algorithm and is inspired by the Detsch-Jozsa algorithm [66]. Shor's algorithm is not deterministic and may require a polynomial number of repeat computations to produce a correct solution with high-probability [77]. Shor's algorithm has thus produced the burgeoning field of quantum cryptanalysis. Cleve et al. [21] present a method to break the well-known RSA cryptography scheme explicitly.
Quantum cryptography is becoming a more prominent field spawning a large volume of research and commercial development. In the near-term, quantum key distribution schemes face rapid progress and immanent implementation [15]. In addition to security concerns, quantum computers pose a threat to relatively novel concepts such as crypto-currencies prompting careful threat analysis [2]. Shor's algorithm to this day remains arguably the most infamous quantum algorithm.

Grover's Search

Unstructured search is a very general problem which is defined simply. Given a finite set of possibilities, find options which satisfy a particular condition. In most practical contexts there exists structure in the search space which is exploited to design efficient algorithms; consider a binary search on a sorted list for example. Grover [35] presents a quantum algorithm for unstructured search with time complexity $\mathcal{O}(\sqrt{N})$. Bennett et al. [8] later show the lower limit of time complexity for this task using a quantum computer is $\Omega(\sqrt{N})$.
Grover's search algorithm starts by preparing a superposition of n-qubits representing $2^n = N$ possible items. In this state, some number M of the N possible items will correspond to satisfactory elements and the rest will not. The vector sum over all desirable elements will produce a basis vector $|\alpha\rangle$ and a sum over all other unsatisfactory elements produce an orthogonal basis vector $|\beta\rangle$. In this new basis, a state with a high amplitude in the $|\alpha\rangle$ axis corresponds to a high probability of measuring a satisfactory item. A subroutine termed a 'Grover iteration' is performs a rotation in the $|\alpha\rangle, |\beta\rangle$ basis in the $|\alpha\rangle$ direction and is applied $\mathcal{O}(\sqrt{N})$ times. Specifically the number of iterations is approximately $\pi\sqrt{N}/4$ when searching for a single item [36]. After this a measurement is made revealing with high-probability an item satisfying our query [66]. The exact number of iterations is dependent on each problem instance growing polynomially and optimally [36]. The optimal nature of Grover's algorithm is a surprising result motivating further research into quantum computing. Exploiting the exponential scale of quantum

information remains a central goal as quantum algorithms are sought to solve previously intractable classical problems.

2.2.4 Quantum Supremacy

Quantum supremacy describes the potential ability for quantum computers to outperform classical computers for some problems. This problem fundamental to the field of quantum computing is difficult to demonstrate for a number of reasons. The performance of a quantum computer must be proven to be superior to any classical computer requiring rigorous proofs of lower-bound complexity for both quantum and classical formulations of a problem. Boixo et al. [11] suggest the construction of particular problems simulating quantum phenomenon to aid in this endeavour. Relaxing the definition of quantum supremacy permit the use of benchmarking and practical performance as measures of supremacy inspired by the evaluation of heuristic-based algorithms in classical computing. Formal supremacy is an important milestone in the field, however useful quantum computation is the true goal of the field.

2.2.5 Classical Quantum Simulation

The obvious approach to establish supremacy is to simulate quantum circuits of increasing size. The point at which simulation becomes intractable reveals a lower limit on formal quantum supremacy. Such a bound is difficult to show analytically resulting in the some of the largest single-task computations in history.
The recently developed high-performance distributed quantum simulator *qHiPSTER* [81] allows for simulation of quantum circuits up to 40-qubits in scale. Smelyanskiy et al. [81] find memory to be the limiting factor of simulation postulating that simulations greater than 49 qubits are in-feasible until 2024.
Boixo et al. [11] build on the work of Smelyanskiy [81] by simulating 42-qubit circuits using 70 terabytes of memory. Further Boixo et al. [11] formalise the task to demonstrate supremacy based on building very dense, partially randomised circuits acting upon a grid of qubits. This introduces both size and depth as simulation bounds. Importantly, such a measure is shown to be efficiently measured on a hypothetical physical quantum processor. The scheme is based on multiple fast evaluations of a circuit revealing a single sampled output. Over a vast number of samples an accurate distribution is achieved.
Häner and Steiger [46] simulate a 45-qubit system. Deriving improvements in memory and communication overheads by kernel optimisation and a scheduling

algorithm ordering the processing of sub-circuits. This simulation, the largest of its time, required 8,192 nodes and 0.5 petabytes of memory.
Pednault et al. [69] provide methods to simulate beyond the previously conceived 49-qubit limit using only three terabytes of memory simulating 56-qubit circuits. The scheme employed by Pednault et al. [69] reformulates circuit simulation as tensor operations cutting down on memory requirements significantly and allows the use of more generalised tensor mathematics. Similar to the scheduling concept used by Häner and Steiger [46], the computation of gates which entangle qubits (introducing an exponential factor) is deferred.
Boixo et al. [12] further improve their original scheme [11] formulating circuit execution as an un-directed graph. A variable elimination scheme is developed reducing average memory requirements. This scheme is especially powerful for smaller circuits allowing workstation simulation of a larger scale than previously possible.
Chen et al. [17] apply more aggressive gate partitioning producing exponentially more independent circuits to simulate, allowing better use of distributed resources. Further, Chen et al. [17] estimate the computational cost of simulating a 72-qubit circuit, deeming it feasible for a computer identical to that used by Pednault et al [69].
Li et al. [58] compute both sampling and full simulation tasks for circuits of 49-qubits at 39 and 55-depths respectively. A gate partitioning scheme in addition to dynamic programming methods are used to construct an efficient ordering of sub-tasks reducing memory overheads. 131,072 nodes and nearly one petabyte of memory are used.
Chen et al. [17] extend the variable elimination work of Boixo et al. [12] and apply it in a distributed manner. Circuits of varying sizes and depths are analysed factoring estimated real-world noise with the intent to derive a lower-bound on hardware accuracy. Again, 131,072 nodes and around one petabyte of memory are used.
Markov et al. [60] refine the benchmarks defined by Boixo et al. [12] further increasing classical simulation complexity. The use of public cloud resources allow Markov et al. [60] to associate a monetary cost with such simulations generating further motivation for implementing such a scheme in quantum hardware.

Quantum computing brings focus to two frontiers; the fundamentals of computing as we gain an understanding of qubits, and cutting-edge classical simulation. The few algorithms already discovered bring large implications to the world of computer science encouraging research into finding quantum advantage in increasingly challenging and diverse classical problems. We introduce one such algorithm in the following section designed to make use of both quantum and classical machines

in an effort to realise practical quantum computation sooner.

2.3 The Quantum Approximate Optimisation Algorithm (QAOA)

In section 1.4 we introduce the algorithm itself, here we discuss prior investigation and experimentation.

2.3.1 The QAOA and Quantum Supremacy

The requirement that a quantum supreme algorithm must exhibit performance superior to any classical algorithm is difficult to formulate. The ultimate goal to implement a quantum supreme algorithm on physical hardware remains an open yet vital problem. Farhi et al. [29] present an analysis of the QAOA applied to the E3LIN2 problem, a linear equation optimisation intending to demonstrate provable supremacy. Taking $p = 1$ Farhi et al. [29] provide an analytic formulation to show their result. The QAOA produces answers satisfying $\frac{1}{2} + \Omega(D^{\frac{-3}{4}})$ of the required clauses.
Spurred by this claim, Barak et al. [7] present a superior classical algorithm for the same problem satisfying $\frac{1}{2} + \Omega(\frac{1}{\sqrt{D}})$ fraction of the required clauses.
However, the formulation of the QAOA examined by Farhi et al. [29] is a coarse approximation of the QAA using only a single trotterisation ($p = 1$). Fahri et al. [29] suggest a number of possible improvements requiring further analysis such as increasing p and introducing variables for each clause to be optimised. Such improvements are difficult to formulate analytically and hence experimental motivation for such analysis is required to justify such work.
For these reasons Farhi and Harrow [30] propose the QAOA may still demonstrate quantum supremacy. The QAOA may demonstrate quantum supremacy in two ways.
Farhi and Harrow [30] argue the inherent quantum nature of the QAOA itself cannot be replicated classically. More specifically if there did exist such an algorithm, Farhi and Harrow [30] propose the complexity hierarchy would collapse. Secondly, physical quantum computers will allow the QAOA to be run on problem instances prohibitively large for classical computation and hence may generate superior solutions in these instances. This provides evidence for the QAOA to be among the first algorithms implemented in quantum hardware despite classical competition to find superior algorithms.

2.3.2 Optimisation of the Classical Component

The hybrid nature of the QAOA naturally leads to two frontiers of development and research, the quantum and classical components. A large proportion of efforts understandably focus on the quantum component an understanding of the classical component is essential. Guerreschi and Smelyanskiy [37] investigate three classical optimisation methods for hybrid quantum algorithms with experimental analysis on the QAOA. Gradient-free and quasi-Newton methods are investigated in an experimental manner. The Nelder-Mead algorithm for gradient-free optimisation is an appropriate method for the general case of a small value of p while the quasi-Newton method using finite derivative methods provides superior results with a matching increase in implementation complexity [37]. The function space the QAOA generates is typically very difficult to form a gradient in, hence the increase in computational complexity. The work of Guerreschi and Smelyanskiy [37] lays a solid foundation for experimental simulation of the QAOA with regards to complexity and performance.

2.3.3 Extensions of the QAOA

Augmentation and extension of the QAOA is possible in addition to direct optimisation. Instead of solving optimisation problems directly Wecker et al. [85] modify the QAOA to find a quantum state which seeks a maximal overlap between the objective function and the ground energy state of the given instance. This approach is applied to the MAX 2-SAT problem. This change in formulation should lead to more accurate results but introduces more complexity into the classical optimisation. The general approach Wecker et al. [85] employ uses classical machine learning techniques to train the algorithm with known difficult instances of the MAX 2-SAT problem. The training yields an optimisation schedule describing how the algorithm explores the state space of possible parameter-values and is subsequently tested on a another set of problem instances. This machine learning approach solves instances of MAX 2-SAT and MAX 3-SAT faster than the well known annealing scheme CFLLS [22], demonstrating the largest improvement in the hardest instances. This experimental approach provides concrete data suggesting this modified QAOA is also suitable for near-term implementation. However, a generalisation of this approach and the specific learning methods employed are not discussed in detail. Additionally, performance comparison to the original formulation of the QAOA is not presented.

Rather than adjusting the main objective of the QAOA, Hadfield et al. [39] extend the QAOA to a more general framework termed the Quantum Alternating

Operator Ansatz (Referred to as QAOA in the original paper out of respect but here as the QAOAn to avoid confusion). The main modification considers general parameterised families of unitaries over the specific family of fixed local Hamiltonians. Loosely speaking this allows the QAOAn to operate on registers describing a wider range of quantum systems which in turn allows the encoding of more varied problems. This is accomplished by varying the formulation of the mixing operator $\hat{U}_B$ to restrict considered bit-strings to valid solutions only. This decomposes the mixing operator into a number of operations which makes the algorithm more powerful but more difficult to implement. Hadfield [38] describes a large number of problem specific formulations of the QAOAn including very well known hard classical problems such as the travelling salesman and job scheduling to demonstrate the potential impact of this reformulation of the QAOA. Unlike Farhi and Harrow's [30] original approach to demonstrating quantum supremacy analytically, Hadfield et al. [39] propose an empirical approach similar to how heuristic algorithms are often analysed. Their reasoning cites the difficulty often found when proving the supremacy of heuristic algorithms directly versus the easier task of bench-marking an algorithm over a suitable well-known set of problem instances. This approach is similar to that of Wecker et al. [85]. Disappointingly no experimental data is presented, however this work provides a solid foundation for future investigation and publications of the QAOAn due to the large number of problem specifications provided.
Mash and Wang [61] propose a more tightly describe alteration to the QAOA for NPO PB problems based on the observation that the mixing operator describes a continuous time quantum walk on the quantum register. By imposing restrictions on this operator bit-strings are partitioned into feasible and in-feasible sets allowing for greater performance.

2.4 Computing the Matrix Exponential

Simulating the QAOA requires solving the time-dependent Schrödinger equation

$$|\Psi(t)\rangle = e^{-iHt} |\Psi(0)\rangle , \tag{2.4.1}$$

as a central component of the algorithm. Efficient and accurate numerical approximate of Equation 2.4.1 requires efficient and accurate computation of the matrix exponential for large, sparse and complex-valued matrices. We define the matrix exponential for completeness.

Definition 2.4.1. $exp(A) \equiv e^A = \sum_{n=0}^{\infty} \frac{A^n}{n!} = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + ...,$

Matrix exponential computation is a highly investigated problem with over 35 years of investigation. Despite these efforts there is no single superior method,

rather an array of methods each with their own intricacies, benefits and shortcomings. Moler and Van Loan [62] present a now canonical review of 19 candidate methods. Their work remains so influential Moler and Van Loan [63] present an updated version 35 years later. Careful algorithm selection and implementation is key to guarantee both performance and numerical accuracy. Such choices are highlighted by implementation for HPC users.
We note that for the diagonal case, computing the matrix exponential involves exponentiation each element of the matrix [62]. Through eigenvalue decomposition one can diagonalise most matrices reducing the exponential to this simpler case in addition to two matrix-matrix multiplications. However, such a method requires the use of sophisticated eigenvalue solvers, a major computational effort in of itself which may be slower than many other methods in the general case.
Computing the Taylor series directly results in a slow-convergence and low-accuracy in the general case. Using a Padé approximation provides better accuracy with less terms, however again, naïve application of series expansion results in poor general-case performance. As such more sophisticated methods provide superior performance and are necessary for practical use.

Scaling and Squaring

The most popular method available for dense matrices is scaling and squaring. This method relies on a property unique to the matrix exponential

$$e^A = (e^{A/m})^m. \tag{2.4.2}$$

Selecting m carefully as the smallest power of two such that $\|A\|/m \leq 1$ allows for accurate and efficient use of Taylor or Padé approximants. Scaling and Squaring is among the most widely used methods due to strong accuracy and elegance. Al-Mohy and Higham [3] champion this error presenting highly in-depth error analysis and precise algorithms for computing optimal m for IEEE precision arithmetic. Further Higham and Tisseur [45] present an algorithm for estimating the 1-norm of arbitrary matrices, a key component of the aforementioned matrix exponential algorithm [3]. Scaling and squaring is widely implemented in many commercial packages such as MATLAB, Scipy, Mathematica and Expokit [79]. Scaling and squaring is best suited for dense matrix exponentiation and is thus poorly suited for distributed implementation due to the use of matrix-matrix products.

2.4.1 Computing $e^A \cdot \mathbf{v}$

In many cases including our own, the computation of the action is the matrix exponential on a vector is our task. This slightly different problem allows for

alternative methods to be used with potentially less computational overhead.

Scaling and Squaring

Higham and Al-Mohy [4] present a method based on their algorithm [3] for computing the action of the matrix exponential. Now their algorithm determines through one-norm estimation the optimal scaling value to minimise the number matrix-vector multiplications required. Aside from the estimation of the one-norm this algorithm is currently untested in a distributed memory implementation.

Krylov Subspace

The most popular method for large, sparse matrix exponentiation is the Krylov subspace method [63]. This method approximates the matrix exponential onto a smaller Kylov subspace which then allow for dense matrix methods to be applies efficiently. Re-using the constructed subspace allows successive value of t to be computed at low-cost and as such is considered the canonical sparse-matrix method. Mathamatica's `MatrixExp[A,v]`, Expokit [79] and SlepC/PetSc [6], [43] implement the Krylov subspace method. Furthermore, SlepC/PetSc offer the only commercially available distributed memory implementation of the matrix exponential.

Chebyshev Approximation

The Chebyshev approximation method spawns from quantum chemistry where a series approximation of the matrix exponential is computed by the Chebyshev polynomial forming each step [28, 84, 83, 64]. Post-multiplying the Chebyshev series with our vector v allows for direct computation of $e^{tA} \cdot \mathbf{v}$ without ever computing a full exponential matrix. Bessel J zero functions form the coefficients of the expansion which allow for fast and accurate convergence. Chebyshev approximation requires either eigenvalue scaling or use of the dense scaling and squaring method. However estimates of the eigenvalues do not effect accuracy greatly, but effects the number of iterations required for convergence [49]. The Chebyshev method is appealing for HPC applications has only matrix-vector or vector-vector operations are required when eigenvalue scaling is used. This allows for trivial memory parallelisation with minimal communication making this method a strong candidate growing support in HPC applications. Furthermore Auckenthaler et al. demonstrate that the Chebyshev method is superior to the scaling and squaring method [5] while Bergmaschi et al. [9] suggest the Chebyshev method is superior

to the Krylov subspace method. Despite practical performance, very few packages implement this method: Expokit [79] implements Chebyshev approximation for dense matrices, and pyCTQW [49] a Python package built upon PetSc/SlepC to simulate continuous time quantum walks.
Computing the action of the matrix exponential is critical to efficient and accurate simulation of the QAOA at desktop to cluster-scale.

2.5 Quantum Computing Applied to Graph Similarity Problems

We are not the first to consider applying quantum computing to classically difficult graph theoretic problems. Lucas [59] provides a vast number of mappings for classical NP-Complete problems to the Ising model of computing. The Ising model of computation can in turn be mapped onto a quantum annealer through the quantum adiabatic algorithm (QAA)[32]. Hen and Young [42] map the graph isomorphism problem to a quantum annealer with experimental results. Hen and Young analyse experimental implementation supporting the conjecture that quantum annealers can discriminate between non-isomorphic graphs [42]. Furthermore they suggest that hardware and simulation improvements will better validate their claims.

Graph similarity is an openly difficult problem to compute classically despite the vast practical use it demonstrates. As the field of quantum computing matures historically intractable problems are explored with often surprising results expanding the scope of what feasible computing permits. We also provide a brief introduction to quantum computing assuming no prior knowledge of quantum physics in addition to a few historic quantum algorithms. We present the Quantum Approximate Optimisation Algorithm as a general method to approach hard combinatorial optimisation problems; optimisation and extensions to the algorithm are an open area of research. Seeking the limits of computation will always be integral to the field of computer science, an endeavour extending into the realm of quantum computing.

CHAPTER 3

Methods

3.1 The Quantum Approximate Optimisation Algorithm (QAOA)

The Quantum Approximate Optimisation Algorithm (QAOA) stands unique among many other quantum algorithms as it is in essence, a Monte-Carlo Algorithm. The solution quality various for a fixed amount of execution but has in practice shown excellent performance inspiring academic and industrial investigation [67]. To encode a problem into the QAOA we require the following.

- A problem Hamiltonian $\hat{C}$ which implements the cost function of our candidate problem
- A mixing Hamiltonian $\hat{B}$ which defines which bit-strings are permitted for evaluation by the algorithm.
- A suitable amount of decomposition (p value)
- An initial state generation scheme

A QAOA iteration is defined as p applications of successive $\hat{U}_C$ and $\hat{U}_B$ unitary operators. At each application of $\hat{U}_C, \hat{U}_B$ a corresponding corrective parameter γ_i, β_i is applied to approximate an annealing scheme while dropping the adiabatic requirement of the QAA. Repeated sampling generates an approximate expectation value which is fed into a parameter optimisation scheme to select new $(\vec{\gamma}, \vec{\beta})$. This process repeats until a termination criteria is met or the quantum compute time is exhausted.
Hadfield [38] and Marsh and Wang [61] introduce restrictions upon the mixing operator restricting the algorithm to feasible solutions at a cost of higher gate depth. Marsh and Wang [61] reverse the order of applying $\hat{U}_C$ and $\hat{U}_B$ in order to account for the non-trivial maximal energy state introduced by applying mixing restrictions. We provide a novel mapping of permutation based problems such as graph similarity via edge-overlap to the QAOA.

3.1.1 General Problem Encoding

Problem operators are encoded into annealing schemes such as the QAA through the Ising spin-glass model of computation [74] which allows the definition of problem Hamiltonians of the following form.

$$\hat{H} = \mu \sum_{i} h_i \sigma_i + \sum_{i,j} \sigma_i \sigma_j. \tag{3.1.1}$$

Problems are encoded using pseudo-boolean functions [13] and permit a wide variety of difficult problem definition [59]. The first term defines the logic for setting a particular spin $x : \{-1, 1\}$. In the gate-based QAOA each 'spin' corresponds to a qubit encoding a binary variable $x : \{0, 1\}$.

3.2 Graph Similarity via QAOA

3.2.1 A Canonical Mapping

The standard method to map permutation based problem use unary encodings of n^2 qubits. To illustrate, consider for an arbitrary pair of graphs each consisting of V vertices a V^2 string of qubits

$$[x_{1,1}x_{1,2}\dots x_{1,v}][x_{2,1}x_{2,2}\dots x_{2,v}]\dots[x_{v,1}x_{v,2}\dots x_{v,v}], \tag{3.2.1}$$

where a binary variable $x_{i,j}$ represents vertex j in G_2 mapping to vertex i in G_1. Under such a scheme the vast majority of the 2^{V^2} bit-strings are not feasible since $n! << 2^{n^2}$. Such an encoding quickly becomes intractable for both simulation and physical quantum hardware where the current state of the art is around 72 qubits [60] permitting graphs of less than eight vertices. Hadfield [38] provides a QAOA mapping for the travelling salesman and other permutation problems but requires mixing constraints to enforce legal candidate solutions. A unary encoding of graph similarity through edge overlap suffers from the same issue.

3.2.2 Defining $\hat{C}$

We propse a novel compact encoding requiring $\mathcal{O}(\lceil log_2(V!)\rceil)$ qubits prepared in $\mathcal{O}(V^3)$ operations. We define our problem Hamiltonian as

$$\hat{C} = A \sum_{\sigma \in V!} \sum_{i=1}^{V} \sum_{j=1}^{V} \sum_{u=1}^{V} \sum_{v=1}^{V} (d_1(i,j) d_2(u,v))(x_{i\sigma(u)} x_{j\sigma(v)}). \tag{3.2.2}$$

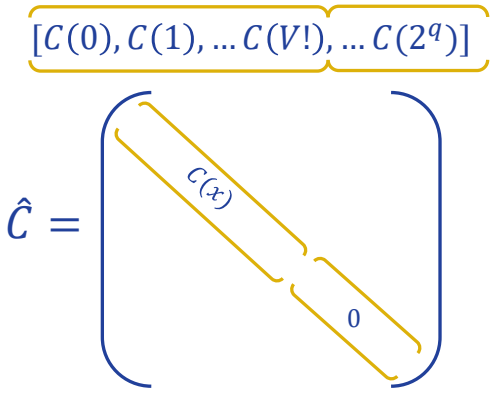


Figure 3.1: $\hat{C}$

This provides us with the edge overlap for all $V!$ permuted labels of $u, v \in G_2$. Our problem operator $\hat{C}\,|\psi\rangle = C(x)\,|\psi\rangle$ that is the application of our cost function to all bit-strings x. Our diagonal problem operator $\hat{U}_C$ becomes

$$\hat{U}_C(\gamma)\,|\psi\rangle = e^{-i\gamma c(x)}\,|\psi\rangle\,. \tag{3.2.3}$$

Welch et al. [87] builds on the work of Childs [19] providing a method to implement $e^{-i\gamma\hat{C}}$ efficiently without the use of additional ancillary qubits where each element of the diagonal $\hat{C}$ is itself efficiently computable. We provide a graphical representation of $\hat{C}$ in Figure 3.1

3.2.3 Defining $\hat{B}$

We define a canonical mixing operator [31]

$$\hat{B} = \sum_{i=1}^{n} \sigma_i^x, \tag{3.2.4}$$

where σ^x is the Pauli-x matrix, the quantum equivalent of the NOT gate. Our mixing operator $\hat{U}_B$ becomes

$$\hat{U}_B(\beta)\,|\psi\rangle = e^{-i\beta\sum_{i=1}^{n}\sigma_i^x}\,|\psi\rangle = e^{-i\beta\hat{B}}\,|\psi\rangle\,. \tag{3.2.5}$$

We provide a graphical representation of $\hat{B}$ in Figure 3.2.

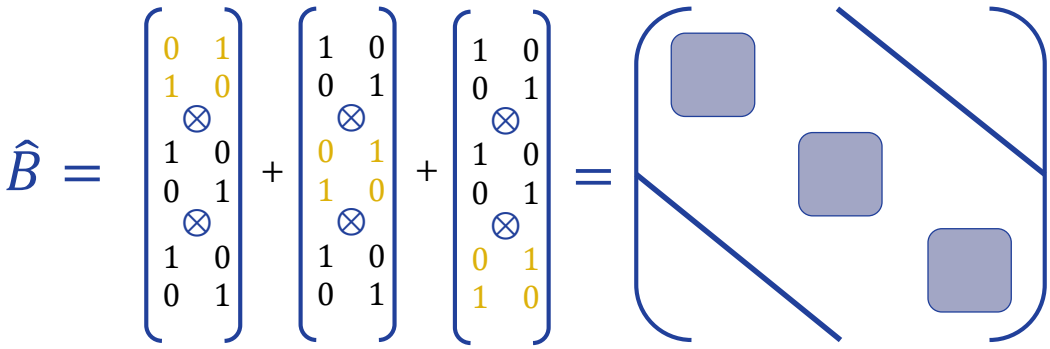


Figure 3.2: $\hat{B}$

3.2.4 Permutation Mapping

Abrams and Lloyd [1] provide a method to encode a superposition of n elements using $\mathcal{O}(n^3)$ operations. Chiew et al. [18] provide a full circuit encoding edge overlap values using $\mathcal{O}(n^2 log^2(n))$ operations and $\mathcal{O}(nlog(n))$ qubits allowing efficient bit-string mapping over the range $[0, n!]$ to edge-overlap values. To index all such values in $\hat{U}_C$ we require $\lceil log_2(V!) \rceil$ qubits (termed q) since for all $n > 2, n! > 2^n$. We trivially map the remaining 'tail' values not included in the range 2^q to zero. Typically the initial state of a QAOA iteration is the poorest possible solution. In our case this difficult to determine thus we use a superposition of all q qubits. This compact representation suffers from these 'tail' values. We plot the first 2000 terms in Figure 3.3. These degenerate values are frustrating from an optimisation standpoint since the QAOA will always start from an equal probability of measuring any bit-string. We cannot apply per-qubit mixing restrictions. The proportion of the tail reveals a uniform distribution (Kolmogorov-Smirnov test [53] for uniform distribution, p-value $1.46e-08$). In Figure 3.3, a DB-clustering [53] of points confirm this observation further, labelling all points as noise. For interest we present a table of initial data points and present a series expansion of this phenomena in Appendix B.

3.2.5 Test-Case Generation

Graphs are generated using the standard Erdös-Rényi method; we randomly assign each edge with 50% probability. Instead of testing two random graphs, we deform an initial graph in order to create difficult test-cases where graphs are quite similar.

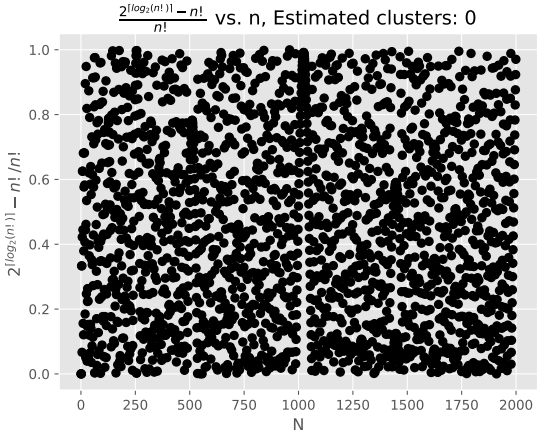


Figure 3.3: The proportion of infeasible to feasible bit-strings

This allows us to test for overall correctness for each individual trial and over multiple trials. We consider the following deformations

- Isomorphism: The original graph is compared to itself. The first element of the problem Hamiltonian is guaranteed to be zero.
- Vertical Flipping: The original graph is mirrored in the vertical axis. This generates very dissimilar graphs with minimal effort shown in Figure 3.4
- Edge Addition: v non-existent edges are added at random
- Edge Removal: v existent edges are removed at random
- Edge Addition and Removal: v edges are added and removed randomly

3.3 Simulation Design

Qolab (Quantum Optimisation Laboratory) is a flexible simulation package for the QAOA and related algorithms from desktop to cluster scale implemented in C. Our package allows for a single interface to both desktop and cluster-based code at the highest level of abstraction possible. The user is required to implement only the cost function used to define $\hat{U}_C$ and any walk masks applied to $\hat{U}_B$ if using the modified QAOA described by Marsh and Wang [61]. We present an extensive description in Appendix C. Exacting implementation targets maximal single node and desktop performance. Multiple processors are utilised by distributing the

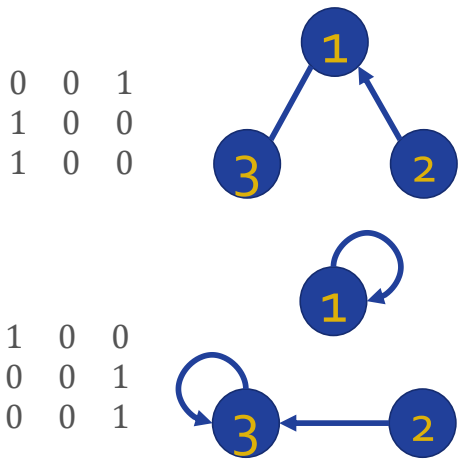


Figure 3.4: Adjacency Flip

state-space.
Matrix operations are handled using Intel's Math Kernel Library [48] and we store matrices in compressed column form. [27].

3.3.1 Simulation Components

Direct state-vector simluation avoids a large amount of overhead present in contemporary gate-based simulations such as IBM's Qiskit Alpha [47]. We capture the high-level structure of our QAOA simulation in Algorithm 2 and a similar description of a walk-restrictive QAOA in Appendix D. The following sections describe the implementation and computational complexity of each component. For clarity we make the following definitions:

- V - The number of vertices in candidate graphs
- q - A number of qubits. For graph similarity this is $2^{\lceil log_2(V!) \rceil}$
- n - A number of classical items
- p - The amount of trotterisation applied to the QAOA
- P - The number of processors present in our cluster
- n_P - The size of the state-vector maintained by each processor
- $|\psi\rangle$ - The state-vector used by the QAOA

Algorithm 2 Qolab Overview

procedure QAOA_Core(numQubits, P, optimisationMethod, C(x))
    $\hat{U}_C \leftarrow$**genUC**(C(x))
    $\hat{U}_B \leftarrow$**genUB**(numQubits)
    $\vec{\gamma}, \vec{\beta} \leftarrow$**initialParameters()**
    while terminateTest() **do**
        $|\psi\rangle \leftarrow$**initialState**
        for $i = 0$ to p **do**
            $|\psi\rangle \leftarrow \hat{U}_C(\gamma_i)\,|\psi\rangle$
            $|\psi\rangle \leftarrow \hat{U}_B(\beta_i)\,|\psi\rangle$
        end for
        $F_p \leftarrow$ **Measure**(S)
        $\vec{\gamma}, \vec{\beta} \leftarrow$ **updateParameters**(F_P)
    end while
end procedure
Report()

3.4 Generating $\hat{U}_C$

3.4.1 Permutation Generation

The power of the QAOA derives performance from the ability to process an exponential number of inputs to our cost function simultaneously. This requires the generation of all $q!$ bit-strings and application to the provided cost function $C(x)$. The time complexity is $\mathcal{O}(q!)$. We are able to generate the k-th permutation through a Lehmer code based algorithm. The time complexity to generate all permutations is $\mathcal{O}(log_2(n!))$. Pseudocode is presented in Appendix D. Heap's algorithm has ldong been considered the fastest method to generate all permutations of n elements[75]. Our sub-optimal k-th based permutation scheme allows for independent generation of permutations across distributed processes achieving optimal $\mathcal{O}(n!/P)$ work per process. Further this scheme matches the order generated by our theoretical QAOA encoding.
Nevertheless, since we require a $2^{\lceil log_2(v!) \rceil}$ state-space generations of the $q!$ permutations are reduced in the asymptotic complexity.

3.4.2 Generating $\hat{C}$

The cost function operator $\hat{U}_C$ is generated by evaluating the provided cost function for all permutations of q bits resulting in a $2^q \times 2^q$ diagonal matrix called $\hat{C}$. The time complexity to build $\hat{C}$ is $\mathcal{O}(q! \times Poly(q))$ where $Poly(q)$ is the cost-function itself. For edge-overlap this is $\mathcal{O}(V^2)$. There is little need to parallelise the cost function as it is computationally dominated by the number of bit-string considered.

3.5 Generating $\hat{U}_B$

$\hat{U}_B$ is less trivial to simulate. The definition of $\hat{B}$ given by Equation 1.4.5 can be re-phrased with respect to matrix elements directly. The operator defines valid transitions from bit-string to bit-string.

Without mixing restrictions the original definition connects all bit-strings which differ by a single element. A naïve implementation computes the original $\hat{B}$ matrix directly by performing a sequence of Kronecker products. However under this new observation the generation method becomes trivially distributable. We present pseudocode in Algorithm 3 which includes the application of walk masks described by Marsh and Wang [61]. When no masks are applied the generated $2^q \times 2^q$ matrix matches that defined in Equation 1.4.5 which represents a maximally connected hypercube between candidate bit-strings.

3.6 Distribution Scheme

The distribution of our simulation is based on the state-vector itself. We initially consider a scheme where each process is responsible for a unique sub-set of the 2^q elements defined as

$$n_P = \lfloor \frac{2^q}{P} \rfloor. \tag{3.6.1}$$

This decomposes seamlessly where $log_2(P) \in \mathcal{Z}$ since the state-space grows exponentially. When this is not true we append the remaining items to the final process. The number of appended items is defined as

$$n_{Pend} = 2^q mod P, \tag{3.6.2}$$

which by definition grows $\mathcal{O}(P)$. Since $P \ll 2^q$ we deem this acceptable. This decomposition has the benefit of optimal load balancing between all processes in the general case. This simple method works well for distributing the state-vector and realising $\hat{U}_C$ however $\hat{U}_B$ requires more nuance.

Algorithm 3 $\hat{B}$ Generation

```
1: procedure GenerateB(numQubits, mask)
2:     UB ← ∅
3:     nnz = 0
4:     for i ← 0 to 2^q do
5:         colB[i] ← nnz
6:         for j ← 0 to 2^q do
7:             row ← i ∨ (1 << j)
8:             if mask(row) then
9:                 values[nnz] ← 1
10:                colE[i] ← nnz
11:                rowInd[nnz] ← col
12:                nnz ← nnz + 1
13:            end if
14:        end for
15:    end for
16: end procedure
17: Report()
```

3.7 Function Evaluation

Function evaluation method realises a single QAOA iteration defined in Equation 1.4.7. The generation of an initial state is in our case an equal superposition of q qubits. This is represented as a 2^q vector of $\frac{1}{\sqrt{2^q}}$ indicating an equal chance of measuring any candidate bit-string. The state-vector $|\psi\rangle$ is distributed equally across all processes according to Equation 3.6.1. We describe each part of the function evaluation separately.

3.7.1 Setup

The root node holds the set of parameters $(\vec{\gamma}, \vec{\beta})$ broadcast to all processes. The communication overhead grows $\mathcal{O}(p)$.

3.7.2 Applying $\hat{U}_C$

We need to realise the action

$$e^{-i\gamma\hat{C}} |\psi\rangle . \tag{3.7.1}$$

Since $\hat{C}$ is a diagonal matrix, we can compute $e^{-i\gamma\hat{C}}$ by exponentiating each element of $\hat{C}$ and apply a point-wise multiplication with $|\psi\rangle$ or dot-product with $|\psi\rangle^T$. This requires $\mathcal{O}(2^q)$ operations but is embarrassingly parallel. We make use of vectorisation and parallelisation afforded by multi-core processors.

3.7.3 Applying $\hat{U}_B$

We need to realise the action

$$e^{-i\beta\hat{B}}\,|\psi\rangle\,. \tag{3.7.2}$$

The non-trivial structure of the $\hat{B}$ matrix requires a sophisticated method to compute the action of the matrix exponential; a difficult problem withstanding four decades of continual investigation. We implement the Chebyshev expansion method similar to pyCTQW [49] and depicted in Equation 3.7.3. This method requires only matrix-vector operations to realise Equation 3.7.2 without storing the final exponentiated matrix at any point. In general,

$$e^{tA} = e^{(\lambda_{max}+\lambda_{min})t/2}[J_0(\alpha)\phi_0(\tilde{A}) + 2\sum_{n=1}^{\inf} i^n J_n(\alpha)\phi_n(\tilde{A})], \tag{3.7.3}$$

where $\lambda_{max}, \lambda_{min} \in \mathbb{C}$ are eigenvalues of A with maximal and minimal real parts. $\alpha = i(\lambda_{min} - \lambda_{max})t/2$ and $\phi(\tilde{A})$ are the Chebyshev polynomials which computed recursively as

$$\phi_0(\tilde{A}) = I, \tag{3.7.4}$$

$$\phi_1(\tilde{A}) = \tilde{A}, \tag{3.7.5}$$

$$\phi_n(\tilde{A}) = 2\tilde{A}\phi_{n-1}(\tilde{A}) - \phi_{n-2}(\tilde{A}). \tag{3.7.6}$$

Similar to other approximation methods normalisation of $\lambda \in [-1, 1]$ encourages minimal convergence time and therefore we scale our matrix

$$\tilde{A} = \frac{2A - (\lambda_{max} + \lambda_{min})I}{\lambda_{max} - \lambda_{min}}. \tag{3.7.7}$$

The use of Bessel function zeros as coefficients means that $J_n(\alpha) \approx 0$ when $n > |\alpha|$ and therefore convergence occurs after $|\alpha| \propto t$ terms. Similar to Izaac and Wang [49] we terminate after the condition

$$|2J_n(\alpha) \leq \epsilon|, \tag{3.7.8}$$

where ϵ is chosen to be 10^{-18}.
However, without knowledge of the maximal and minimal eigenvalues we would

be required to solve for these values, a time-consuming and laborious calculation for large matrices. Since we are computing the matrix exponential for only a particular type of matrix (non-negative, Hermitian and symmetric) we find that

$$\lambda_{min,max} = \pm q. \tag{3.7.9}$$

A derivation of Equation 3.7.9 is found in Appendix E. Furthermore, the only downside to over-estimating the range of these eigenvalues is computation time, not accuracy. In the case of a restricted $\hat{B}$ matrix the symmetric, Hermitian and non-negative properties hold and can only have fewer non-zero elements than a fully connected, canonical $\hat{B}$ hence our method is appropriate for simulation of the QAOA.
When considering the decomposition of $\hat{B}$ among multiple processes we consider three options. In all cases the total work required is $O(n^2/p)$ since all values must be considered in the multiplication.

Column Distribution

Each process builds and operates upon a section of columns of our overall matrix governed by Equation 3.6.1. This produces optimal load balancing between all processors due to the symmetry of $\hat{B}$. A matrix vector multiplication sees each process operate upon a subset of the state-vector but produces a full 2^q length vector. These need to be combined across all processes resulting in $\mathcal{O}(n^2/p+n+p)$ work. The difficulty here is that after each multiplication each process contains a full state-vector with $1/p$ of the total solution which must be reduced and scattered to all processes in order to continue work. If we consider α as the time for a single scalar operation, λ as the communication latency and each complex value is 32 bytes long and β as the buffer length of a message the total execution time is given by

$$\Theta(\alpha n\lceil\frac{n}{p}\rceil + (p-1)(\lambda + \frac{32}{p\beta})). \tag{3.7.10}$$

Moreover the scalability of this scheme is n^2p. Since $n = 2^q$ this scheme scales poorly.

Row Distribution

Similarly, we could decompose across rows which again does not alter the distribution scheme of $\hat{C}$. In this scheme, after a matrix-vector multiplication each process will hold a separate section of the resultant vector which again needs redistributing

to all processes through an all-gather operation. The total execution time is given by

$$\Theta(\alpha n\lceil\frac{n}{p}\rceil + \lambda\lceil logp\rceil + \frac{32n}{\beta}). \tag{3.7.11}$$

We see a similar scalability function of n^2p which is again very infeasible but slightly better.

Checker-board Distribution

We could alternatively decompose $\hat{B}$ across both rows and columns forming a grid of processors. The input vector is distributed across the first row of processes and each subsection is copied down each column. After a vector multiplication operation the result is then reduced across each row. A process in each row will contain a separate section of the final state vector prepared for a $\hat{U}_C$ operation. This scheme exhibits a total execution time of

$$\Theta(\frac{\alpha n^2}{p^2} + \lambda\frac{32nlogp^2}{\sqrt{p^2}\beta}), \tag{3.7.12}$$

if we consider the squared number of processes required to implement this method. More machines are required however the scalability is much better and is given by $n^2log^2p^2$.

3.8 Measurement

After our final QAOA state is prepared $|\psi\rangle$ results must be communicated back to the root process. The naïve method would involve each processor sending its final discrete component of the state-vector to the root process. This would require $\mathcal{O}(2^q)$ communication at the root process. We formulate three schemes for final state measurement.

The expectation value can be computed in an embarrassingly parallel fashion on each process and reduced at the root process. This scheme requires $\mathcal{O}(P)$ communication of a single value.

Aggregating over the state-space and cost-function allows us to reduce the entire state-vector which grows $\mathcal{O}(2^q)$ to a smaller distribution of unique cost-function values which is problem-dependent in size. By definition, the QAOA demands a polynomial cost function however and for graph simialrity this is $\mathcal{O}(v^2)$ which is significantly more feasible.

3.9 Optimisation

The well-established nlopt non-linear optimisation library [51] to implement local and global derivative-free optimisation schemes. This library implements useful features such as run-time constraints. Notably IBM's Qiskit [47] utilises this package and as such Qolab matches the state of the art in functionality. Johnson [51] provides an explanation of each available method. Initialising multiple processors to complete independent optimisation on the same problem is a trivial scheme providing no increase in problem size but increases for evaluation.

CHAPTER 4

Results

We simultaneously investigate our mapping of graph similarity to the QAOA providing performance analysis of our Qolab package. Source code is available in Appendix C, the full set of original data-files, aggregate data and code to generate all plots is available in Appendix F. The exponential complexity of quantum state-vector simulation places a unique strain on performance analysis since we can by definition do no better than $\mathcal{O}(2^q)$. We provide correctness investigation of the QAOA up to 19 qubits and performance analysis for 22 qubits. We investigate eight different classical optimisation schemes.

4.1 Definitions

We consider a number of performance metrics defined below. We cannot present a standard approximation factor for our problem since the optimal value of our cost function is zero. We provide alternate correctness measures which illuminate the same trends.

- Number of Evaluations - We consider an evaluation the generation of a single $\left|\vec{\gamma}, \vec{\beta}\right\rangle$ state. This measure provides a rough estimate of how many logical (aptly sampled) quantum states are required. Lower is better.
- Sample Error - We use the expectation value of our system until the last iteration which samples the resulting state space V^2 times. The best value observed is compared to the known optimal as a standard approximation ratio. Higher is better.
- Expectation Error - The difference between the optimal solution and our expectation value normalised over the minimal possible value to account for changing graph sizes. Lower is better. A value of zero indicates certain measurement of the optimal solution

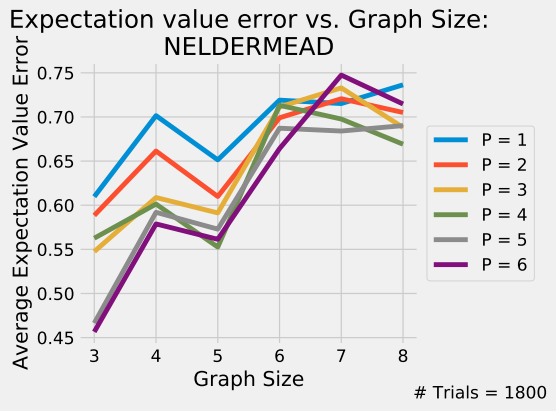
- Classical Comparison - We compare the final expectation value of our state with the corresponding expectation value of random solution selection. A negative value indicates better likelihood of a better solution from the QAOA. Higher is better
- Expectation Improvement - The net improvement in expectation value from the state generated by initial parameters. Higher is better.

4.1.1 Methodology

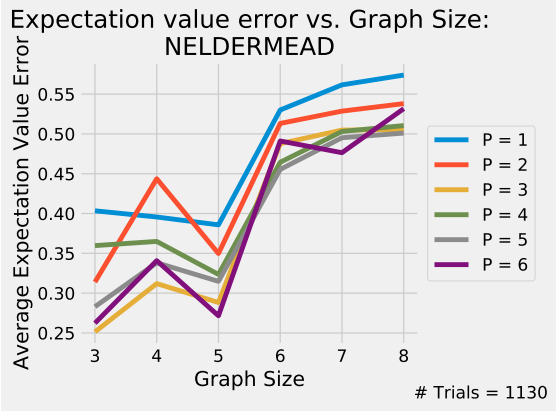
We present results from three datasets:

- An initial set using an old cost function containing $16,190$ trials. This alternate cost function sets the optimal value to zero, sub-optimal solutions to negative values and pads infeasible solutions with a maximal penalty of $-V^2$.
- A set of directed graph results containing $10,800$ trials.
- A set of undirected graph results containing 2500 trials.

In all trials we consider all deformation methods described in Section 3.2.5 equally and are aggregated together to provide an overall impression of QAOA performance.
The classical parameter optimisation component of the QAOA is critical to correctness and efficiency. Guerreschi and Smelyanskiy [37] provide the most in-depth analysis to date, we investigate eight different optimisation schemes. We provide extensive plotting of our results in Appendix F. We explicitly discuss the Nelder-Mead [65], Subplex [73], BOBYQA [70], Multi-Level Single-Linkage (MLSL) [52] and Dividing Rectangles (DIRECT) [72] methods. The Subplex algorithm is a variation on the legendary Nelder-Mead simplex algorithm [65] designed to target noisy function spaces. Simplex-based algorithms maintain a simplex of $n+1$ points to optimise n parameters. The BOBYQA algorithm estimates trust regions by forming quadratic models of the cost function. MLSL maintains a variety of local optimisations starting from a series of random points using heuristics to avoid repeated searching of local optima. The DIRECT algorithm is a deterministic global search algorithm based on dividing the search space into increasingly smaller hyper-rectangles.



(a) Alternate Cost Function



(b) Final Cost Function

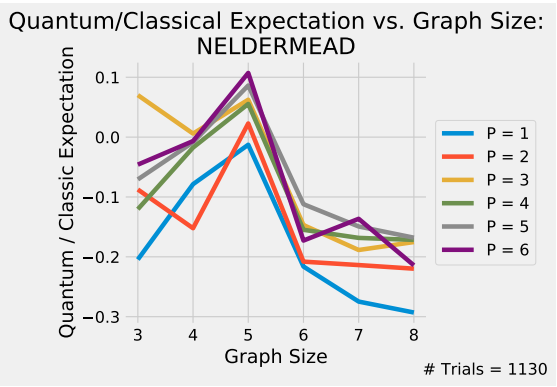
Figure 4.1: Expectation error for differing cost functions

4.2 Alternate Cost Function

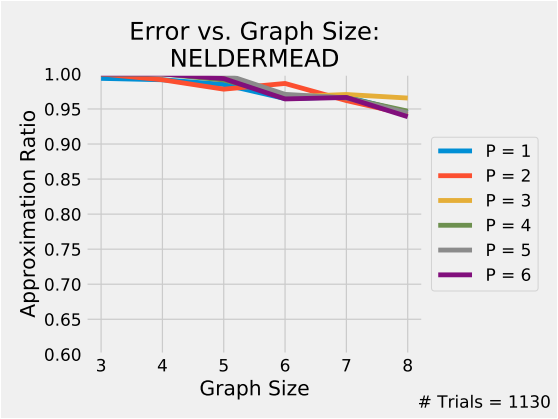
We see the effect of our 'tail' values most prominently when considering a slightly different cost function. Mapping optimal solutions to zero and penalising missing edges to negative values and mapping non-solution bit-strings to a minimal value. However, since these values now contribute to our expectation value, the overall performance is diminished depicted in Figure 4.2. Our final cost function performs significantly better where error grows significantly slower for all amounts of decomposition. This trend holds across all optimisation methods tested.

4.3 Increased Decomposition

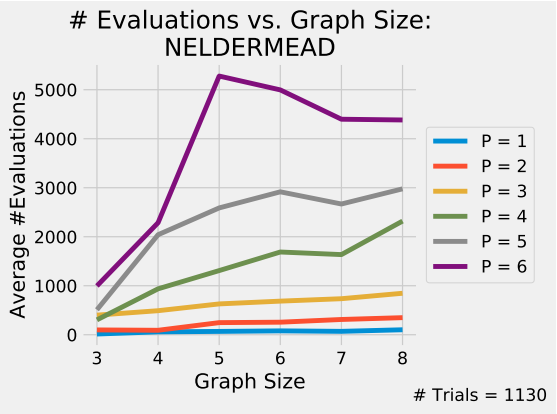
Figure 4.3 depicts performance metrics for the Nelder-Mead algorithm. We see excellent performance for this method matching current literature [37]. In Figure 4.3 we see the effect increased decomposition makes despite inferior end-results. We see that increasing QAOA decomposition results in an almost monotonic improvement in both sampled and expected solution quality at the cost of nearly double the number of function evaluations required for termination. This generalises to most optimisation methods tested. This is intuitive since we can always zero out parameters to provide the performance of a coarser QAOA scheme. The QAOA suffers from dimensionality issues since each increase in p exponentially increases the number of possible values requiring more optimisation iterations to make use of these additional parameters. This is most clearly seen in Figure 4.4(c).



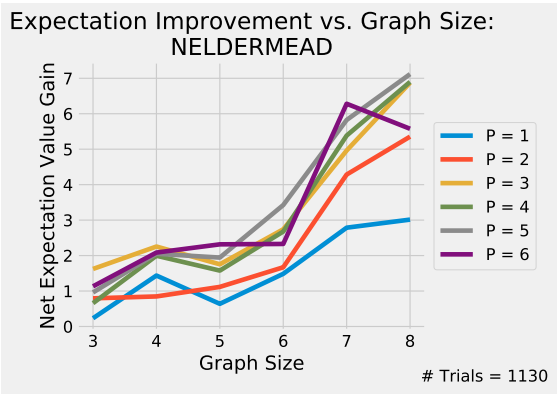
(a) Expectation value comparison



(b) Solution error

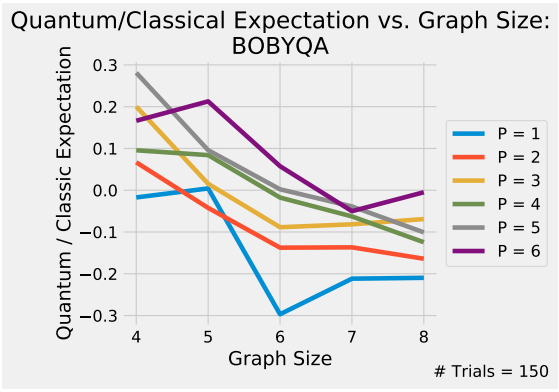


(c) Function evaluations required

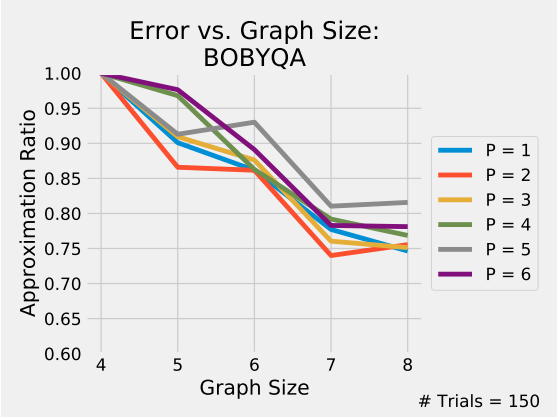


(d) Improvement

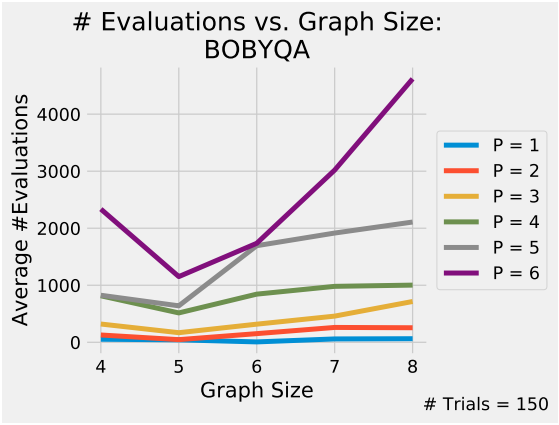
Figure 4.2: Final performance for the Nelder-Mead algorithm (directed graphs)



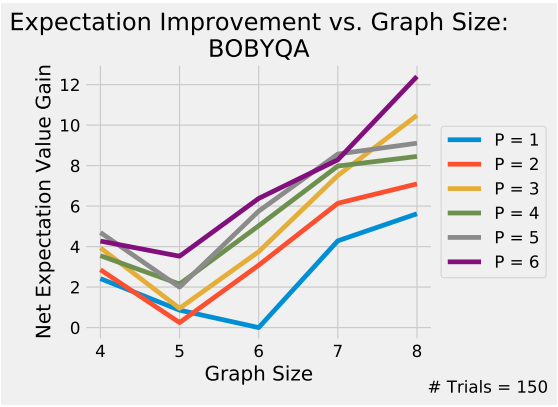
(a) Expectation value comparison



(b) Solution error



(c) Function evaluations required



(d) Improvement

Figure 4.3: Final performance for the BOBYQA algorithm (directed graphs)

4.4 Optimisation Methods

4.4.1 Correctness

The global nature of both the MLSL and DIRECT algorithms results in generally superior sampled and expected solutions over local methods. The MLSL algorithm achieves this by exhausting the maximal number of function evaluations defined as

$$MAX = S \times P \times GraphSize, \tag{4.4.1}$$

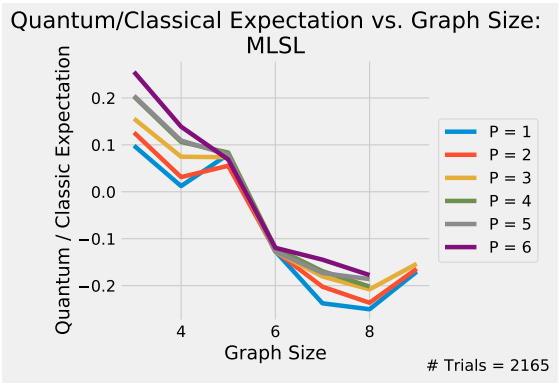
where S is a scaling parameter (we nominally choose 200, the standard value used by Scipy. [53]). The DIRECT algorithm terminates significantly earlier than other methods with a significant degradation in performance. For all algorithms tested the effect of infeasible solutions results in a lower expectation value versus classical sampling in the larger test cases indicating either more optimisation time is required or a more nuanced problem encoding. The cost function for graph similarity can span a range of V^2 values and produces a cost-function landscape containing many local optima. The derivative and non-linear nature of the optimisation schemes tested results in the scheme terminating at said optima.

4.4.2 Efficiency

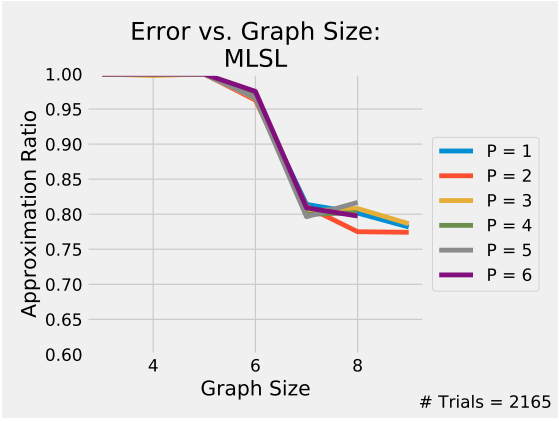
There is no clearly superior optimisation method for our mapping of graph similarity to the QAOA. Generally, global methods provide more correct solutions at a cost of vastly more function evaluations whereas local algorithms typically terminate with fewer iterations but produce poorer results as one would expect.

4.5 Directed vs. Undirected Graphs

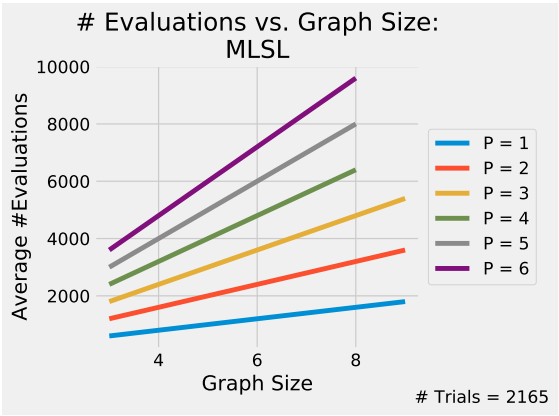
We compare the performance of the Subplex algorithm between directed and undirected graphs in Figure 4.5. We see superior solution quality in the undirected case. This is expected since the mappings of node labellings in the undirected case generates a cost function landscape spanning fewer unique values resulting in a 'smoother' cost-function landscape.



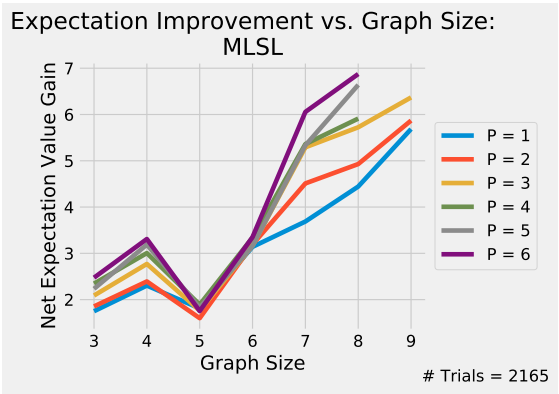
(a) Expectation value comparison



(b) Solution error

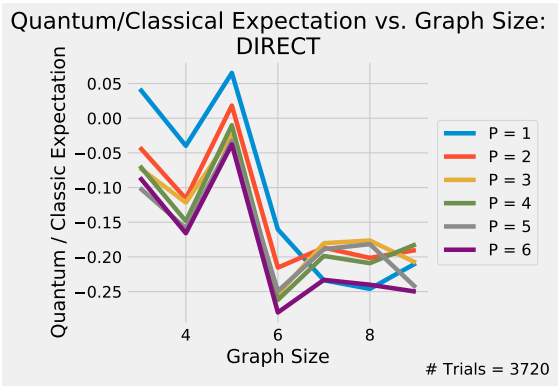


(c) Function evaluations required

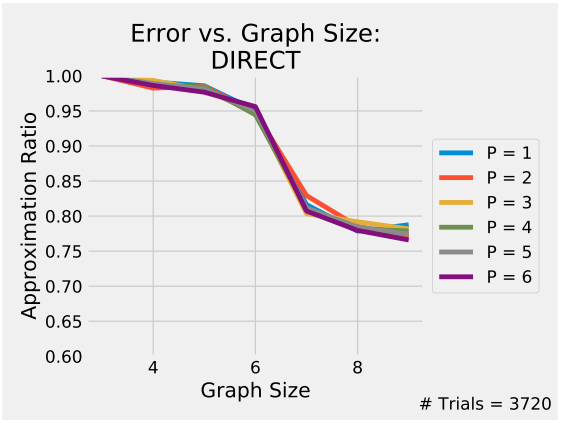


(d) Improvement

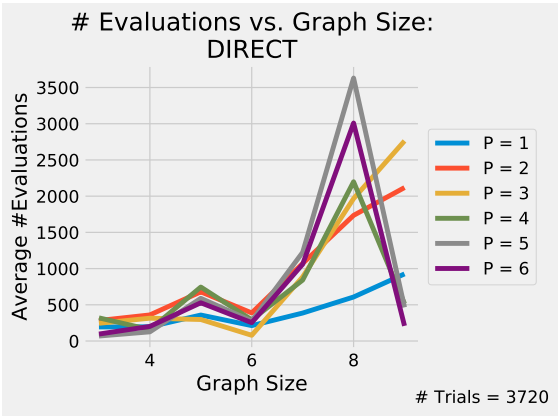
Figure 4.4: Final performance for the MLSL algorithm (directed graphs)



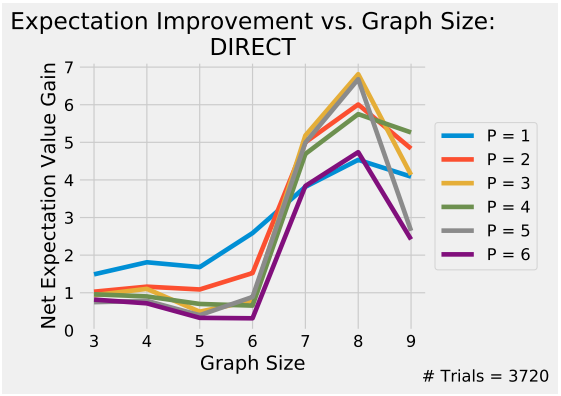
(a) Expectation value comparison



(b) Solution error

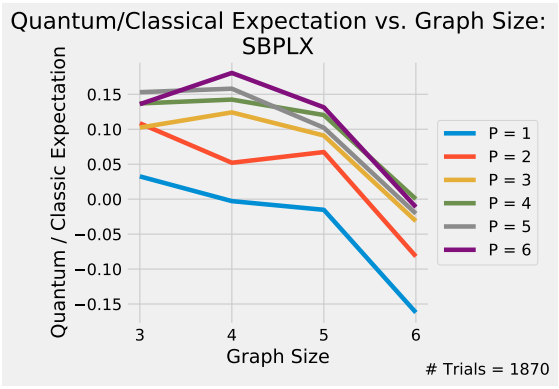


(c) Function evaluations required

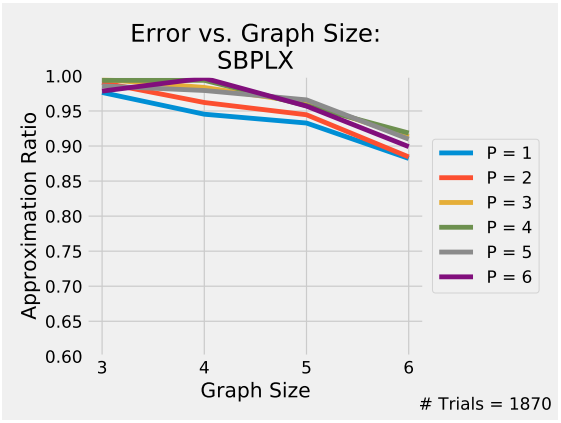


(d) Improvement

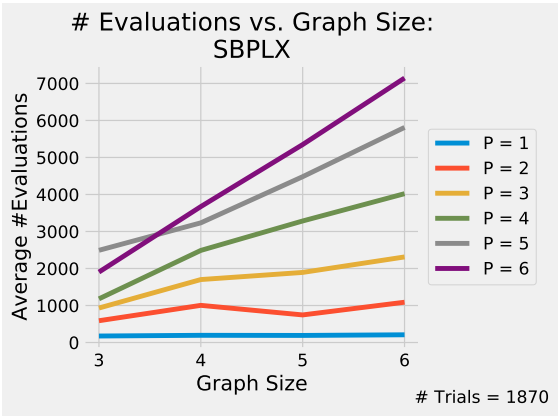
Figure 4.5: Final performance for the DIRECT algorithm (directed graphs)



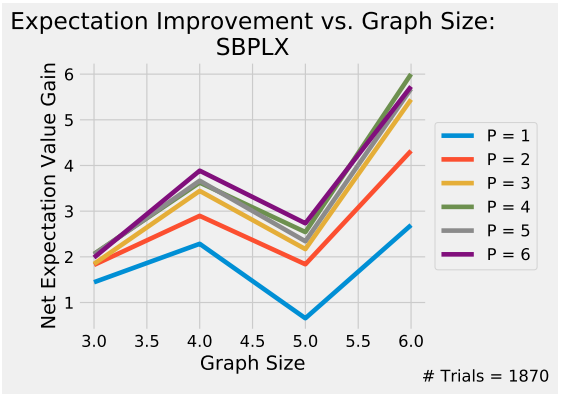
(a) Expectation value comparison



(b) Solution error

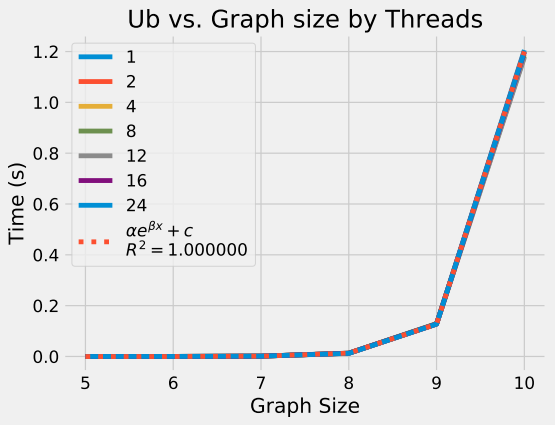


(c) Function evaluations required

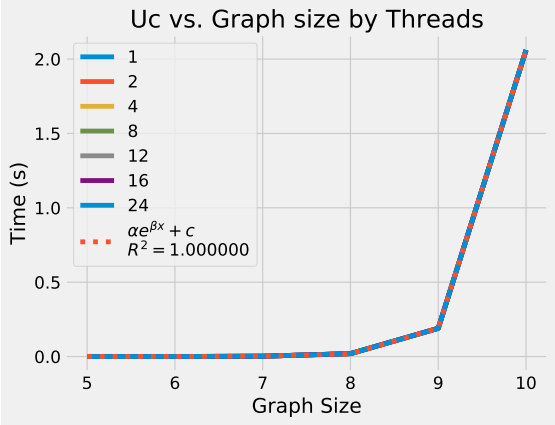


(d) Improvement

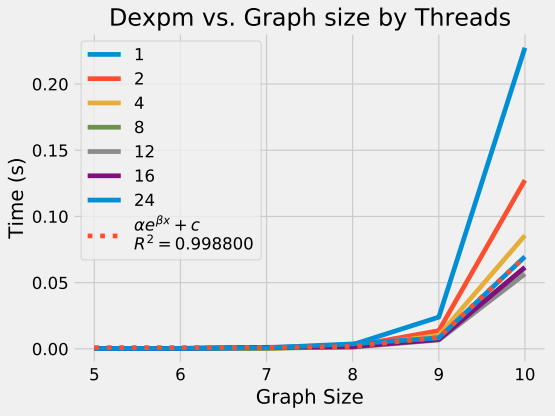
Figure 4.6: Final performance for the Subplex algorithm (undirected graphs)



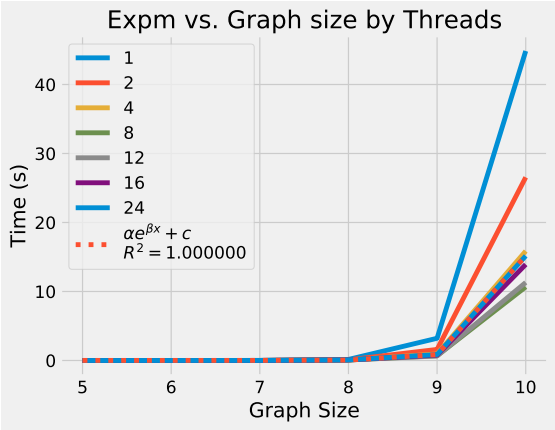
(a) $\hat{B}$ generation



(b) $\hat{C}$ generation



(c) Diagonal $\hat{C}$ Matrix Exponential



(d) $\hat{B}$ Matrix Exponential

Figure 4.7: Threading performance

4.6 Simulation Performance

We plot timing data for four major components of the simulation in Figure 4.6 and relative speed-up for each task in Figure 4.6. We compare the simulation time of each task with optimal threading in Figure 4.6, we see that the full matrix exponential dominates the computational workload. Although the work required by all tasks scales exponentially with graph-size the most intensive task is the computation of $\hat{U}_B$. This justifies our effort to make this operation efficient and lack of threading for the simpler tasks. Generating $\hat{C}$ and $\hat{B}$ occurs once in a trial whereas the $\hat{U}_C$ and $\hat{U}_B$ operations are performed thousands of times.

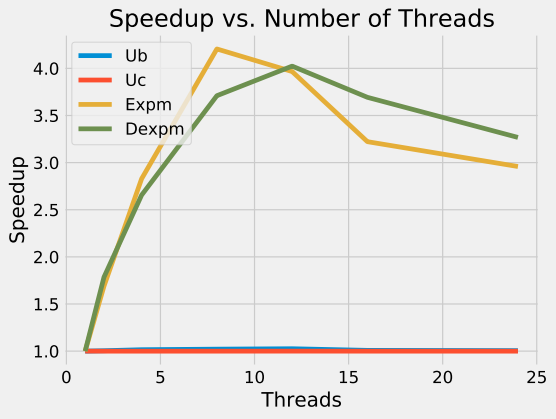


Figure 4.8: Relative speed-up for critical simulation tasks

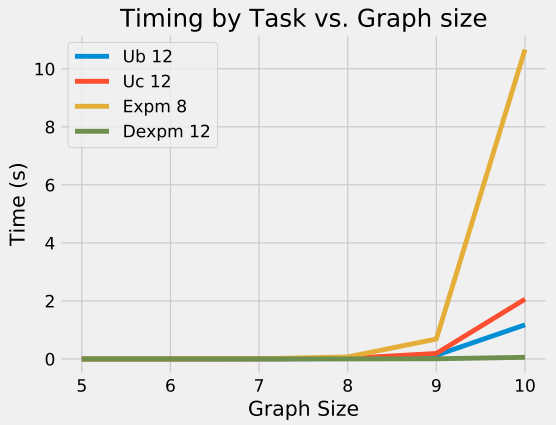


Figure 4.9: Timing for simulation tasks with optimal threading

CHAPTER 5

Conclusion

We explore the Quantum Approximate Optimisation Algorithm (QAOA) [31], an efficient combinatorial optimisation algorithm designed to approach NP-Complete problems. The low gate-depth, hybrid nature and generality makes the QAOA a strong candidate for near term implementation and practical use. We explore the algorithm in a novel manner independent of physical implementation through bespoke high-performance simulation testing up to 22 qubits. Our Quantum Approximate Optimisation package provides class-leading flexibility allowing simulation of both the QAOA and derivative algorithms [61] while providing efficient scaling from desktop to cluster scale. We investigate the use of eight classical optimisation schemes seeking insight into any differences in performance offered by different optimisation paradigms finding a general trade-off between correctness and efficiency when considering global versus local methods.

We encode the problem of graph similarity to the QAOA by considering the edge-overlap between two unlabelled vertices for both directed and undirected graphs; a similarity measure not yet considered classically. We provide a novel encoding scheme saving $\mathcal{O}(V)$ qubits at the cost of considering infeasible solutions. The run-time cost of this space saving extends through both the quantum and classical portions of the algorithm; the former due to the increased gate-depth and circuit complexity required to encode the problem and the latter due to the non-solutions. The QAOA is able to optimise out of an initial local minima including the infeasible solutions but requires significantly more optimisation to find good solutions. We investigate an initial starting state of superposition between all qubits testing the resilience of the QAOA against a significant number of infeasible and undesirable solutions.

Run-time restrictions placed on high-performance compute resources make evaluating larger problem instances problematic for continuous simulation of the QAOA; implementation of a check-point system or multi-start paradigm would allow for exploration of larger problem instances. Investigation of a wider variety of hard combinatorial optimisation problems may reveal additional avenues to exploit the sub-structure of certain problems and further discriminate between classical opti-

misation schemes. Moreover the formulation of a parameter optimisation schemes targeted directly for the QAOA may lead to significant performance benefits. Finally, tighter integration into pre-existing quantum computing resources (both simulators and physical implementations) will facilitate a more complete investigation of this promising and exotic algorithm.

Bibliography

[1] Daniel S. Abrams and Seth Lloyd. "Simulation of Many-Body Fermi Systems on a Universal Quantum Computer". In: *Physical Review Letters* 79.13 (Sept. 29, 1997), pp. 2586–2589. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.79.2586. URL: https://link.aps.org/doi/10.1103/PhysRevLett.79.2586 (visited on 10/15/2018).

[2] D. Aggarwal et al. "Quantum attacks on Bitcoin, and how to protect against them". In: *ArXiv e-prints* (Oct. 2017).

[3] Awad H. Al-Mohy and Nicholas J. Higham. "A New Scaling and Squaring Algorithm for the Matrix Exponential". In: *SIAM Journal on Matrix Analysis and Applications* 31.3 (Jan. 2010), pp. 970–989. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/09074721X. URL: http://epubs.siam.org/doi/10.1137/09074721X (visited on 09/16/2018).

[4] Awad H. Al-Mohy and Nicholas J. Higham. "Computing the Action of the Matrix Exponential, with an Application to Exponential Integrators". In: *SIAM Journal on Scientific Computing* 33.2 (Jan. 2011), pp. 488–511. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/100788860. URL: http://epubs.siam.org/doi/10.1137/100788860 (visited on 08/02/2018).

[5] T. Auckenthaler et al. "Matrix exponentials and parallel prefix computation in a quantum control problem". In: *Parallel Computing* 36.5 (June 2010), pp. 359–369. ISSN: 01678191. DOI: 10.1016/j.parco.2010.01.006. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167819110000244 (visited on 09/17/2018).

[6] Satish Balay et al. *PETSc Web page*. 2018. URL: http://www.mcs.anl.gov/petsc.

[7] Boaz Barak et al. "Beating the random assignment on constraint satisfaction problems of bounded degree". In: *arXiv:1505.03424 [cs]* (May 13, 2015). arXiv: 1505.03424. URL: http://arxiv.org/abs/1505.03424 (visited on 10/05/2018).

[8] Charles H. Bennett et al. "Strengths and Weaknesses of Quantum Computing". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1510,1523. ISSN: 0097-5397.

[9] Luca Bergamaschi and Marco Vianello. "Efficient computation of the exponential operator for large, sparse, symmetric matrices". In: *Numerical Linear Algebra with Applications* 7.1 (Jan. 2000), pp. 27–45. ISSN: 1070-5325, 1099-1506. DOI: 10.1002/(SICI)1099-1506(200001/02)7:1<27::AID-NLA185>3.0.CO;2-4. URL: http://doi.wiley.com/10.1002/%28SICI%291099-1506%28200001/02%297%3A1%3C27%3A%3AAID-NLA185%3E3.0.CO%3B2-4 (visited on 09/17/2018).

[10] Jacob D. Biamonte, Mauro E. S. Morales, and Dax Enshan Koh. "Quantum Supremacy Lower Bounds by Entanglement Scaling". In: *arXiv:1808.00460 [cond-mat, physics:quant-ph]* (Aug. 1, 2018). arXiv: 1808.00460. URL: http://arxiv.org/abs/1808.00460 (visited on 08/03/2018).

[11] S. Boixo et al. "Characterizing Quantum Supremacy in Near-Term Devices". In: *ArXiv e-prints* (July 2016).

[12] S. Boixo et al. "Simulation of low-depth quantum circuits as complex undirected graphical models". In: *ArXiv e-prints* (Dec. 2017).

[13] Endre Boros and Peter L Hammer. "Pseudo-Boolean optimization". In: *Discrete Applied Mathematics* (2002), p. 71.

[14] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual Web search engine". In: *Computer Networks and ISDN Systems* 30.1 (1998), pp. 107,117. ISSN: 0169-7552.

[15] André Chailloux, María Naya-Plasencia, and André Schrottenloher. *An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography*. Published: Cryptology ePrint Archive, Report 2017/847. 2017. URL: https://eprint.iacr.org/2017/847.

[16] J. Chen et al. "Classical Simulation of Intermediate-Size Quantum Circuits". In: *ArXiv e-prints* (May 2018).

[17] Z.-Y. Chen et al. "64-Qubit Quantum Circuit Simulation". In: *ArXiv e-prints* (Feb. 2018).

[18] M. Chiew et al. "Graph comparison via nonlinear quantum search". In: *arXiv:1810.01647 [quant-ph]* (Oct. 3, 2018). arXiv: 1810.01647. URL: http://arxiv.org/abs/1810.01647 (visited on 10/15/2018).

[19] Andrew MacGregor Childs. "Quantum Information Processing in Continuous Time". In: (2004), p. 140.

[20] Carlo Ciliberto et al. "Quantum machine learning: a classical perspective". In: *Proceedings. Mathematical, Physical, and Engineering Sciences* 474.2209 (2018), Proceedings. Mathematical, Physical, and Engineering Sciences, 2018, Vol.474(2209). ISSN: 1364-5021.

[21] R. Cleve et al. “Quantum algorithms revisited”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 339,354. ISSN: 1364-5021.

[22] E. Crosson et al. “Different Strategies for Optimization Using the Quantum Adiabatic Algorithm”. In: *ArXiv e-prints* (Jan. 2014).

[23] D. Deutsch. “Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer”. In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences (1934-1990)* 400.1818 (), pp. 97,117. ISSN: 0080-4630.

[24] David Deutsch and Richard Jozsa. “Rapid Solution of Problems by Quantum Computation”. In: *Proceedings of the Royal Society: Mathematical and Physical Sciences (1990-1995)* 439.1907 (1992), pp. 553,558. ISSN: 0962-8444.

[25] Reinhard. Diestel. *Graph theory*. 2nd ed. Graduate texts in mathematics ; 173. New York: Springer, 2000. ISBN: 0-387-95014-1.

[26] P. A. M. Dirac. “A new notation for quantum mechanics”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 35.3 (July 1939), p. 416. ISSN: 0305-0041, 1469-8064. DOI: 10.1017/S0305004100021162. URL: http://www.journals.cambridge.org/abstract_S0305004100021162 (visited on 09/12/2018).

[27] I Duff, Roger Grimes, and John Lewis. “Sparse matrix test problems”. In: *ACM Transactions on Mathematical Software (TOMS)* 15.1 (Mar. 1, 1989), pp. 1,14. ISSN: 1557-7295.

[28] Jian-Yun Fang. “One step time propagation method for systems with time-dependent Hamiltonians”. In: *Chemical Physics Letters* 263.6 (Dec. 1996), pp. 759–766. ISSN: 00092614. DOI: 10.1016/S0009-2614(96)01272-9. URL: http://linkinghub.elsevier.com/retrieve/pii/S0009261496012729 (visited on 09/17/2018).

[29] E. Farhi, J. Goldstone, and S. Gutmann. “A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem”. In: *ArXiv e-prints* (Dec. 2014).

[30] E. Farhi and A. W Harrow. “Quantum Supremacy through the Quantum Approximate Optimization Algorithm”. In: *ArXiv e-prints* (Feb. 2016).

[31] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. “A Quantum Approximate Optimization Algorithm”. In: *arXiv:1411.4028 [quant-ph]* (Nov. 14, 2014). arXiv: 1411.4028. URL: http://arxiv.org/abs/1411.4028 (visited on 10/05/2018).

[32] Edward Farhi et al. “A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem”. In: *Science* 292.5516 (2001), pp. 472,5. ISSN: 00368075. URL: http://search.proquest.com/docview/213575154/?pq-origsite=primo.

[33] Richard Feynman. “Simulating physics with computers”. In: *International Journal of Theoretical Physics* 21.6 (1982), pp. 467,488. ISSN: 0020-7748.

[34] Michael R. Garey. *Computers and intractability : a guide to the theory of NP-completeness*. A Series of books in the mathematical sciences. San Francisco: W. H. Freeman, 1979. ISBN: 0-7167-1044-7.

[35] L. K. Grover. “A fast quantum mechanical algorithm for database search”. In: *eprint arXiv:quant-ph/9605043* (May 1996).

[36] L. K. Grover. “How fast can a quantum computer search?” In: *eprint arXiv:quant-ph/9809029* (Sept. 1998).

[37] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. “Practical optimization for hybrid quantum-classical algorithms”. In: *arXiv:1701.01450 [quant-ph]* (Jan. 5, 2017). arXiv: 1701.01450. URL: http://arxiv.org/abs/1701.01450 (visited on 10/05/2018).

[38] S. Hadfield. “Quantum Algorithms for Scientific Computing and Approximate Optimization”. In: *ArXiv e-prints* (May 2018).

[39] S. Hadfield et al. “From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz”. In: *ArXiv e-prints* (Sept. 2017).

[40] Yongkoo Han et al. “Topological Similarity-Based Feature Selection for Graph Classification.” In: *The Computer Journal* 58.9 (2015), pp. 1884–1893. DOI: 10.1093/comjnl/bxt123. URL: http://dx.doi.org/10.1093/comjnl/bxt123.

[41] M Hattori et al. “Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways”. In: *Journal Of The American Chemical Society* 125.39 (2003), pp. 11853,11865. ISSN: 0002-7863.

[42] Itay Hen and A. P. Young. “Solving the graph-isomorphism problem with a quantum annealer”. In: *Physical Review A* 86.4 (Oct. 10, 2012). ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.86.042310. URL: https://link.aps.org/doi/10.1103/PhysRevA.86.042310 (visited on 10/15/2018).

[43] Vicente Hernandez, Jose E. Roman, and Vicente Vidal. “SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems”. In: *ACM Trans. Math. Software* 31.3 (2005), pp. 351–362.

[44] Maureen Heymans and Ambuj K. Singh. "Deriving phylogenetic trees from the similarity analysis of metabolic pathways". In: *Bioinformatics* 19.1 (2003), pp. 138,146. ISSN: 1367-4803.

[45] Nicholas J. Higham and Françoise Tisseur. "A Block Algorithm for Matrix 1-Norm Estimation, with an Application to 1-Norm Pseudospectra". In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (Jan. 2000), pp. 1185–1201. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/S0895479899356080. URL: http://epubs.siam.org/doi/10.1137/S0895479899356080 (visited on 08/11/2018).

[46] Thomas Häner and Damian S. Steiger. "0.5 Petabyte Simulation of a 45-Qubit Quantum Circuit". In: *arxiv1704.01127* (2017), Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC 2017. Article No. 33.

[47] IBM. *Qiskit Aqua*. URL: https://github.com/Qiskit/aqua.

[48] *Intel(R) Math Kernel Library 2018 Update 4*. URL: https://software.intel.com/en-us/mkl.

[49] Josh A. Izaac and Jingbo B. Wang. "pyCTQW: A continuous-time quantum walk simulator on distributed memory computers". In: *Computer Physics Communications* 186 (Jan. 2015), pp. 81–92. ISSN: 00104655. DOI: 10.1016/j.cpc.2014.09.011. URL: http://linkinghub.elsevier.com/retrieve/pii/S0010465514003166 (visited on 09/16/2018).

[50] Kathryn James. "Six Degrees of Information Seeking: Stanley Milgram and the Small World of the Library". In: *Journal of Academic Librarianship* 32.5 (2006), pp. 527,532. ISSN: 0099-1333.

[51] Steven G. Johnson. *The NLopt nonlinear-optimization package*. 2011. URL: http://ab-initio.mit.edu/nlopt.

[52] D. Jones, C. Perttunen, and B. Stuckman. "Lipschitzian optimization without the Lipschitz constant". In: *Journal of Optimization Theory and Applications* 79.1 (1993), pp. 157,181. ISSN: 0022-3239.

[53] Eric Jones et al. *SciPy: Open source scientific tools for Python*. 2001. URL: http://www.scipy.org/.

[54] Julian Kelly. *A Preview of Bristlecone, Google's New Quantum Processor*. 2018. URL: https://research.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html.

[55] Jon Kleinberg. "Authoritative sources in a hyperlinked environment". In: *Journal of the ACM* 46.5 (1999), pp. 604,632. ISSN: 0004-5411. URL: http://search.proquest.com/docview/1808072946/.

[56] Giorgos Kollias et al. “Fast parallel algorithms for graph similarity and matching”. In: *Journal of Parallel and Distributed Computing* 74.5 (2014), pp. 2400 –2410. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2013.12.010. URL: http://www.sciencedirect.com/science/article/pii/S0743731513002529.

[57] M. Lades et al. “Distortion invariant object recognition in the dynamic link architecture”. In: *IEEE Transactions on Computers* 42.3 (Mar. 1993), pp. 300–311. ISSN: 0018-9340. DOI: 10.1109/12.210173.

[58] R. Li et al. “Quantum Supremacy Circuit Simulation on Sunway Taihu-Light”. In: *ArXiv e-prints* (Apr. 2018).

[59] Andrew Lucas. “Ising formulations of many NP problems”. In: *Frontiers in Physics* 2 (2014). ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005. arXiv: 1302.5843. URL: http://arxiv.org/abs/1302.5843 (visited on 09/20/2018).

[60] Igor L. Markov et al. “Quantum Supremacy Is Both Closer and Farther than It Appears”. In: *arXiv:1807.10749 [quant-ph]* (July 27, 2018). arXiv: 1807.10749. URL: http://arxiv.org/abs/1807.10749 (visited on 08/01/2018).

[61] Samuel Marsh and Jingbo Wang. “A quantum walk assisted approximate algorithm for bounded NP optimisation problems”. In: *arXiv:1804.08227 [quant-ph]* (Apr. 22, 2018). arXiv: 1804.08227. URL: http://arxiv.org/abs/1804.08227 (visited on 10/15/2018).

[62] Cleve Moler and Charles Van Loan. “Nineteen Dubious Ways to Compute the Exponential of a Matrix”. In: *SIAM Review* 20.4 (Oct. 1978), pp. 801–836. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/1020098. URL: http://epubs.siam.org/doi/10.1137/1020098 (visited on 09/17/2018).

[63] Cleve Moler and Charles Van Loan. “Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later”. In: *SIAM Review* 45.1 (Jan. 2003), pp. 3–49. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/S00361445024180. URL: http://epubs.siam.org/doi/10.1137/S00361445024180 (visited on 08/02/2018).

[64] Mamadou Ndong et al. “A Chebychev propagator with iterative time ordering for explicitly time-dependent Hamiltonians”. In: *The Journal of Chemical Physics* 132.6 (Feb. 14, 2010), p. 064105. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.3312531. URL: http://aip.scitation.org/doi/10.1063/1.3312531 (visited on 09/17/2018).

[65] J. A. Nelder and R. Mead. “A Simplex Method for Function Minimization”. In: *The Computer Journal* 7.4 (1965), pp. 308,313. ISSN: 0010-4620.

[66] Michael A. Nielsen. *Quantum computation and quantum information*. Cambridge: Cambridge University Press, 2000. ISBN: 0-521-63235-8.

[67] J. S. Otterbach et al. "Unsupervised Machine Learning on a Hybrid Quantum Computer". In: *ArXiv e-prints* (Dec. 2017).

[68] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. "Web graph similarity for anomaly detection". In: *Journal of Internet Services and Applications* 1.1 (May 2010), pp. 19–30. ISSN: 1869-0238. DOI: 10.1007/s13174-010-0003-x. URL: https://doi.org/10.1007/s13174-010-0003-x.

[69] E. Pednault et al. "Breaking the 49-Qubit Barrier in the Simulation of Quantum Circuits". In: *ArXiv e-prints* (Oct. 2017).

[70] M. J. D. Powell. "Direct search algorithms for optimization calculations". In: *Acta Numerica* 7 (1998), pp. 287–336. DOI: 10.1017/S0962492900002841.

[71] John W. Raymond, Eleanor J. Gardiner, and Peter Willett. "RASCAL: Calculation of Graph Similarity using Maximum Common Edge Subgraphs". In: *The Computer Journal* 45.6 (2002), pp. 631,644. ISSN: 0010-4620.

[72] A. Rinnooy Kan and G. Timmer. "Stochastic global optimization methods part II: Multi level methods". In: *Mathematical Programming* 39.1 (1987), pp. 57,78. ISSN: 0025-5610.

[73] Thomas H. Rowan. "Functional stability analysis of numerical algorithms". PhD thesis. 1990. 218 pp. URL: https://search-proquest-com.ezproxy.library.uwa.edu.au/docview/303865032?accountid=14681.

[74] G. E. Santoro. "Theory of Quantum Annealing of an Ising Spin Glass". In: *Science* 295.5564 (Mar. 29, 2002), pp. 2427–2430. ISSN: 00368075, 10959203. DOI: 10.1126/science.1068774. URL: http://www.sciencemag.org/cgi/doi/10.1126/science.1068774 (visited on 10/15/2018).

[75] Robert Sedgewick. "Permutation Generation Methods". In: *ACM Comput. Surv.* 9.2 (June 1977), pp. 137–164. ISSN: 0360-0300. DOI: 10.1145/356689.356692. URL: http://doi.acm.org/10.1145/356689.356692.

[76] E. A. Sete, W. J. Zeng, and C. T. Rigetti. "A functional architecture for scalable quantum computing". In: *2016 IEEE International Conference on Rebooting Computing (ICRC)*. Oct. 2016, pp. 1–6. DOI: 10.1109/ICRC.2016.7738703.

[77] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM journal on computing* 5 (1997), SIAM journal on computing , (5), p.1509. ISSN: 0097-5397.

[78] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* (1994), pp. 124,134.

[79] Roger B. Sidje. "Expokit: a software package for computing matrix exponentials". In: *ACM Transactions on Mathematical Software* 24.1 (Mar. 1, 1998), pp. 130–156. ISSN: 00983500. DOI: 10.1145/285861.285868. URL: http://portal.acm.org/citation.cfm?doid=285861.285868 (visited on 09/17/2018).

[80] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–9. ISSN: 0028-0836.

[81] M. Smelyanskiy, N. P. D. Sawaya, and A. Aspuru-Guzik. "qHiPSTER: The Quantum High Performance Software Testing Environment". In: *ArXiv e-prints* (Jan. 2016).

[82] B. Strug. "Automatic design quality evaluation using graph similarity measures". In: *Automation in Construction* 32 (2013), pp. 187 –195. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2012.12.015. URL: http://www.sciencedirect.com/science/article/pii/S0926580512002543.

[83] J. B. Wang and S. Midgley. "Quantum waveguide theory: A direct solution to the time-dependent Schrödinger equation". In: *Physical Review B* 60.19 (Nov. 15, 1999), pp. 13668–13675. ISSN: 0163-1829, 1095-3795. DOI: 10.1103/PhysRevB.60.13668. URL: https://link.aps.org/doi/10.1103/PhysRevB.60.13668 (visited on 09/17/2018).

[84] J. B. Wang and T. T. Scholz. "Time-dependent approach to scattering by Chebyshev-polynomial expansion and the fast-Fourier-transform algorithm". In: *Physical Review A* 57.5 (May 1, 1998), pp. 3554–3559. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.57.3554. URL: https://link.aps.org/doi/10.1103/PhysRevA.57.3554 (visited on 09/17/2018).

[85] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. "Training a quantum optimizer". In: *Phys. Rev. A* 94.2 (Aug. 2016), p. 022309. DOI: 10.1103/PhysRevA.94.022309. URL: https://link.aps.org/doi/10.1103/PhysRevA.94.022309.

[86] Eric W. Weisstein. *Graph Isomorphism Complete*. MathWorld - A Wolfram Web Resource. 2018. URL: http://mathworld.wolfram.com/GraphIsomorphismComplete.html (visited on 09/13/2018).

[87] Jonathan Welch et al. “Efficient quantum circuits for diagonal unitaries without ancillas”. In: *New Journal of Physics* 16.3 (Mar. 31, 2014), p. 033040. ISSN: 1367-2630. DOI: 10.1088/1367-2630/16/3/033040. URL: http://stacks.iop.org/1367-2630/16/i=3/a=033040?key=crossref.8d88fafdb935d9cac92d5040d8b6a5c5 (visited on 10/16/2018).

[88] L. Wiskott et al. “Face recognition by elastic bunch graph matching”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.7 (July 1997), pp. 775–779. ISSN: 0162-8828. DOI: 10.1109/34.598235.

[89] Laura A. Zager and George C. Verghese. “Graph similarity scoring and matching”. In: *Applied Mathematics Letters* 21.1 (2008), pp. 86 –94. ISSN: 0893-9659. DOI: https://doi.org/10.1016/j.aml.2007.01.006. URL: http://www.sciencedirect.com/science/article/pii/S0893965907001012.

[90] Weiguo Zheng et al. “Efficient Graph Similarity Search Over Large Graph Databases”. In: *Knowledge and Data Engineering, IEEE Transactions on* 27.4 (2015), pp. 964,978. ISSN: 1041-4347.

APPENDIX A

Original Honours Proposal

Title: Quantum Graph Similarity and Applications

Author: Nicholas Pritchard

Supervisor: Professor Jingbo Wang and Professor Amitava Datta

Degree: BSc(Hons.)

Background

The goal of quantum computing is the exploit the complexity of quantum systems for useful computation. Such a motivation arises from the fact that despite decades of research modelling quantum systems in classical computers has alluded the scientific community [11]. The development of near-term physical hardware [54], [76] has combined with a surge of interest to apply quantum computing to numerous fields in computation [15, 29, 10, 20, 21]. Graph similarity and graph-isomorphism are long-standing difficulties in computer science. Many useful formulations of graph similarity exist such as compound matching in chemistry [40, 41, 71], machine vision [57] and web-search [14]. This problem has no tractable exact formulation for graphs with unknown node correspondence and as such approximate solutions are considered industry standard. More specifically we define whole-graph similarity

Definition A.0.1. Whole Graph Similarity: Given two graphs $G_1(v_1, e_1)$ and $G_2(v_2, e_2)$ with possibly different numbers of vertices and edges, find an algorithm which returns a measure of similarity $S|S \in [0, 1]$. Furthermore:

1. $S(G_1, G_1) = 1$
2. $S(G_1, G_2) = S(G_2, G_1)$

Aim

To investigate a quantum algorithmic approach to graph similarity and its applications this project will examine graph similarity by applying the recently proposed 'Quantum Approximate Optimisation Algorithm' (QAOA) [29]. The QAOA can be formulated to reduce a combinatorial optimisation problem to a parameter search on around two variables. This project primarily aims to investigate if such an approach leads to any improvements in speed, accuracy or robustness over classical methods.
A secondary objective is to extend the generated model of graph similarity into a real-world contextual use such as object tracking or common sub-graph matching for example. Currently it is unknown whether a quantum advantage will yield any benefits in speed, accuracy or robustness over classical counter-parts and hence makes a suitable topic for research.

Method

There will be a large amount of theoretical work in the development and validation of a quantum or hybrid quantum/classical algorithms to tackle the problem of graph similarity. Testing will require a series of standardised sources as well as comparison results or implementations of classical algorithms. Due to the compute-heavy nature of simulating quantum systems it is likely the Magnus supercomputer at the Pawsey Super-computing Centre will need to be utilised. *Python*, *C/C++* and possibly *Fortran* will be used to simulate various approaches and to create visualisations of resulting circuit designs.

Status

This project does not follow from previous work commencing at the start of the 2018 academic year. This research is a collaboration between the *Quantum dynamics and computation* research group and the department of *Computer science and software engineering*.
Currently, a general understanding of quantum computing and potential object detection methods are being investigated in addition to classical object-tracking frameworks with the aim of finding a suitable starting point to apply quantum methods. An investigation into state-of-the-art quantum simulations yields a number of possible methods and frameworks [81, 69, 12, 17, 16]. Preliminary-work on simulating the Quantum Approximate Optimisation Algorithm (QAOA) locally is underway.

Software and Hardware Requirements

Costs are expected to be negligible as access to required software packages are openly available on-line or through the University of Western Australia. The majority of testing is to be run on personal machines. Use of the Pawsey Supercomputing Centre is available if needed based on an agreement with the University.

APPENDIX B

Tail Complexity

Vertices (V)	$V!$	Qubits (q)	2^q	Difference($2^q - V!$)	$2^q/V!$
2	2	1	2	0	0
4	6	5	32	26	4.3
8	40320	16	65536	25216	0.65
10	3628800	22	4194304	565504	0.15
12	479001600	29	536870912	57869312	0.12
15	$1.30767E12$	41	$2.19902E12$	$8.91349E11$	0.68
22	$1.124E21$	70	$1.18059E21$	$5.65909E19$	0.05

Table B.1: Graph-size compared to qubit state-space

B.1 Series Expansion at $n = \infty$

$$2^{log_2(n!)}(\frac{\sqrt{\frac{1}{n}}}{\sqrt{2\pi}} + \mathcal{O}((\frac{1}{n})^{3/2}))exp((1 - log(n))n + \mathcal{O}((\frac{1}{n})^2)) - 1 \qquad \text{(B.1.1)}$$

APPENDIX C

Description of Qolab

Qolab is a near problem agnostic simulation of the QAOA [31] with extended constraint ability [61]. Implementation using Intel's Math Kernel Library [48] facilitates maximal desktop and single node performance. The open-source nlopt optimisation suite [51] allows for a variety of optimisation algorithms to be tested simply. Qolab supports cluster execution allowing exacting state-space decomposition. Code is available at

`https://bitbucket.org/qaoa_uwa/graphsimilarity/src/master/`

Currently, Qolab supports the following arguments:

- Arbitrary number of qubits
- Arbitrary cost function
- Mixing masks on the $\hat{U}_B$ operator with an alternate function evaluation path as per Algorithm 5 proposed by Marsh and Wang [61].
- Arbitary trotterisation depth (p variable)
- Confidence interval based sampling scaling
- Arbitrary sampling
- Command line support for nlopt optimisation selection
- QAOA $(\vec{\gamma}, \vec{\beta})$ argument optimisation tolerances
- Variable function output tolerance selection
- Full-desktop and cluster implementations

APPENDIX D

Pseudocode

D.1 Lehmer Code Permutation

Algorithm 4 Factoradic Permutation Generator

```
1: procedure K_PERM(n, k)
2:     facts[] ← ∅
3:     items[] ← 0, ..., n
4:     out[] ← ∅
5:     nnz ← 0
6:     size ← n
7:     while size > 0 do
8:         f ←factorial(size−1)
9:         i ← k/f
10:        x ←items[k]
11:        k ← kmodf
12:        out[nnz] ← x
13:        for j = 0 to n - 1 do
14:            items[j] ← items[j+1]
15:        end for
16:        size ← size - 1
17:        nnz ← nnz + 1
18:    end while
19:    Return out
20: end procedure
```

This procedure returns the k-th permutation of natural numbers $[0, \ldots, n)$

Algorithm 5 NPO QAOA Overview

procedure NPO_QAOA_Core(numQubits, P, optimisationMethod, C(x), Mask(x))
    $\hat{U}_C \leftarrow$**genUC**(C(x))
    $\hat{U}_B \leftarrow$**genUB**(numQubits, Mask(x))
    $\vec{\gamma}, \vec{\beta} \leftarrow$**initialParameters()**
    while terminateTest() **do**
        $|\psi\rangle \leftarrow$**initialState**
        for $i = 0$ to p **do**
            $|\psi\rangle \leftarrow \hat{U}_B(\beta_i)\,|\psi\rangle$
            $|\psi\rangle \leftarrow \hat{U}_C(\gamma_i)\,|\psi\rangle$
        end for
        $|\psi\rangle \leftarrow \hat{U}_B(\beta_{p+1})\,|\psi\rangle$
        $F_p \leftarrow$ **Measure**(ψ)
        $\vec{\gamma}, \vec{\beta} \leftarrow$ **updateParameters**(F_p)
    end while
end procedure
Report()

D.2 NPO QAOA

Note the subtle differences to the QAOA in Algorithm 5, an additional $\hat{U}_B$ operation is applied at the start and a validation mask is passed to the $\hat{B}$ generation routine.

APPENDIX E

Proofs

We rely on the Perron-Frobenius theroem which states

Theorem 1. A symmetric elements of only real, non-negative entries has all real non-negative eigenvalues. Furthermore there exists a Perron-Root r for which all other eigenvalues $\lambda \leq r$.

We show $\lambda_{min,max} = \pm q$. Let B be our $2^q \times 2^q$ hypercube adjacency matrix constructed according to

$$\hat{B} = \sum_{i=1}^{n} \sigma_i^x \tag{E.0.1}$$

We note that B is Hermitian and therefore symmetric and by definition contains real values. We shall show that there exists and eigenvalue λ of B such that for any vector $\mathbf{v} \in \mathbb{R}^n$ we have

$$\mathbf{v} \cdot B\mathbf{v} \leq \lambda \|\mathbf{v}\|^2 \tag{E.0.2}$$

For a real symmetric matrix, there exist eigenvectors $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ corresponding to $\lambda_1, \lambda_2, \ldots \lambda_n$ such that

$$E = \mathbf{v_1}, \mathbf{v_2} \ldots \mathbf{v_n} \tag{E.0.3}$$

forms an orthonormal basis of $\mathbb{R}^n$. Equivalently, every real symmetric matrix is diagonalisable by an orthogonal matrix. If this were not the case, this operator would be impossible to implement in a quantum computer since all operations must be unitary in nature.
Let $\mathbf{v}$ be any vector in $\mathbb{R}^n$
Since E is a basis of $\mathbb{R}^n$ we can write

$$\mathbf{v} = c_1\mathbf{v_1} + c_2\mathbf{v_2} + \cdots + c_n\mathbf{v_n} \tag{E.0.4}$$

Where $c_1 \ldots c_n \in \mathbb{R}$ We then calculate $B\mathbf{v}$ as

$$B\mathbf{v} = B(c_1\mathbf{v_1} + \cdots + c_n\mathbf{v_n} \tag{E.0.5}$$
$$= c_1 B\mathbf{v_1} + \cdots + c_n\mathbf{v_n} \tag{E.0.6}$$
$$= c_1\lambda_1\mathbf{v_1} + \cdots + c_n\lambda_n\mathbf{v_n} \tag{E.0.7}$$

Knowing $B\mathbf{v_i} = \lambda_i \mathbf{v_i}$ for $i = 1, \ldots, n$ We can then apply another $\mathbf{v}$

$$\mathbf{v} \cdot B\mathbf{v} = (c_1\mathbf{v_1} + \cdots + c_n\mathbf{v_n}) \cdot (c_1\lambda_1\mathbf{v_1} + \cdots + c_n\lambda_n\mathbf{v_n}) \tag{E.0.8}$$

$$= c_1^2\lambda_1 + \cdots + c_n^2\lambda_n \tag{E.0.9}$$

Using the facto that E is an orthonormal basis of $\mathcal{R}^3$
Since λ is the largest eigenvalue of B we show

$$\mathbf{v} \cdot B\mathbf{v} = c_1^2\lambda_1 + \cdots + c_n^2\lambda_n \tag{E.0.10}$$

$$\leq c_1^2\lambda + \cdots + c_n^2\lambda \tag{E.0.11}$$

$$= \lambda(c_2^2 + \cdots + c_n^2) \tag{E.0.12}$$

$$= \lambda\|\mathbf{v}\|^2. \tag{E.0.13}$$

We finally make the observation that for any $\hat{B}$ according to Equation 1.4.5 will have at most q elements in each row. When considering a positive $\mathbf{v}, \lambda = q$ and for a negative $\mathbf{v}, \lambda = -q$ □

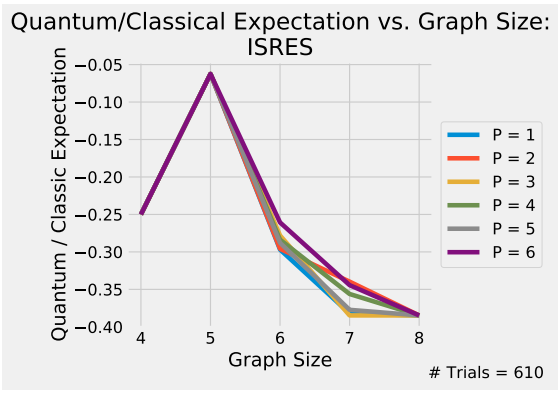
APPENDIX F

Data

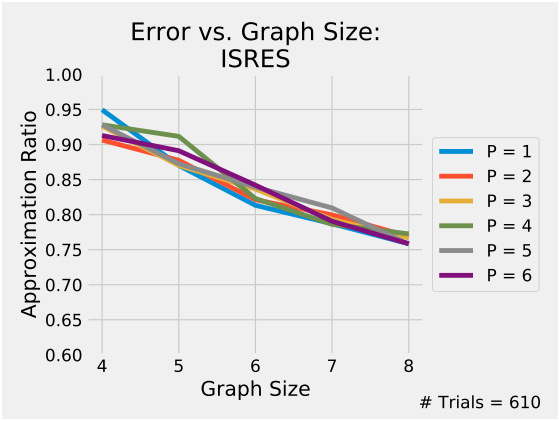
All source data files, plots, csv aggregates and the code used to generate them are available at

`https://bitbucket.org/qaoa_uwa/results/src/master/`

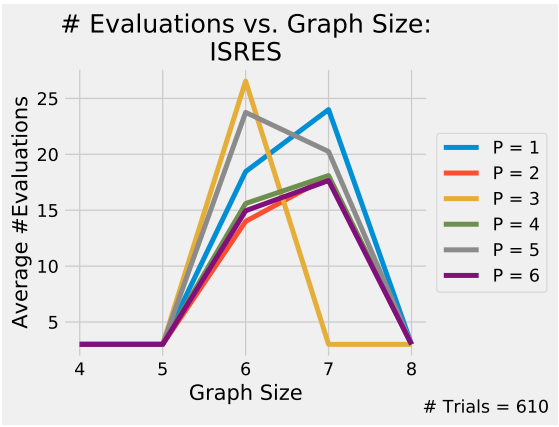
We present the remaining plots for all optimisation methods considered.‘



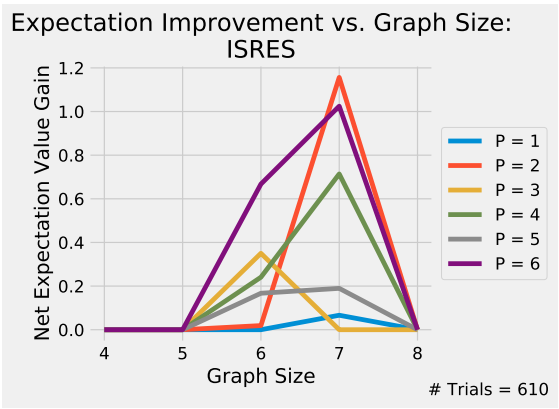
(a) Expectation value comparison



(b) Solution error

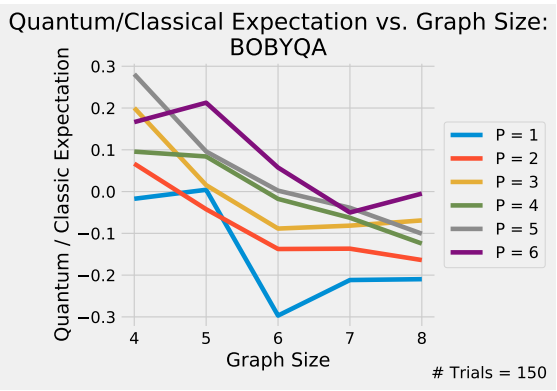


(c) Function evaluations required

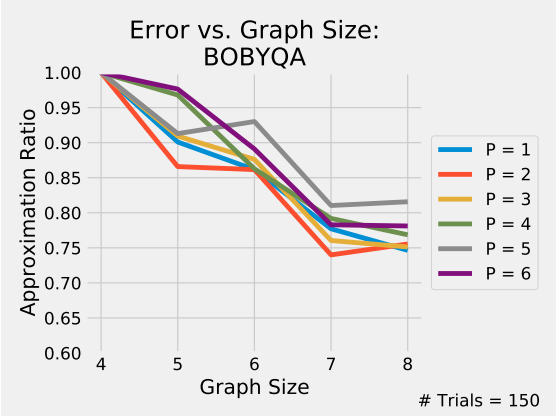


(d) Improvement

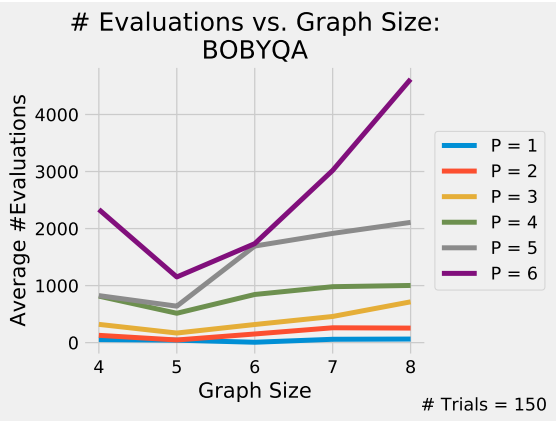
Figure F.1: Final performance metrics for the ISRES algorithm (global) (directed graphs)



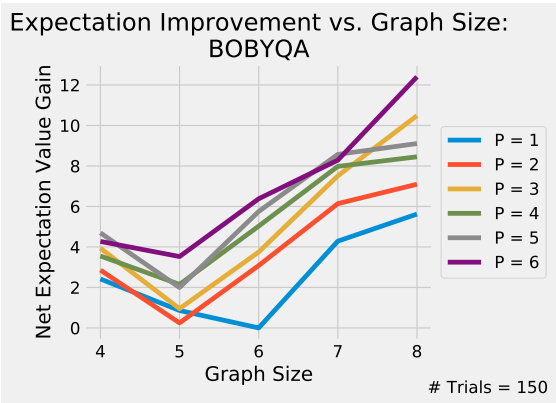
(a) Expectation value comparison



(b) Solution error

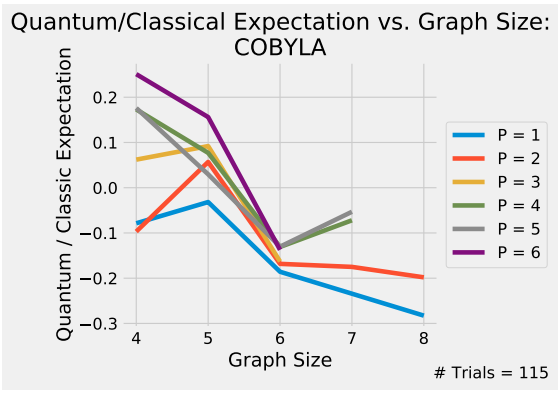


(c) Function evaluations required

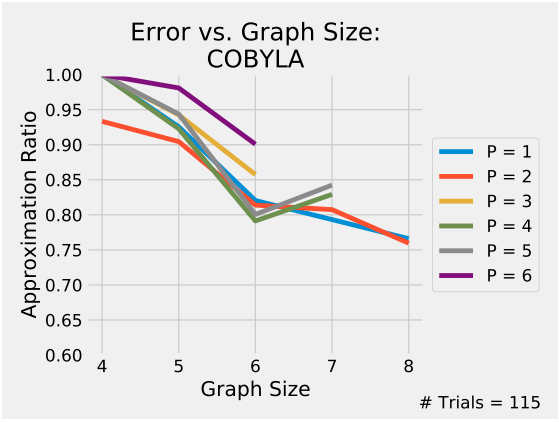


(d) Improvement

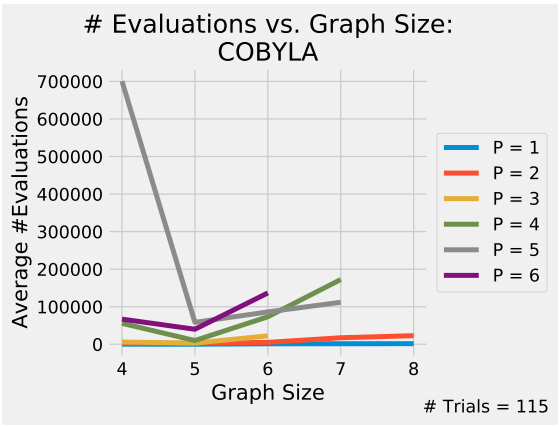
Figure F.2: Final performance metrics for the BOBYQA algorithm (local) (directed graphs)



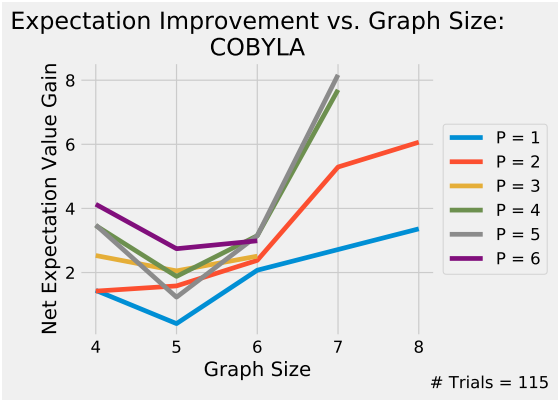
(a) Expectation value comparison



(b) Solution error

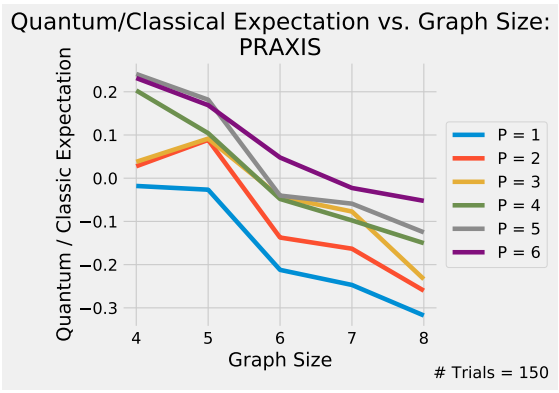


(c) Function evaluations required

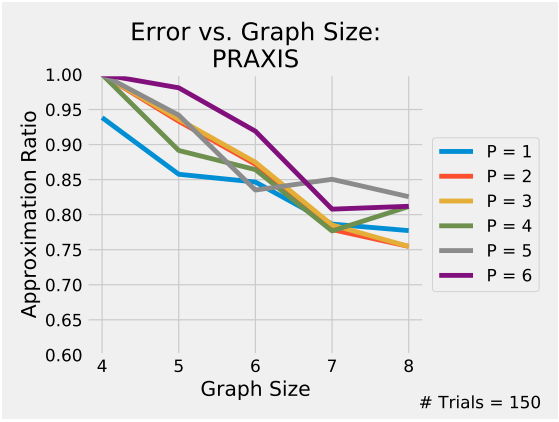


(d) Improvement

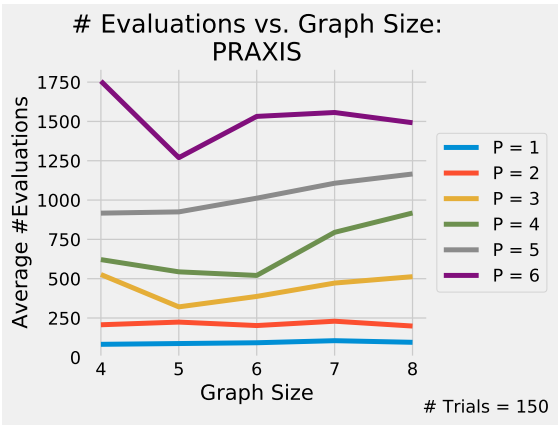
Figure F.3: Final performance metrics for the COBYLA algorithm (directed graphs)



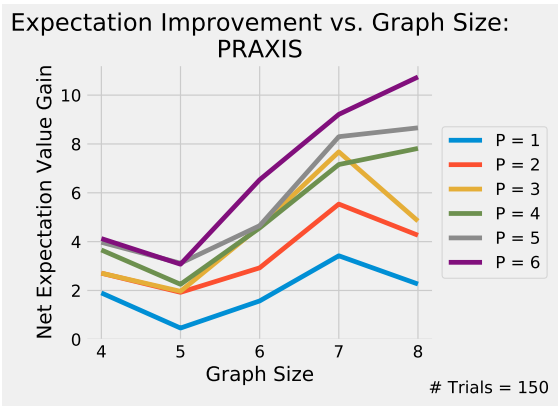
(a) Expectation value comparison



(b) Solution error



(c) Function evaluations required



(d) Improvement

Figure F.4: Final performance metrics for the PRAXIS algorithm (local) (directed graphs)