RESEARCH DISSERTATION

Review of Quantum Optimisation Techniques for the Quadratic Assignment Problem

Author:
Maxine Khumalo (1604282)

Supervisors:
Prof. Ken Nixon
Ms Krupa Prag

UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

A research dissertation submitted in fulfillment of the requirements for the degree of MSc in Computer Science

in the

School of Computer Science and Applied Mathematics

June 7, 2023

Declaration of Authorship

I, Maxine KHUMALO (1604282), declare that this research dissertation titled, "Review of Quantum Optimisation Techniques for the Quadratic Assignment Problem" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this research dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this research dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the research dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 7 June 2023

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

Abstract

Faculty of Science
School of Computer Science and Applied Mathematics

MSc in Computer Science

Review of Quantum Optimisation Techniques for the Quadratic Assignment Problem

by Maxine KHUMALO (1604282)

The Quadratic Assignment Problem (QAP) is one of the fundamental Combinatorial Optimisation Problems (COP), solving many real-life scenarios. However, the exponential increase in CPU time taken to deterministically solve the $\mathcal{NP}$-Hard QAP, as the problem size scales, has resulted in a search for non-deterministic optimisation solution techniques to obtain solutions to the QAP efficiently. The work presented extends and contributes to research into the QAP and using classical and quantum methods to solve it. While previous work has focused on improving classical heuristics, a study has not compared the Variational Quantum Eigensolver (VQE) algorithm and Quantum Approximate Optimisation Algorithm (QAOA) results from the QAP. This paper presents a warm start technique for the VQE and QAOA, allowing a comparison between both methods and the classical benchmark deterministic and heuristic methods. The comparison of the methods uses metrics to measure solution quality, introduced from previous work on this topic. The results show that classical methods are more accurate and time-efficient than the VQE and the QAOA on IBM Quantum systems and simulators. The results illustrate the evolution of the processors and their effects on the VQE. For the QAOA, there is an investigation into the impact of the p-value (a determination of circuit depth). Of the two quantum hybrid heuristics, the VQE is more time-efficient, and the smaller circuit size means solving more significant instances with this warm start method. The QAOA performs better in solution quality over increased problem size than the VQE. This work investigates the comparison of the impact of a new set of basis gates on the quantum optimisation techniques for the VQE; the results did not reflect any consistent relationship. The results show that the VQE has fewer operations than QAOA and that increasing p-values have more complex circuits. Results of a quantum warm-start method imply the QAOA may not maintain higher solution quality for instances larger than size 4, but more results are required. These conclusions contribute to those working to solve COP, particularly those using quantum computers.

Acknowledgements

I would like to acknowledge and thank my supervisors, Ms Krupa Prag and Prof. Ken Nixon, for their guidance and help. Thank you to my peers who provided support and assistance. Thank you to my friends and family for their support and encouragement.

I acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

I acknowledge the Allan Gray Orbis Foundation for supporting me in completing this research and in my academic endeavours.

Computations were performed using High Performance Computing infrastructure provided by the Mathematical Sciences Support unit at the University of the Witwatersrand. Thank you to the School of Computer Science and Applied Mathematics at the University of the Witwatersrand.

Contents

List of Figures

List of Tables

List of Abbreviations

BNB	**B**ranch a**N**d **B**ound
BQP	**B**ounded-error **Q**uantum **P**olynomial-time
COP	**C**ombinatorial **O**ptimisation **P**roblem
NISQ	**N**oisy **I**ntermediate-**S**cale **Q**uantum
QAOA	**Q**uantum **A**pproximation **O**ptimisation **A**lgorithm
QAP	**Q**uadratic **A**ssignment **P**roblem
SA	**S**imulated **A**nnealing
SPSA	**S**imultaneous **P**erturbation **S**tochastic **A**pproximation
SR95	**S**uccess **R**ate at **95** %
SR99	**S**uccess **R**ate at **99** %
VQE	**V**ariational **Q**uantum **E**igensolver

List of Symbols

f, b	flow between facilities matrix, size n by n
d, c	distance between locations matrix, size n by n
c_i	coefficients of the basis vectors, scalars
$\mathbf{z}$	register of qubit, set of size N
$\|\mathbf{z}\rangle$	outer product of the basis states of each individual qubit
H, $H(t)$, H_P	Hamiltonian, matrix size 2 by 2
$V(x)$	potential (dependent on position) , scalar
$\hat{X}$, $\hat{Y}$, $\hat{Z}$	Pauli Matrices, size 2 by 2
E_{Ising}	Ising energy, scalar
π	vector of facility allocation to location, length n
x	matrix of facility allocation to location, size n by n
$\|\psi\rangle$, $\|\psi(t)\rangle$, $\|z_i\rangle$,$\left\|\psi(\vec{\theta})\right\rangle$	general state of a qubit, vector length 2
λ_n	energy (eigenvalue) of the nth qubit, scalar
$\|\Phi_n\rangle$	stationary (eigenstate)
t	time, scalar
n	problem size, number of facilities/locations, scalar
N	number of qubits, scalar
m, x	mass, position (quantum mechanics theory), scalar
h_i	bias parameter

J_{ij}	parameter
s_i	spin, scalar
θ, ϕ	polar angle, azimuthal angle
ω	frequency

Chapter 1

Introduction

The Quadratic Assignment Problem (QAP) is a fundamental Combinatorial Optimisation Problem (COP) with many real-world applications. However, as the problem instances scale in size, the QAP becomes intractable using exact classical computing as its complexity classification is NP-Hard. Quantum computing has specialised algorithms that have the potential to overcome this limitation of the QAP [1], [2]. This research compares the solution quality and computational time performance of popular classical algorithms and quantum algorithms in their abilities to solve instances of the QAP. The quantum algorithms execute on Noisy Intermediate-Scale Quantum (NISQ) devices available via the IBM Quantum Lab, where the technology is limited as it is in its infancy.

This research describes quantum computing and its application to solving the QAP using the IBM Quantum systems and simulators. The exploration depicts the solution quality and computational time metrics as a measure of the performance of quantum algorithms in solving the QAP. The research aims to explore the effects of hardware and software on solutions of the QAP to investigate if quantum optimisation can overcome the complexity constraint of the QAP. This research also explores and attempts mitigation tools to overcome the limitations of NISQ devices.

1.1 Background

This section describes the QAP, the applications of classical computing methods, critical quantum computing building blocks, and applied quantum algorithms used to solve this COP.

1.1.1 Mathematical description of the QAP

In 1957 [3], Koopmans and Beckmann outlined the QAP as a linear programming problem where facilities are allocated to locations such that a quadratic objective function is minimised. The full original mathematical formulation [3], [4] is:

$$\text{minimise} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}c_{\pi(i),\pi(j)} + \sum_{i=0}^{n} a_{i,\pi(i)}, \tag{1.1}$$

where a, b, and c are matrices of dimensions $n \times n$ representing weights for the assignment cost, facilities, and locations. In this formulation, the vector π represents the allocation of facilities to locations and n the number of facilities/locations (i.e. the problem size). Note, in Figure 1.1, where $\pi = [3, 1, 2]$, facility 3 is assigned to location 1, facility 1 is assigned to location 2 etc. In the 0-1 formulation of the QAP, the assignment of facilities to locations is represented by a $n \times n$ permutation matrix, x. The formulation is:

$$\begin{aligned} \text{minimise} \quad & \sum_{i \cdot j=1}^{n} \sum_{k \cdot p=1}^{n} b_{ij} c_{kp} x_{ik} x_{jp} \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{ij} = 1 \quad \forall\ j, \\ & \sum_{j=1}^{n} x_{ij} = 1 \quad \forall\ i, \\ & x_{ij} \in \{0, 1\}. \end{aligned}$$

The QAP allocation, detailed in Figure 1.1, shows where the locations and facilities are allocated and combined to find the final cost. Block 1 is the three locations, and block 2 is the three facilities. There are six possible permutations of facilities to locations ($n! = 3! = 6$). In Figure 1.1 the key shows how distance (d_{ij}) describes the relationship between the locations i and j. Similarly, the flow (a weight appropriate for the specific problem studied) (f_{ij}) describes the relationship between the facilities i and j. The best permutation is the smallest objective function. The objective function is the sum of the products of chosen distance-flow pairs. In the example in Figure 1.1, permutation $\pi = [3, 1, 2]$ is optimal where facility 3 is allocated to location 1, facility 1 to location 2 and facility 2 to location 3. The three distance-flow products in block 3 combine to get the objective function value.

Both Mathematical formulations, the original formulation and the 0-1 formulation, are popular representations in the literature. The classical algorithms use the original formulation, solving for a vector with a permutation of integers (π in Equation (1.1)). The quantum heuristics use the 0-1 formulation, solving a binary decision matrix (x in Equation (1.2)). There exist many other formulations, some of which relax the QAP to ensure convexity [5], [6]. Convexity is but a sufficient condition for optimality. Therefore methods used do not require convexity to find feasible solutions [7].

1.1.2 QAP applications

The QAP is a fundamental benchmark COP. Many well-known problems are simplified formulations or special cases of the QAP - including Travelling Salesmen Problem (TSP) [8], Maximum Clique [9], and Graph Partitioning [10]. The QAP is significant as it aims to optimise relative cost. In real-life problems, relative cost (for example, something might cost more in one location than it would if it were in another location) is more prevalent and challenging to solve than direct cost [3]. It is classified as an

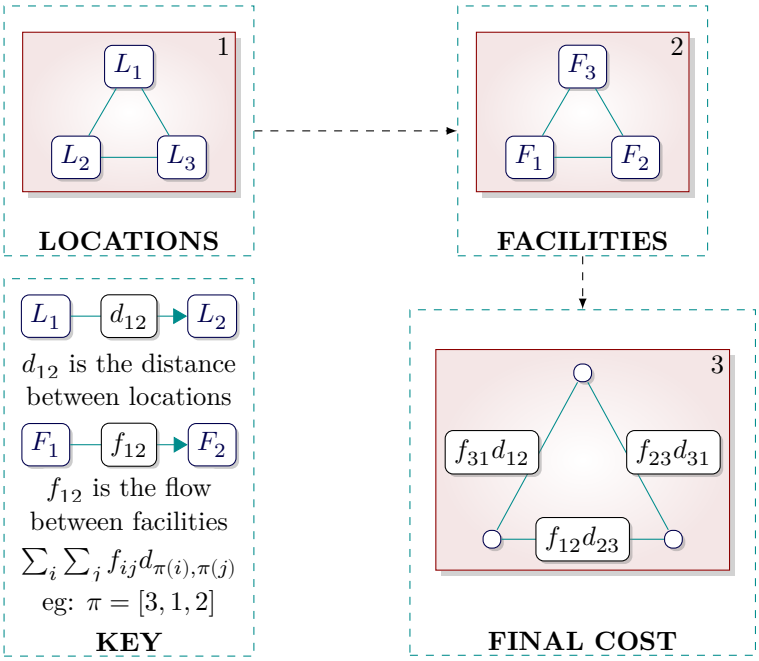


FIGURE 1.1: This figure displays how the implementation of the QAP with an example problem of size 3 (i.e. $n = 3$). In this example, facility 3 is at location 1, facility 2 at location 3, and facility 1 at location 2; hence the permutation vector is [3,1,2]. The key shows the relationship between location and distance and flow and facilities.

NP-Hard problem [11] (note Chapter 1.1.6). The characteristics of the QAP appear in many real-life problems besides facility allocation, including backboard wiring [12], typewriter keyboard design [13], hospital layout [14], campus building arrangement [15] and energy systems [2].

1.1.3 Classical methods to solve the QAP

Many classical algorithms have solved the QAP. The two main classes of algorithms are deterministic and heuristic methods. Deterministic methods are guaranteed to reach the optimal solution of the QAP, but they become intractable for QAP instances of considerable size ($n > 30$) [16]. Heuristic methods can find answers faster, but they do not always find optimal results. In this research, the two classical algorithms implemented compare the performance of the quantum algorithms.

The classical algorithms are Simulated Annealing (SA) and Branch and Bound (BNB). BNB is popular in the literature as it searches the entire solution space but uses strategic bounding to avoid explicitly calculating every feasible solution [17]. This characteristic allows for the certainty of deterministic algorithms without comparing every viable solution (a number that grows exponentially with problem size). Therefore the BNB is less wasteful than an exhaustive search [18]. This benefit is true in problems other than the QAP. SA is a good choice because it is a commonly used heuristic in the literature [4]. SA also works well in solving other COP, getting the best results on the QAP compared to other methods [19].

Branch and Bound

BNB is a deterministic method that is more time-efficient than a brute force algorithm [20]. The kind of BNB implemented is single-assignment branching [21] where each assignment is branched. It is deterministic and starts with a depth-first search of the first permutation (using the formulation in Equation (1.1)) and uses that total to bound the calculation of future permutations [12]. BNB was developed in 1960 to solve discrete optimisation problems classified as NP-Hard (first used to solve the TSP for British Petroleum) [22]. In QAPLIB [19], Branch and Bound has been the most effective method in solving QAP instances built by E.D. Taillard in 1995 [23]. These Taillard instances were solved and proven most effective [24] using a clustering model [25]. The Branch and Bound algorithm is custom designed for each problem it solves, which leads to varying efficiency from problem to problem [17]. BNB is a tree search method, so computation continues far down the tree for most branches if the bounds are too loose. Therefore the algorithm does not bound often, and the computation time is closer to an exhaustive search algorithm [26], [27].

Simulated Annealing

SA is a popular iterative heuristic method for solving non-convex, non-linear COP [28]. SA comes from statistical physics, where annealing is the process where a solid in a heat bath is slowly cooled to thermal equilibrium (the minimum energy point) [12]. The physical annealing process slowly "calms" the chaos of the particles at a high temperature so that they form a structured lattice when they reach their ground state [29]. SA works because of the Metropolis Algorithm [30], [31] which is moderated by a decreasing temperature parameter. This parameter starts high, allowing for test solutions to go in multiple directions and makes SA more likely to find the optimal [10], [32]. This temperature controls the randomness of the algorithm. The temperature decreases slowly to avoid trial solutions remaining in local minima but still converging on or near the global optimal [10]. Included is a Markov Chain, which moderates the temperature [33]. This process ends at "the ground state" (the lowest temperature value allowed), revealing the best solution found [34]. SA was first used to solve the QAP by R.E. Burkard in 1984 [35]. SA was found the best-known algorithm for solving QAPLIB [19] instances with large size ($n = 50$ and $n = 1000$) [34], [36].

1.1.4 Quantum Computing

Quantum computing has existed as a concept for many years but has only recently become a viable form of computing [37]. Where classical computing uses binary classifications of bits to perform computations, quantum computing uses many more classifications of qubits [38].

Quantum computing is possible because of quantum mechanics, where spin states can be represented by vectors [39] as:

$$1 : |0\rangle = \begin{pmatrix} 1 \\ \\ 0 \end{pmatrix}, \qquad 0 : |1\rangle = \begin{pmatrix} 0 \\ \\ 1 \end{pmatrix}. \tag{1.2}$$

The general state of a qubit, $|\psi\rangle$, is a linear combination of these basis vectors such that:

$$|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle . \tag{1.3}$$

The square of the magnitude of the coefficient of a certain basis vector, is the probability of finding such basis vector. Therefore, the sum of the squared magnitudes of the coefficients of the linear combination of the basis vectors equals one, as shown by Equation (1.4) [39].

$$|c_0|^2 + |c_1|^2 = 1 \ , \qquad c_0, c_1 \in \mathbb{C}. \tag{1.4}$$

One of the main ways to visualise the spin states, is with the Bloch sphere shown Figure 1.2; the general qubit state from Equation (1.3) then is transformed to:

$$|\psi\rangle = \cos\frac{\theta}{2} |0\rangle + e^{i\phi} \sin\frac{\theta}{2} |1\rangle . \tag{1.5}$$

Therefore, for example, the values for Equation (1.5) are: $\theta = \frac{\pi}{2}$, $\psi = \frac{\pi}{2}$ and the radius is length 1 which produces Figure 1.2.

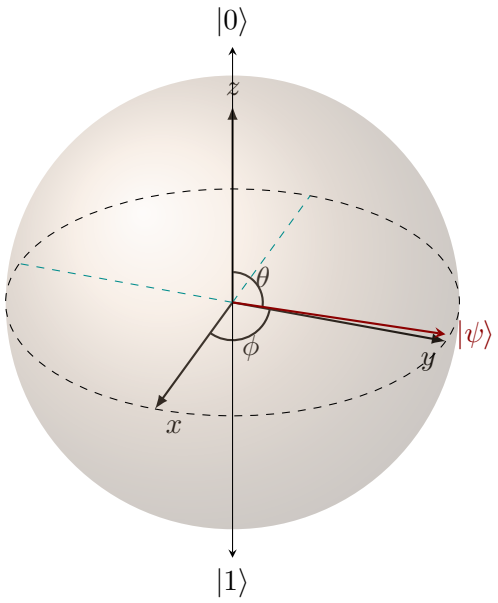


FIGURE 1.2: Bloch Sphere from Equation (1.5)

A Hamiltonian is an important mathematical object in quantum computing described in the time-independent form of the Schrödinger equation (this is the one-dimensional version):

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V(x)\psi = E\psi \tag{1.6}$$

where $V(x) = \frac{1}{2}m\omega^2x^2$ [40]. Hamiltonians represent a model - this can model an optimisation problem or other mathematical problems. The mechanics of quantum computing solves for the eigenstates used to interpret the solution to the model.

An important phenomenon in quantum computing is tunnelling. Tunnelling happens when a particle can move through a potential energy barrier without increasing its energy to do so [41]. If one constructed a potential well in which a particle could not tunnel out of, the Hamiltonian describing it would be: [39]:

$$H_P = -\sum_{i=0}^{N} h_i\hat{Z}_i + \sum_{\langle ij\rangle\in\mathbb{E}} J_{ij}\hat{Z}_i\hat{Z}_j, \tag{1.7}$$

where J_{ij} is an example specific parameter, h_i is a bias and $\hat{Z}$ represents a Pauli matrix [39].

Quantum gates are a structure that takes a set of binary inputs and returns a single binary input. They are represented by matrices of which the most famous are the Pauli matrices - $\hat{X}$, $\hat{Y}$ and $\hat{Z}$ that equal [42]:

$$\hat{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \hat{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \hat{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad . \tag{1.8}$$

A system of gates forms the circuit that the quantum computer solves, and thus Pauli matrices are crucial to solving problems.

The user must first build an Ising model to solve binary optimisation problems with quantum computers. The Ising model, first outlined by Ernst Ising in 1925[43], describes simple phase transitions of particles [44]. They model COP to find a Hamiltonian which quantum computers solve [1]. The Ising energy, E_{Ising}, is a sum of linear and quadratic variables such that [39]:

$$H_P\left|z_1, z_2, ..., z_N\right\rangle = E_{Ising}(s_1, s_2, ..., s_N)\left|z_1, z_2, ..., z_N\right\rangle \quad where \;\; s_i = 1 - 2z_i \tag{1.9}$$

where s_i refers to the spin.

The Hamiltonian is H_P in Equations (1.7) and (1.9) for which the quantum computer solves for the eigenstate and eigenenergy [45]. The eigenenergy and the eigenstates which solve the specific cost function for E_Ising from:

$$E_{Ising}(s_1, ..., s_N) = -\sum_{i=1}^{N} h_i s_i + \sum_{\langle ij\rangle\in\mathbb{E}} J_{i,j}s_i s_j, \tag{1.10}$$

where $s = \pm 1$ [39].

The challenge in solving optimisation problems with quantum computers is encoding the binary form of the optimisation problem in a Hamiltonian solved by the quantum computer [1]. This Hamiltonian dictates the depth and complexity of the circuit (since it is a combination of gates) and is crucial in the computational time and solution quality [46].

1.1.5 Quantum Optimisation Methods

The user must first map COP objective functions to a Hamiltonian form to solve them using quantum devices. The Hamiltonian of Ising model formulation mappings of the QAP, described in Chapter 1.1.4, are presented in this section.

The objective and constraints of the QAP can be described by:

$$\begin{aligned} & \sum_{m=1}^{n}\sum_{u=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} b_{lk}c_{um}x_{ul}x_{mk} \\ + \quad & B\sum_{l}\left(1-\sum_{u}x_{ul}\right)^2 + B\sum_{u}\left(1-\sum_{l}x_{ul}\right)^2, \end{aligned} \tag{1.11}$$

by assigning

$$x_{ul} \to (1 - Z_{ul})/2,$$

where Z_{ul} is a Pauli operator that maps the QAP to a Hamiltonian [1], [2]. A and B are free parameters that satisfy the constraints, and b and c are matrices of dimensions n $\times$n representing facilities and locations, respectively.

At the time of writing, there are two popular ways to perform quantum optimisation - gate-model quantum computers and quantum annealers [2], [47]. This research explains variational methods and implements them on IBM's gate-model quantum computers. Qiskit is a software development kit - the package is in Python with functions that execute on IBM Quantum systems. The two quantum hybrid algorithms that are implemented in Qiskit and have success in COP are the Variational Quantum Eigensolver (VQE) and Quantum Approximation Optimisation Algorithm (QAOA) [48].

Variation Quantum Eigensolver

The VQE is used to solve classical optimisation problems and fermionic atomic problems by finding eigenstates and eigenenergies to given Hamiltonians (H) [49]. To demonstrate how the VQE works, this example objective function is the starting point:

$$\min_{\theta} \quad \left\langle \psi(\vec{\theta}) \right| H \left| \psi(\vec{\theta}) \right\rangle, \tag{1.12}$$

where the trial states $|\psi(\theta)\rangle$ on the quantum computer depend on θ (the classical parameter) [46]. The expectation estimation subroutine within the VQE requires $O(1/\epsilon)$ samples from circuits with depth $O(1)$ [50].

The principle that allows the VQE to solve for eigenstates is the Rayleigh-Ritz Variational Principle. The principle states that for any parametrised trial state $|\psi(\theta)\rangle$:

$$\left\langle \psi(\vec{\theta}) \right| H \left| \psi(\vec{\theta}) \right\rangle \leq \langle \psi(\theta_0) | \, H \, | \psi(\theta_0) \rangle = E_0 \, , \tag{1.13}$$

where E_0 is the ground state [40]. In the VQE, a series of trial states (parametrised by a classical parameter θ) are chosen such that an expectation value is calculated on the quantum computer [51]. The expectation value helps to find the cost function of the optimisation problem. Therefore, the goal is to get an optimal expectation value to get eigenstates [45]. Because of the Variational Principle, selecting an arbitrary trial state either improves or keeps the solution the same - enabling convergence. A classical optimiser built into Qiskit optimises θ such that the VQE converges on the eigenstate [52]. Therefore, the quantum computer sends expectation values to the classical optimiser that sends better trial states to the quantum computer until finding the eigenstate. This cycle makes the VQE an effective hybrid optimiser [51].

Quantum Approximation Optimisation Algorithm

The QAOA was designed in 2014 [53], proving that the QAOA improves with the increasing problem size of the Maxcut problem. The QAOA is similar to the VQE as it also based on the Variational Principle, but the trial state selection is unique. Where the VQE chooses a generic trial function $|\psi(\theta)\rangle$, the QAOA uses a Hamiltonian to build the trial function [46], [54]. A Hamiltonian is used to described a cost function $C(x)$. The QAOA begins by finding a trial state $\left|\psi_p(\gamma, \beta)\right\rangle$ with real parameters γ and β such that the expectation value of the Hamiltonian is satisfied,

$$\min F_p(\gamma, \beta) = \left\langle \psi_p(\gamma, \beta) \right| H \left| \psi_p(\alpha, \beta) \right\rangle \tag{1.14}$$

The trial state, which depends on the parameters γ and β, is prepared on a quantum computer. The measurement of this trial state (in the computational basis) obtains a bit string x^n. Continuous repetition of this process using the original value of γ and β produces an optimal bit string x^*. The probability of minimising the cost function $C(x^*)$ increases with each repetition. The integer p determines the circuit's depth level at the preparation of each trial state. The depth level of the circuit is defined by $p \geq 1$, where the accuracy of the QAOA is said to improve as p increases [55].

The QAOA has no initial variational form. A variational form is an initial circuit used to find the expectation of the first trial state included in the VQE [42]. Therefore, the QAOA is less configured and potentially more effective at converging. The QAOA using quantum computers for finding the parameters means it uses more quantum functions and exposes the results to more error [56].

The classical optimiser is key to optimising the trial state in the VQE and the QAOA, so choosing an appropriate optimiser improves solution quality [42]. Optimisers use different approaches (gradient descent methods, line search methods and

conjugate direction methods, to name a few) and have parameters that can be optimised [42], [46]. Simultaneous Perturbation Stochastic Approximation (SPSA) is a suitable optimiser for large-scale population models, and simulation optimisation [57], [58]. SPSA and other classical optimisers are available in Qiskit, whose successes vary based on their parameters and the number of circuit layers [42].

Advantages and disadvantages

Quantum methods have unique benefits and limitations in solving COP. One of the limitations is error mitigation. Unlike modern classical computing, a theoretical qubit would not get the same result as a physical one [45], [46]. This noise comes from the current limits of hardware and the fact that the qubits interact with the environment [59]. Fortunately, these variational optimisation methods are robust to noise. According to the Variational Principle, solutions should not overshoot the minimum (i.e., infeasible). Secondly, while the ansatz of $\vec{\theta}$ includes the ground state, the minimum is found [45]. However, these quantum methods are black-box heuristics - meaning that finding the optimal solution is not guaranteed [51], [53]. Another disadvantage that hinders quantum methods from optimally performing is barren plateaus [60]. Barren plateaus are a result of using gradient descent methods as classical optimisers (like Analytic Quantum Gradient Descent (AQGD) and SPSA) where the gradient decreases exponentially as a function of the number of qubits [61]. The depth of the variational form increases as the number of qubits required for a problem increases. The result is the parameters of a variational form exponentially increase with the number of qubits that solve the QAP instance. Due to the limited number of qubits currently available on NISQ devices, the size of problem instances is restricted [62].

1.1.6 Complexity Theory

The argument for using quantum computing methods to solve the QAP begins with Computational Complexity Theory. The QAP is classified as an NP-Hard problem [11] which means that as the problem size scales, it is intractable in polynomial time for a classical computer to solve [4]. Figure 1.3 gives an overview on complexity classes. The QAP falls into the shaded section (NP-Hard).

Computational Complexity Theory is the study of using standard measures to quantify the difficulty of tasks and the efficiency required to solve them [63], [64]. Complexity theory describes the difficulty and solvability of problems (note that quantum complexity is distinct from classical complexity) [38]. Non-convexity is also a common characteristic of problems classified as NP-Hard [65]. There exist problems too complex to solve with classical computing methods that quantum computers can solve [66]. It is unknown how quantum complexity fits into the classical model of complexity [67], but a classical probabilistic Turing machine and a Quantum Turing Machine (QTM) can be compared [38], [68]. Between the two, a polynomial-time QTM is more powerful [66], [68]. Search algorithms can compare performance

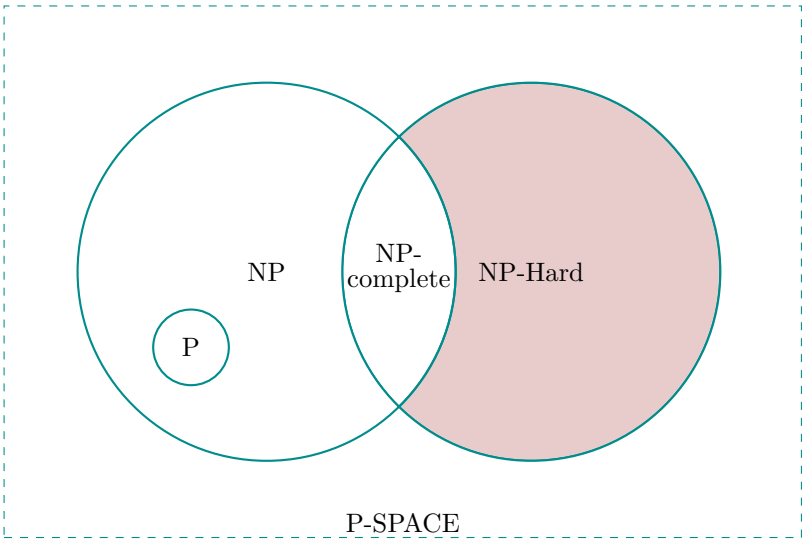


FIGURE 1.3: The visualisation of the sets defined in Computational Complexity Theory

between classical and quantum computers [69]. A basic search algorithm for optimisation problems would be $\mathcal{O}(2^n)$ on a classical computer. There is a quantum search algorithm for solving a problem classified as NP-Complete with $\mathcal{O}(\sqrt{N})$ showing that quantum computing could be more efficient at optimisation than classical computers [69]–[71].

Quantum Complexity Theory uses a family of circuits to class problems as representing polynomial-time quantum computations [67]. The complexity class Bounded-error Quantum Polynomial-time (BQP) is crucial as it represents decision problems solvable by quantum computers [67] (note Figure 1.4). Some problems (integer factoring and discrete logarithm problems) are BQP problems [72]. The QAP has no classification at the time of writing.

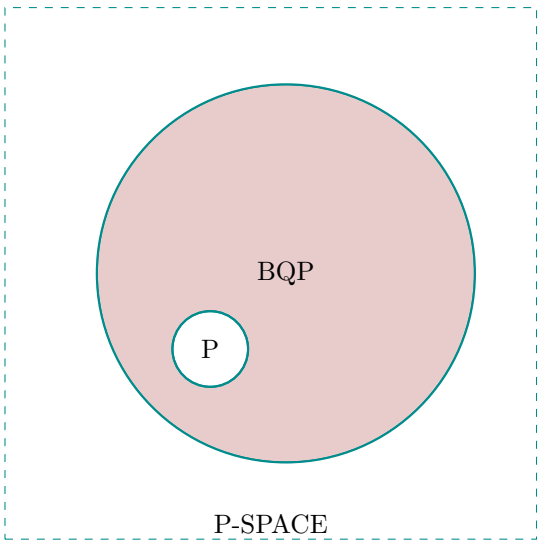


FIGURE 1.4: Image representing Quantum Complexity Theory where BQP is the shaded region - the region solvable by quantum computers

Therefore, there is reason to believe there may be a speedup from quantum computing for optimisation problems, and this approach is one way to investigate the possibility.

1.1.7 Research Problem

The QAP has the complexity classified of NP-Hard, which means it takes polynomial time to optimally solve the problem for larger sizes [4]. The QAP's complexity means the QAP instances larger than 20 locations cannot be solved deterministically using classical methods [12]. The literature suggests that quantum computers cannot solve problems classified as NP-Hard (like the QAP) deterministically [46]. Thus quantum heuristics like the VQE [51], and the QAOA [53] are suitable to investigate on the QAP [46]. Therefore, in this research, two benchmark classical algorithms' computational performance is compared to quantum algorithms available in Qiskit by IBM Quantum. This research investigates quantum algorithms to optimise computational performance using the appropriate parameters and compare which quantum heuristics have the best computational performance when solving QAP. This study also examines the results of quantum approaches to the QAP (like parameters and techniques within the algorithms). As a part of the computational review, this confirms which algorithms work for different types and sizes of problems.

1.1.8 Current Research in the QAP and Quantum Computing

The most recent work on the QAP focuses on new and more impactful heuristics, especially in evolutionary algorithms and natural processes [4]. The Quantum-Inspired Evolutionary Algorithm (QIEA), developed in [73], uses the Genetic Algorithm (GA) but instead uses quantum-like operators to make mutation and crossover functions in the algorithm while using classical computers.

The recent developments in quantum computing in optimisation focus on Quantum Annealers like D-Wave over Gate-Model devices like IBM Quantum systems [2]. This focus on Quantum Annealers is because of greater accessibility to qubits and because they are designed specifically for solving COP [2], [74]. However, with a growing number of qubits becoming available through Qiskit, the VQE and other hybrid algorithms are becoming more viable for instances of the QAP with a small size[42]. The mapping of NP-Hard problems to Quantum Unconstrained Binary Optimisation (QUBO) [39] and to Mixed Binary Optimisation is growing and developing with the tools of encoding [46].

Most of the recent work in the VQE is solving chemistry problems, but there are a few exciting papers in the VQE solving COP. There have been preliminary results using NISQ technology to find solutions to the QAP [2]. There is literature on how to find quality parameters and compare the performance of classical optimisers [75]. There is literature on optimising the measurement of the VQE [76]. The conclusion in previous work [77] (note Appendix B) was that the VQE is less able to solve for

the QAP and the TSP than SA and BNB in terms of computational time and success rate. This research aims to expand on the metrics and results used in co-authored previous work [77] and compare them to those of other algorithms.

There have been many recent developments of the QAOA [55]. A popular computational review includes solving Max-Cut (an important COP classified as NP-Complete) with the QAOA [54], [78]–[80]. The developments to improve the QAOA focus on using graph theory to optimise how the Hamiltonian is solved [76]. The p-value is another critical way to improve how the QAOA is solved [81]. In previous work, [82] (note Appendix B), results for the QAOA solving the QAP were limited due to long computational times. This research aims to expand on QAOA results in co-authored previous work [82] to provide a fair comparison between the VQE and the QAOA.

1.2 Problem Statement

The research compares methods of solving the QAP using classical and quantum computing. Classical methods act as a benchmark for quantum computing methods. The metrics compared are computational time, success rate and other metrics of solution quality. This research builds a mapping of the QAP to a form solvable by a quantum computer. This technique ideally decreases the time taken to solve large instances of the QAP and improves the solution quality of these solutions. However, in quantum computing, the QAP requires one qubit for each discrete variable in the 0-1 representation, which is n^2 [2]. This qubit requirement (a limit of 65 qubits) means only instances of size 8 or smaller of the QAP are solvable ($n^2 = 8^2 = 64$) [80], [83]. This research investigates the parameters and the setup of the quantum methods on the quality of the solution.

1.3 Significance and Motivation

Optimisation in science and business is very important [46] and the QAP is a model that replicates many real-life problems experienced in hospitals [14], wiring [4], and typing [13]. Solving the complexity problem with the QAP is better for these real-life problems because it will result in more accurate solutions, solved quicker. The encoding process is unique to quantum computing and could open more ways to understand COP and integer programming problems. Quantum methods even inspire classical techniques like the QIEA [73]. There is a lot to discover in this new form of computing and, more specifically, quantum optimisation. Quantum computing is also growing in capabilities, increasing access to more qubits and better processors soon [42].

1.4 Research Aims and Objectives

The QAP is a COP used to solve allocation problems. This research aims to use NISQ computers to solve instances of the QAP. The complexity of the QAP means that as the instances increase in size, they cannot be solved within polynomial time by classical computers. This research investigates whether NISQ devices can overcome this complexity limitation by comparing the computational time and solution quality as the problem size increases. These objectives facilitate the mentioned aims:

- outlining basic concepts underlying quantum computing.
- investigating the impact of different hardware architecture between IBM Quantum systems.
- comparing computational time and solution quality performance between well-known classical algorithms and a suite of quantum algorithms.
- describing the application of quantum computing on energy system optimisation problems (focus on QAP)*.

1.5 Research Questions

How do quantum algorithms (such as the VQE and the QAOA) perform on solution quality and computational time metrics to solve the QAP compared to classical algorithms (such as BNB and SA) when solving the maximum problem size possible on NISQ devices and why?

1.6 Delineations, Limitations and Assumptions

This research documents the results of experiments conducted on QAP data. An "instance" of the QAP refers to one set of data with a set "problem size"/"size" (number of locations/facilities and dimensions of the input matrices). In the literature, QAP instances tested for computational performance ordinarily use the QAPLIB [19]. QAPLIB is unusable for the NISQ devices because the smallest instance available is size 12. Therefore, the limitation is that the results found cannot be compared to the literature on QAP that classical computers solved. Fortunately, these random QAP instances are openly available on Github and the QAP instances used in previous work [77].

Available IBM Quantum systems are used along with the Qiskit package in Python [48]. There is a limit to the number of qubits available, limiting the size of the QAP instances that this research can test. There are long queue times for IBM Quantum systems, and the devices often go out of commission regularly. The IBM Quantum

*This objective went unexplored as it is considered an extension of the work developed, and the remainder of the research is not dependent on this aim. Exploring applications in real-life examples is a place for future research.

systems are called "quantum devices" or "NISQ devices" compared to IBM Quantum simulators.

Quantum computers have a significant degree of error (unlike classical computers), so the more noise from the quantum machines, the less reliable the result [37].

The metric of "computational time" described throughout the research refers to the time taken executing the algorithm on its respective device. Therefore, computational time excludes the performance of the grid search for suitable parameters for the heuristics. Additionally, the quantum methods exclude the time taken for the warm-start to find the initial point and time waiting in the queues for quantum devices in the computational time.

This project made two assumptions. Firstly, the computational time tests for the deterministic methods were conducted on the same computer, assuming no other factors influenced their performance. Secondly, the project took that the selected metrics sufficiently captured the accuracy of each method. It thoroughly discusses the strengths and weaknesses of each metric to address any potential bias in the metrics and enhance confidence in their effectiveness.

1.7 Outline

The research unpacks the history of the QAP and how the literature solves it - using classical deterministic and heuristic methods and quantum computing (Chapter 1). It discusses the techniques themselves, the selection of the data and algorithms, and their implementation (Chapter 2). It outlines the preliminary results of all the methods and discusses how the methods perform in terms of computation time and solution quality (Chapter 3). This dissertation argues for the research into the aims described in Chapter 1.

Chapter 2

Research Methodology

This chapter aims to describe all the methods used and specify the approach of each algorithm. Next, this chapter explains how the techniques are implemented and tested by detailing the equipment used and the rationale behind each metric. Discussed are the instances of the Quadratic Assignment Problem (QAP) tested and the limitations of the methodology chosen.

2.1 Algorithm Outline

This section outlines the process and implementation of the algorithms. The notation is consistent with the Nomenclature, and the pseudo-code is Zero-based as it is in Python. The methods implemented are Branch and Bound (BNB), Simulated Annealing (SA), the Variational Quantum Eigensolver (VQE) and the Quantum Approximation Optimisation Algorithm (QAOA). In previous work [77], these implementations of BNB and SA outperformed other classical algorithms. The reasons for the selection of BNB and SA as classical benchmarks is given in Chapter 1.1.3.

2.1.1 Branch and Bound

BNB is a deterministic algorithm with a depth-first search using single-assignment branching (comparing cost functions to the best running total where each assignment is branched). There are many different ways to implement the BNB algorithm (detailed in Chapter 1.1.3). Algorithm 1 shows the details of BNB.

This research uses the formulation of the QAP in Equation (1.1) for BNB. The permutation function π describes the allocation of factories to locations (i.e. [2, 0, 1] represents factory 2 in allocation 0, factory 0 in allocation 1 and factory 1 in allocation 2).

Algorithm 1 Deterministic Branch and Bound (BNB)

Input:

f, d ▷ ***f: flow matrix; d: distance matrix***

Output:

π^*, z^* ▷ ***π^*: optimal permutation vector; z^*: optimal cost value***

initialisation;

n ▷ ***number of facilities***

$perms$ ▷ ***list of all possible π vectors***

no_perms ▷ ***length of*** $perms$

$costs$ = empty vector

$better\_solution$ = infinity

$k = 0$

while $k < no\_perms$ **do**

  $\pi = perms_k$

  $z = 0$

  $i = 0$

  while $i \to 0 : n-1$ **and** $better\_solution \geq z$ **do** ▷ ***Bounding***

    $j = 0$

    while $j \to 0 : n-1$ **and** $better\_solution \geq z$ **do** ▷ ***Bounding***

      if $i \neq j$ **then**

        $z = z + f_{ij} d_{\pi_i \pi_j}$ ▷ ***Cost calculation***

      end if

      $j = j + 1$

    end while

    $i = i + 1$

  end while

  if $i \leq n-2$ **then** ▷ ***Branching***

    $temp = k$

    $temp = temp + 1$

    $temp\_perm = \pi[0 : i]$

    while $temp < no\_perms$ **and** $perms_{temp}[0 : i] == temp\_perm$ **do**

      $temp = temp + 1$

    end while

    $k = temp - 1$

  end if

  if $better\_solution > total$ **then**

    $better\_solution = z$

    $better\_solution\_index = k$

  end if

  $k = k + 1$

end while

$z^* = better\_solution$

$index = better\_solution\_index$

$\pi^* = perms_{index}$

2.1.2 Simulated Annealing

SA is a heuristic algorithm based on the concept of physical annealing, a process of slowly cooling metal to its ground state. This research uses the formulation of the QAP in Equation (1.1) for SA in a similar way to R.E. Burkard in 1984 [35]. Firstly, the fitness function solved in SA is defined to solve the problem in question (Algorithm 2 shows the QAP fitness function).

Algorithm 2 Fitness Function

Input:

1: f, d, π ▷ ***f: flow matrix; d: distance matrix; π: permutation vector***

Output:

2: *cost*

3:

4: **function** OBJECTIVE(f, d,π)

5: initialisation;

6: n ▷ ***number of facilities***

7: $cost = 0$

8: **for** $i \rightarrow 0 : n-1$ **do**

9: **for** $j \rightarrow 0 : n-1$ **do**

10: $cost = cost + f_{ij} d_{\pi_i \pi_j}$

11: **end for**

12: **end for**

13: **return** *cost*

14: **end function**

In SA, a while loop represents a high-temperature value (T) slowly cooling at a constant rate (α). The loop ends at a low stopping temperature (ϵ) that acts like a tolerance value for the final solution [33]. A Markov chain (of length L) aids convergence and, along with other parameter selection, is tailored to the specific problem [31]. The Metropolis Algorithm (shown in Algorithm 3) is what models the Boltzmann curve [30]. These components work together to form SA - shown in Algorithm 4.

Algorithm 3 Metropolis Algorithm

Input:

1: x, y, T, F_x, F_y ▷ x & y: ***permutations;*** T: ***starting temperature;*** F_x & F_y: ***respective function values***

Output:

2: x, F_x

3:

4: **function** MetropolisAlgorithm(x, y, T, F_x, F_y)

5: initialisation;

6: $temp$ = a random value between 0 and 1

7: **if** $F_y < F_x$ **then**

8: $x = y$

9: $F_x = F_y$

10: **else if** $temp < \exp\left(-\frac{F_y - F_x}{T}\right)$ **then**

11: $x = y$

12: $F_x = F_y$

13: **end if**

14: **return** x, F_x

15: **end function**

Algorithm 4 Simulated Annealing (SA)

Input:

f, d, T, ϵ, L, α ▷ ***f: flow matrix; d: distance matrix; T: starting temperature; ϵ: final temperature; L: number of iterations; α: controls the temperature decrease***

Output:

π^*, $cost^*$ ▷ ***π^*: optimal permutation vector; $cost^*$: optimal cost value***

initialisation;

n ▷ ***number of facilities***

x = a random permutation π

while $T > \epsilon$ **do**

  for $i \rightarrow 0 : L-1$ **do**: ▷ ***Markov Chain***

    num_1 = a random integer from 0 to $n-1$

    num_2 = a random integer from 0 to $n-1$

    F_x = OBJECTIVE(x, f, d)

    $y = x$

    swap $y[num_1]$ and $y[num_2]$

    F_y = OBJECTIVE(y, f, d)

    x, F_x = METROPOLISALGORITHM(x, y, T, F_x, F_y)

  end for

  $T = \alpha \times T$

end while

$cost^*$ = F_x

π^* = x

2.1.3 Variational Quantum Eigensolver

The VQE is a quantum hybrid algorithm where the Hamiltonian is calculated on the quantum computer, while the classical computer calculates the ansatz. The Qiskit framework has a library called Qiskit Aqua* that supports the VQE. The parameters associated with the VQE affect its overall performance when applied to Combinatorial Optimisation Problems (COP). The VQE includes many sub-functions used to set up and evaluate the algorithm with the appropriate parameters. This subsection explains the pseudo-code in Algorithm 5 and justifies each step in the algorithm.

Using Discrete Optimisation CPLEX (DOcplex), the 0-1 formulation of the QAP (Equation (1.2)) can be converted into a Quadratic Program using Qiskit (*operator_docplex*). The functionality of DOcplex allows the choice of the circuit on which the preset algorithm executes, the optimiser and an initial point to be the only decisions the user makes.

*The results in this research used this library. Qiskit continues to develop, and Aqua has since been deprecated (although older versions of Qiskit are still available). To find the suite's current design, visit `https://quantum-computing.ibm.com/docs/`

The optimiser used is Simultaneous Perturbation Stochastic Approximation (SPSA), as is the recommendation in Qiskit literature, because it is more precise than other optimisers. The *max_iterations* parameter is adjustable to suit the problem - higher *max_iterations* means more computations.

The variational form is a defining parameter for the VQE. This parameter approximates a circuit that determines the ground state of a Hamiltonian system. The TwoLocal and RealAmplitude circuits are the options used in this research for the VQE.

A simulator with Matrix Product State (MPS) [84] finds an initial point that results from a feasible trial of the VQE. That *initial_point* is an input when the VQE executes on actual quantum devices. The MPS method uses a compact form to formulate two-qubit gate operations, implementing the VQE on simulators [42], [85]. The initial point results in an estimation of solutions in the neighbourhood of the optimal solution. A good quality initial point provides a high probability of converging to the optimal solution - especially for the VQE [45]. This approach is known as a *warm-start*. Classical simulators do not experience as much noise compared to actual quantum devices. An initial point, found through the simulators, improves the performance of noisy quantum devices.

Different backends are available - both classical simulators and quantum devices. The Qiskit package offers a library named Aer, which has multiple classical simulators. Each simulator has different specifications and uniquely simulated features to solve quantum methods locally on classical computers. IBM Quantum systems are accessible with an IBM Quantum account where the commands and results are accessible through the IBM Quantum Lab. Access to IBM Quantum systems varies based on the length of queues, whether a device is undergoing maintenance or if calibration occurs on the device. Each quantum device is unique - with a varied number of qubits, maximum circuits or types of circuit formation.

The number of shots is a parameter available on simulated and quantum backends. Shots represent the number of times the algorithm executes simultaneously on the machine. The VQE function in Qiskit executes with the default number of shots - 1024 (note that the shots provide a distribution of the results).

This implementation of the VQE has a function (Run) that returns all the feasible solution eigenstates and their accompanying probabilities. The VQE results in a distribution of probabilities for a suite of solutions, so this function lists the returned eigenstates that are feasible and their associated probability.

This last function is called FeasibleOutput in Algorithm 5. The following function is a feasibility verifier where the permutation matrix x is checked against the constraints of the integer formulation of the QAP. The feasibility verifier ensures each facility has one location assigned to it and vice versa. The VQE results have various states, each with an associated probability. The most probable, feasible solution is the final solution. Additionally, FeasibleOutput includes the fitness function using

the 0-1 formulation of the QAP, therefore, providing the cost if a feasible output exists.

Algorithm 5 Variational Quantum Eigensolver (VQE)

Input:

f, d, *circuit*, *initial_point*, *max_iterations* ▷ *f:* ***flow matrix;*** *d:* ***distance matrix;*** *circuit:* ***variational form initial circuit;*** *max_iterations:* ***maximum SPSA iterations***

Output:

x^*, $cost^*$ ▷ x^****: optimal permutation matrix;*** $cost^*$***: optimal cost value***

initialisation;

initial_point ▷ ***optimal point from the Aer Backend***

operator_docplex = BuildModel(f, d) ▷ ***DOcplex creates operator***

N = number of qubits of *operator_docplex*

spsa = SPSA(*max_iterations*)

if *circuit* = "RA" **then** ▷ ***Setup the initial circuit***

    var_form = RealAmplitudes(N, entanglement="linear")

else if *circuit* = "TL" **then**

    var_form = TwoLocal(N, entanglement="linear")

end if

vqe = VQE(*operator_docplex*, *var_form*, *spsa*, *initial_point*) ▷ ***VQE setup***

quantum_instance = selected backend and number of shots

result = Run(*vqe*, *quantum_instance*) ▷ ***dictionary of eigenenergies, optimal point, eigenstates and other information about the result***

x^*, $cost^*$ = FeasibleOutput(*result*[*eigenstate*])

2.1.4 Quantum Approximation Optimisation Algorithm

There are many similarities between the two variational algorithms. Qiskit Aqua also supports a built-in algorithm for the QAOA. The QAOA develops its variational form - there is no user-configured variational form at the start like the VQE.

SPSA is the classical optimiser, like in the VQE. The formulation of the operator is identical to the VQE. By keeping these methods as similar as the algorithms allow, this research can make claims about their direct comparison.

The QAOA depends on a p-value, determining the parametrised circuit's depth. The p-value is varied to study its effect on solution quality and computational time.

The warm-start method (implementing the MPS method) is applied to both the VQE and the QAOA to allow direct comparison. The number of shots is another parameter that affects the VQE and the QAOA performance. The QAOA uses 8192 shots (the maximum shots available at the time of results). The QAOA uses more shots than the VQE to counter the effects of the increased circuit size. Shots do not affect computation time and, since the success rate and uncertainty are percentages, have no mechanism to affect solution quality. However, with large circuits getting

0%, feasibility is likely and more shots means that feasibility is more likely to be non-zero. A valid initial point is salvageable as long as one shot is feasible.

The QAOA executes on the same backends (where available) as the VQE for ease of comparison. Identical information captures the *result* value for the QAOA as the VQE, so the same metrics are applicable.

Algorithm 6 Quantum Alternating Optimisation Ansatz (QAOA)

Input:

f, d, *initial_point*, *max_iterations*, p ▷ ***f: flow matrix;*** d***: distance matrix;*** *max_iterations****: maximum SPSA iterations;*** p***: circuit depth***

Output:

x^*, $cost^*$ ▷ x^****: optimal permutation matrix;*** $cost^*$***: optimal cost value***

initialisation;

initial_point ▷ ***optimal point from the Aer Backend***

operator_docplex = BuildModel(f, d) ▷ ***DOcplex creates operator***

N = number of qubits of *operator_docplex*

spsa = SPSA(*max_iterations*)

qaoa = QAOA(*operator_docplex*, *spsa*, *initial_point*, p) ▷ ***QAOA setup***

quantum_instance = selected backend and number of shots

result = Run(*qaoa*, *quantum_instance*) ▷ ***dictionary of eigenenergies, optimal point, eigenstates and other information about the result***

x^*, $cost^*$ = FeasibleOutput(*result*[*eigenstate*])

2.1.5 A quantum warm-start for QAOA

Classical simulators find the initial point for both the VQE and the QAOA because simulators do not experience as much noise compared to actual quantum devices, increasing the performance of the quantum algorithms when applied to quantum devices. There are limitations to using simulators to generate an initial point for solving the QAOA - the circuits get so large that classical computers cannot accurately simulate them. Simulators cannot generate an initial point for instances of $n > 4$ of the QAP. However, quantum devices (seen in Table 2.1) have a better capacity to solve the deep circuits than simulators (specifications in Table 2.2). Therefore, this research also studies a quantum warm-start that uses "ibmq_toronto" to generate the initial point. This device is suitable because it is reservable in advance, and long queue times are less limited. The algorithms look identical to Algorithms 5 and 6 just the *initial_point* input was different and generated from ibmq_toronto and not on a simulator.

2.2 Research Design

This section introduces how this research assessed the performance of the classical and quantum algorithms and the importance of the selected data.

2.2.1 Computational Resources

This research investigates the methods of solving the QAP with multiple algorithms and software. There is the classical computing approach which consists of deterministic and metaheuristic models. Additionally, the quantum computing approach includes different heuristics and optimisers.

An Intel® Core™ i5-6200U CPU @ 2.30GHz 4GB 64 bit Microsoft Windows 10 operating system obtained the classical results on Python 3.6.5. Quantum algorithms are from the Aqua library on the open-source Python framework - Qiskit. The IBM Quantum Lab hosted all quantum experiments across various IBM Quantum simulators and quantum systems using the Qiskit version (0.26.3). All code for all the experiments is available at Github. Note updates and deprecations often occur as Qiskit upgrades and improves. Therefore, techniques and code structure may need adaptation to more recent versions of Qiskit. At the time of writing, Qiskit Aqua is deprecated with similar functions as part of the "qiskit-optimization" library [48].

The IBM Quantum systems have different processors with different functionalities. The naming convention includes the Family (e.g. Hummingbird) that refers to the capacity of the chip (largely based on the number of qubits) and the Revision number that refers to changes over time [42]. IBM Quantum deprecated the Penguin processors as they had a high connectivity [42]. Falcon and Hummingbird processors use a heavy-hexagonal lattice structure which decreases connectivity [86]. High connectivity reduces device performance and gate fidelity [87], [88]. Hummingbird allows for 65 qubits and above systems while Falcon is for 27 qubits or less [42]. Each revision added new features to improve the device performance - thus improving average cnot error, average readout error and quantum volume. The quantum volume is an all-encompassing metric that summarises the impact of connectivity, spectator errors, the number of qubits and others to describe the performance of a quantum device [86].

2.2.2 New Set of Basis Gate on IBM Quantum Systems

Quantum algorithms are often graphically represented as quantum circuits in the literature [79]. Logic gates describe the computational operations that change a state of a qubit in a quantum circuit [42], [89]. Some of the most common and basic quantum gates are the Hadamard gate and the Controlled NOT (CX) gate. The new set of basis gates on the IBM Quantum systems currently consists of the (CX, ID, RZ, SX, X); these changed from (CX, ID, U1, U2, U3) the original set of basis gates on [42]. IBM Quantum launched conditional reset in October 2020 in v0.22 of Qiskit. The recent change to the IBM Quantum gates is a conditional reset. The conditional reset feature initialises qubits in the $|0\rangle$ state through measuring, followed by a conditional NOT (CNOT) gate. Conditional reset allows faster circuit execution by reducing the initialisation time between shots - i.e. a decrease in computation time is the hypothesis. Conditional reset also allows qubit reuse for circuits requiring

TABLE 2.1: Specifications of the IBM Quantum Systems

Quantum Device	Version	Available Qubits	Quantum Volume	Max * Shots	Max * Circuits	Processor Type	Average Readout Error	Average CNOT † Error	Online Date
ibmq_guadalupe	1.3.6	16	32	8192	900	Falcon R4P	1.833e-2	1.041e-2	2021-01-08
ibmq_johannesburg	1.2.2	20	32	8192	900	Penguin R3	8.412e-2	-	2019-09-13
ibmq_boeblingen	1.2.9	20	16	8192	900	Penguin R4	5.258e-2	-	2019-07-03
ibmq_cambridge	1.2.0	27	8	8192	900	Falcon R1.1	12.682e-2	-	2019-11-08
ibmq_montreal	1.9.7	27	128	8192	900	Falcon R4	2.067e-2	1.010e-2	2020-06-03
ibmq_sydney	1.0.37	27	32	8192	900	Falcon R4	3.876e-2	1.291e-2	2020-09-02
ibmq_toronto	1.4.22	27	32	8192	900	Falcon R4	4.253e-2	1.157e-2	2020-06-03
ibmq_mumbai	1.4.5	27	128	8192	900	Falcon R5.1	3.887e-2	9.089e-3	2020-11-13
ibm_hanoi	1.0.0	27	64	8192	900	Falcon R5.11	1.467e-2	1.237e-2	2021-04-24
ibm_cairo	1.0.0	27	64	8192	900	Falcon R5.11	1.160e-2	1.039e-2	2021-05-20
ibmq_rochester	1.2.0	53	8	8192	75	Hummingbird R1	14.448e-2	-	2019-10-01
ibmq_manhattan	1.19.1	65	32	8192	900	Hummingbird R2	3.843e-2	1.541e-2	2020-07-24

* In November 2021 the number of shots and circuits increased.
Most measurements were taken with before this change.
To understand the importance of throughput in quantum computing applications,
read `https://research.ibm.com/blog/circuit-layer-operations-per-second`

† Omitted Average CNOT Error values of decommissioned IBM quantum devices.
Further details of these devices can be found:
`https://github.com/Qiskit/qiskit-terra/tree/master/qiskit/test/mock/backends` and
`https://quantum-computing.ibm.com/services?services=systems`.

TABLE 2.2: Specifications of the IBM Quantum Simulators

Quantum Simulator	Version	Available Qubits	Noise Modelling	Max Shots	Max Circuits	Type
ibmq_qasm_simulator	0.1.547	32	Yes	8192	300	General, context-aware
simulator_statevector	0.1.547	32	Yes	8192	300	Schrödinger's wavefunction
simulator_mps	0.1.547	100	No	8192	300	Matrix Product State

TABLE 2.3: Description of the IBM Quantum Systems Basis Gates

Gates	Description
Controlled NOT gate (CX)	Creates entanglement if target qubit is in superposition
Identity gate (ID)	No operation is performed on a qubit
RZ	Rotation of qubit state around z axis
Square Root of X gate (SX)	Creates superposition on qubit in downstate
X	Changes state of qubit from down state to up state and vice versa

a source of "fresh ancillas". Ancillas moderate the interaction of qubits in a quantum register. Qubit reuse should make calculating circuits more efficient. IBM Quantum implemented state-discrimination logic in control systems to decode qubit states from received microwave pulses. Over 1 μs, the control identifies the qubit states and executes their reuse. Enabling hardware state discrimination negates the necessity for discrimination calibration circuits (these were previously attached to each job). Therefore, the prediction is that the conditional reset should improve the quality of results on the metrics investigated. This research tests this hypothesis using the VQE.

2.2.3 Data

Matrices symmetric over the leading diagonal (a defining descriptor of the problem) and randomly generated make up the QAP instances studied. The matrices are of size 3 until 8 with no basis in reality. Their small size makes them appropriate for trivial testing and benchmarks across deterministic and heuristic methods.

The heuristics parameters (both classical and quantum) are optimised using a grid search before using those optimal parameters for 30 experiments. From this, the collected performance data allows a comparison between the abilities of the different algorithms. The time tests performed on all methods execute on one machine to compare them fairly. The trend for the time test results should show an increasing N^N law because of the order of the permutation function in the deterministic cases.

The results are in CSV files that store the parameters, the computational time and the different result indicators. For the VQE and the QAOA, the files include the full eigenstate of all 30 trials. The eigenstate determines the frequency and feasibility of all 1024 shots for each test. Examples of the files are in the GitHub repository Github. The files names include their method and device. Initial points for the VQE and the QAOA are in similar CSV files. These CSV files determine the values of the metrics shown in Chapter 3.

2.2.4 Research Procedure

The research procedure varies among algorithms, and this section describes them all.

The data (instances of the QAP studied) are randomly generated matrices in Python, with the matrices saved in pairs in CSV files. SA uses QAPLIB instances for sizes 12 and above. These files can be read into Python as a data frame using Pandas, and then it can be used as a NumPy array. The two NumPy arrays (matrix of location distances and a matrix of factory flow value) are inputs in the respective algorithms.

Classical Computing Methods

The deterministic method studied is BNB which is an algorithm coded from scratch. This Python code calculates the computational time as the duration between the

beginning and end of the algorithm. This computational time includes reading in the matrix and the acquisition of the permutations for the problem size. The time to find all the permutations increases with problem size as the number of permutations is the factorial of the problem size by design. BNB has an advantage over a brute force algorithm because it has fewer cost function calculations. Only the small matrices ($n < 12$) are available for testing because the QAP is classified as an NP-Hard problem and therefore is intractable as the size of the instances increases.

SA is the classical heuristic method tested for the QAP. A series of performance metrics are comparable for investigating the performance of heuristic methods. Firstly, a grid search finds the best combination of parameters for the algorithm for each instance of the QAP. SA executes for 30 trials with the best parameters. Computational time is the average over the 30 trials. Success rate describes the accuracy of SA (described in Chapter 2.2.5). The prediction is that the success rate decreases with problem size - especially for the instances in the large group ($n > 20$) - as this is the phenomenon noted in the literature and from the QAPLIB data solutions.

Quantum Computing Methods

To encode the QAP into a form for quantum computing, DOcplex is one way to encode the QAP. DOcplex is a library in Python that solves linear programming problems [90]. DOcplex and Qiskit functions encode the Mixed Integer Linear Programming version of the QAP into a Hamiltonian. These become more complex as the problem size increases because there are many more variables with a slight increase in facilities. The mathematical expression of the encoding is Equation (1.11).

The VQE and the QAOA are the quantum heuristics tested. First, a grid search searches the parameters - *circuit* and *max_iterations* for the VQE and *max_iterations* for the QAOA. The grid search uses the MPS method on the "qasm_simulator" backend supported on Qiskit Aer using a random point and one shot for the VQE and 8192 shots for the QAOA. The QAOA uses the maximum number of shots to mitigate noise. MPS is a simulator method that uses a tensor product and two-qubit gates to solve circuits more effectively. The grid search executes Algorithm 5 for different combinations of different values of the variables in question. The grid search allows us to find the best circuit and SPSA iterations combination for the VQE and the best SPSA iterations for the QAOA. The best parameters generate the best solutions and thus the best optimal point. The best optimal point from the grid search is the initial point for that instance of the QAP. The VQE (Algorithm 5) and the QAOA (Algorithm 6) executes using the optimal circuit from the grid search, one SPSA iteration and 1024 shots on a suite of IBM Quantum systems. This experiment repeats 30 times to evaluate the quality of the solution.

This two-phase (or warm-start) method harnesses the effectiveness of MPS to find the best parameters for the VQE and the QAOA. MPS requires one shot to be the most effective, but this is unnecessary. The more iterations of SPSA, the more likely SPSA finds the correct answer, but the longer the VQE and the QAOA computational

time. By limiting the quantum devices to 1 SPSA iteration, the shortest amount of time is taken to find the optimal solution. The warm-start technique used in this research depends on the simulators to find an optimal point that acts as an initial point on the quantum devices. This research also benchmarks simulators against quantum devices. Simulators have a time limit of 10 000 seconds and have more size restrictions than the largest qubit quantum device. The specifications are displayed in Table 2.2.

The feasible solutions returned come with a probability for the quantum algorithms, and the most probable, feasible solution is the final solution. The algorithm executes with the best parameters for 30 experiments. Comparisons happen over different performance metrics. Because of their overt limitations, only the smaller matrices ($3 \leq n \leq 8$) are solvable by quantum devices. This research compares classical and quantum according to their shared metrics: feasibility, computational time and success rate.

2.2.5 Metrics

This section discusses the metrics used to compare the results. The BNB deterministic results execute once while calculating the computational time. All solutions are feasible and optimal. The SA classical heuristic results include finding the success rates and computational time - all solutions to SA are viable. The best parameters result from a grid search for all heuristics. The quantum heuristics execute on the smallest matrices due to device limitations. DOcplex is used to model the problem as an Ising problem to get the Hamiltonian that is solved to find the solution. Different circuits and optimisers are available to discover which are the most effective. Quantum heuristics results include feasible solutions and associated probabilities from solving the quantum circuit. The final solution is the most probable, feasible solution. Quantum heuristics are comparable on computational time, success rate, feasibility and uncertainty percentage.

Parameters

The classical heuristic SA and the quantum heuristics first undergo a grid search to find the optimal parameters for all problem sizes. With those parameters, the heuristics execute 30 experiments for each instance of the QAP. Optimal parameters find the optimal solution in the least amount of time. Therefore parameters are a trade-off of solution quality over time, and the value of parameters can indicate the complexity of a problem.

Computational time

Computational time measures the CPU's time to solve the QAP instance - measured in seconds. For BNB, the computational time is the only metric measured. Additionally, the measurement of computational time does not include the generation of

the list of permutations used - it is from the first iteration to the last iteration of the BNB.

All three heuristics (SA, VQE and QAOA) exclude the computational time to find the best parameters (the grid search) is d in the computational time. For SA, the computational time is from the first to the last iteration. The nature of heuristic algorithms expands the number of performance measures beyond computational time.

IBM Quantum systems are subject to calibration where a device is taken offline for performance testing. The CPU time for any experiment on a device occurring during calibration is longer than other experiments. As a result, this research records two types of CPU time performances - the average CPU time with outliers (AT) and the average CPU time without outliers (MT). For the VQE and the QAOA, the computational time measures while computation occurs - no queue times are included.

Success rate and feasibility

The success rate metric describes the percentage of the 30 trials within 95% and 99% of the optimal global solution (found deterministically).

The feasibility percentage details the percentage of trials that generate a feasible solution. Feasibility is only recorded for the quantum heuristics since all the SA results are feasible by design. From Algorithm 4, by swapping values in the permutation π, all results are feasible.

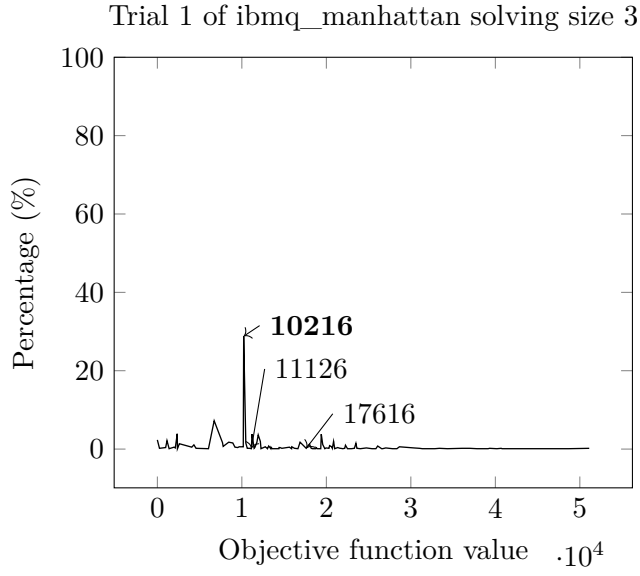
These metrics are referred to as "solution quality" since they describe the objective function value found by the heuristic.

Uncertainty percentage

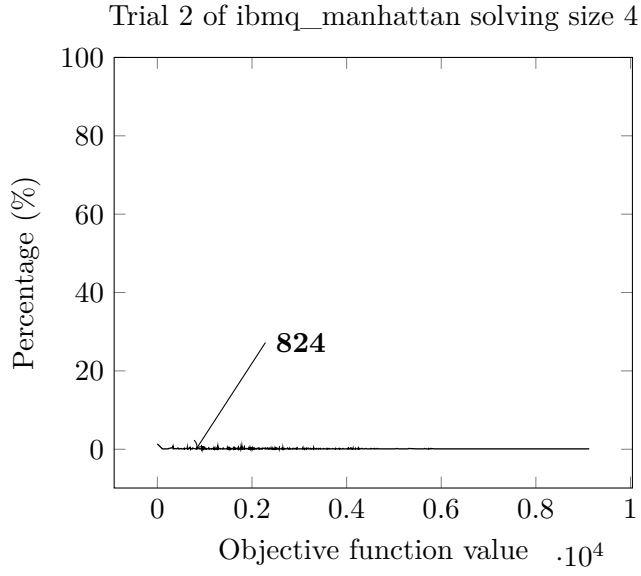
This research also looks into a metric specific to quantum results. By default, the number of shots is 1024 for the VQE and the QAOA. The shots lead to a distribution of all the results obtained for that QAP instance. The optimal solution for a particular trial is the most probable feasible quantum eigenstate in that distribution. This research expands that idea introduced in previous work [77] by considering a new metric called the *uncertainty percentage*. This uncertainty percentage tests the quantum solution's quality by looking at how likely it is to obtain that solution. There is an uncertainty percentage for each feasible solution in 30 trials using the frequency of the quantum eigenstate over the total shots. The uncertainty percentage analysis includes the average probability from all 30 trials, the maximum and minimum probability rates, and the standard deviation (indicated in the headings in Table A.8).

Figure 2.1 describes the uncertainty percentage by using two trials for comparison. The labelled results are the feasible results (three results for Figure 2.1a and one for Figure 2.1b). The other objective function values are infeasible. The value in bold was the solution for that trial. From these two images, in Figure 2.1a the probability of finding the optimal solution is larger since 28.8% of the shots found this result.

The results in Figure 2.1a show the number of shots for each eigenstate at the QAP objective function value. The answer for this trial of VQE is the bold value because it has the highest peak of the three feasible eigenstates.



(A)



(B)

FIGURE 2.1: The result of VQE showing the percentage of each value of all eigenstates (1024 shots total).

2.3 Limitations

The first limitation is the computational time required to complete the algorithms. Because the QAP is classified as an NP-Hard problem [11], as the size of the instances grows, the algorithms take longer to find optimal or near-optimal solutions. This effect is likely to be the most prominent in deterministic methods, but heuristics also are

impacted by the computational complexity of the problem [91]. Algorithms are less comparable in problem size because of the limited quantum results. The patterns concluded by this research result from a more trivial subgroup of QAP instances.

If time is a priority, heuristics may be preferred to solve a problem. The improved speed is at the expense of solution quality, and therefore solutions to QAP instances of large size are less accurate. Therefore, solution quality is a limitation for both classical and quantum heuristics.

The VQE demonstrates limitations linked to the drawbacks of quantum computing. Quantum computers are affected by the environment and are vulnerable to noise and error. Since IBM Quantum systems are accessed remotely and publicly available, queues and internet disturbances can affect the time to complete experiments. The quantum devices demand a minimum number of qubits to solve instances of the QAP that are a problem size. At the time of writing, the number of qubits available is limited. Therefore, the size of solvable QAP instances is limited.

2.4 Conclusion

This chapter outlines all the algorithms with pseudo-code for both classical and quantum methods. Details of these algorithms are available in their entirety on GitHub. This chapter describes the hardware on which the algorithms are executed and lists the hardware's limitations. The QAP instances are randomly generated and sourced from various papers, allowing one to discover the best way to solve certain instances. This chapter details the procedure of executing the algorithms and the metrics used to compare them.

Chapter 3

Results and discussion

This chapter presents the outcomes obtained from applying various approaches to solving the Quadratic Assignment Problem (QAP). The methods employed include classical deterministic techniques, classical heuristics, and quantum-hybrid heuristics. Furthermore, these results are evaluated, considering factors such as success rate, parameters, and computational time. Plots in this chapter are used to compare these metrics between methods, quantum devices and gate architectures.

3.1 Classical Computing

Classical computers have two ways of solving Combinatorial Optimisation Problems (COP) - deterministically and non-deterministically. Deterministic methods are guaranteed to find the optimal solution, while heuristic methods are not guaranteed. These methods were selected as they are commonly used for the Quadratic Assignment Problem (QAP) and can serve as a benchmark compared to quantum computing methods.

3.1.1 Branch and Bound Results

Deterministic Branch and Bound (BNB) is a depth-first search algorithm. The most computationally expensive part is generating all the permutations as lists (it scales as an order of n^n where n is problem size). BNB has a shorter computational time than a brute force method because not all permutations have their cost function measured (note Chapter 2.1.1). Figure 3.1 shows the relationship between problem size and computational time for the BNB on a semilog scale. The number of branching and bounding procedures is dependent on the magnitude of the coefficients in the matrices. From Figure 3.1 it is evident that the relationship is between problem size and computational time is exponential for the QAP. Therefore, the size of the matrix and the matrices' values impact the time taken to solve the QAP with BNB. From Table 3.1, BNB executes for more than 10000 seconds for instances greater than size 10 (making it intractable as the size of the QAP instance scales).

As shown in Figures 3.3 and 3.9, when comparing the computational time taken to solve the QAP using BNB, it is found that the BNB method is faster than the quantum methods for all the QAP instances investigated. Thus, BNB is best for small

TABLE 3.1: BNB: computational time results over problem size (n)

n	**Time (s)**
3	0.000 064
4	0.001 726
5	0.012 494
6	0.037 495
7	0.277 208
8	4.305 403
9	96.247 979
10	2485.746 120
11	72 015.224 627

BNB: computational time results over problem size (n)

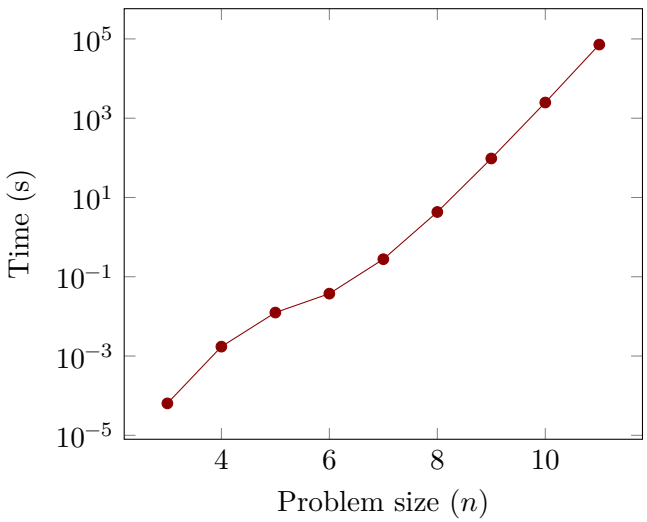


FIGURE 3.1: This is a semi-log plot of BNB time results that shows how the QAP is intractable as the problem size increases.

solutions when prioritising solution quality (success rate). The classical heuristic is more time-efficient than BNB as the problem size increases (note QAP instance of size 7 in Figure 3.3).

3.1.2 Simulated Annealing

The heuristics' results are evaluated based on their success rate. The success rate is the percentage of the total trials, in this case, 30 trials, within 99% (SR99) and 95% (SR95) of the known optimal solutions recorded in the literature. The classical heuristic method implemented is Simulated Annealing (SA).

Analogous to physical annealing, SA aims to take an initial state with the minimum energy using the Metropolis-Hastings algorithm and a Markov chain, so the algorithm converges to the solution. Figure 3.2 shows the success rate of this research's implementation of the SA algorithm. Although the algorithm performs worse when applied to certain problem instances, there is a slight decrease in success with increased size. Table A.2 tabulates the corresponding ground state temperature parameters used for each of the instance sizes and the corresponding success rate results obtained from applying SA to the QAP instances. The ground state temperature parameters in Table A.2 are the final temperature, (ϵ), length of Markov chain (L), cooling coefficient (α) and starting temperature (T). A grid search selects the combination of parameters with the highest success rates in the shortest time. These parameters become less time-efficient as the size of the QAP instances increases (i.e. higher starting temperatures and smaller cooling coefficients). SA can solve the QAP instances of size 12 or larger within 95% of the solution. As mentioned in Chapter 1.1.3, SA has been successful in solving the QAP (instance of size 100) in [34]. The results obtained, tabulated in Table A.2, are consistent with the literature.

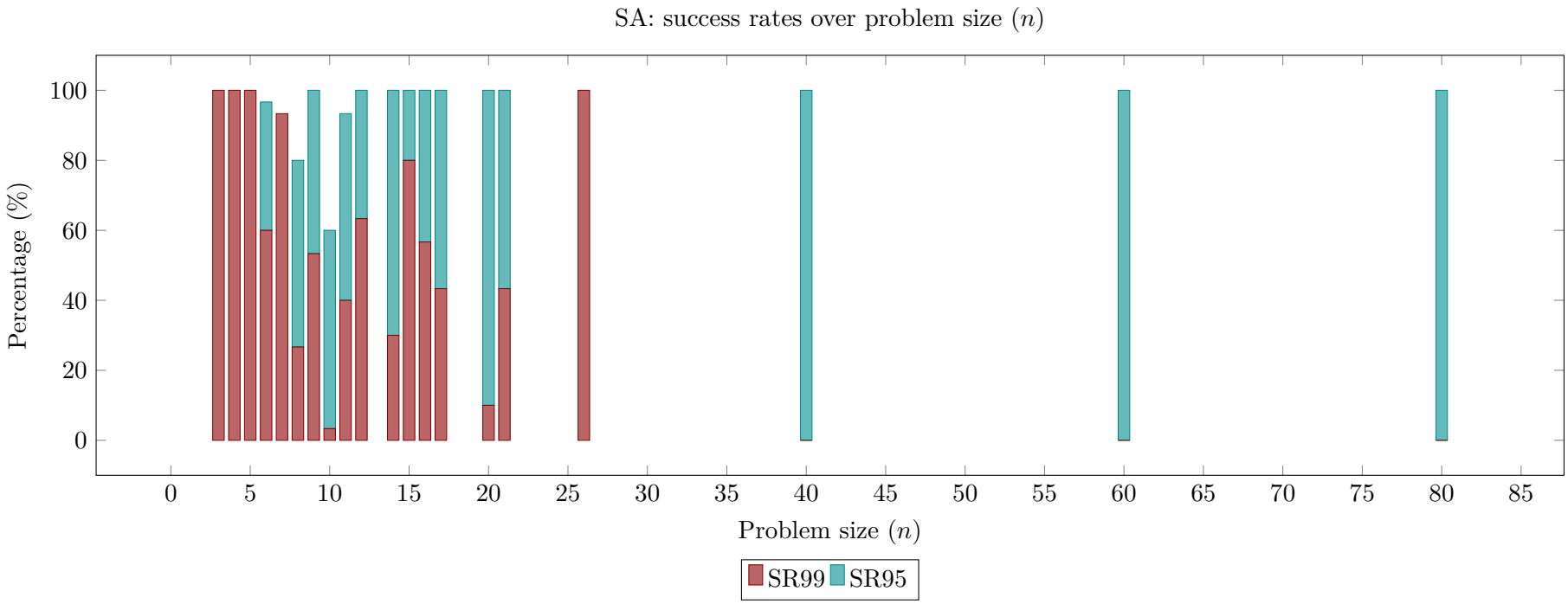


FIGURE 3.2: This plot shows the solution quality of SA for the QAP with a stacked bar graph. Since the SR95 is a relaxation of SR99 (i.e. SR95≤SR99), 100% SR99 is also 100% SR95. This plot shows that SA obtains high SR95 as the problem size increases while SR99 is 0% for the three largest instances.

Computational times are comparable to the other methods using Figures 3.3 and 3.9. SA is faster than the quantum methods for all QAP instances investigated, proving that SA (and potentially other classical heuristics similar to it) are best for solutions where solution quality (success rate) and time is a priority. The largest instances of the QAP ($n = 40$ and $n = 80$) have an SR95 of 100% for SA. This high success is higher than most quantum results for much smaller instances of sizes 3 and 4. Therefore, the performance of classical heuristics like SA is significant because it means that the Noisy Intermediate-Scale Quantum (NISQ) devices have not outperformed classical benchmarks for solving the QAP. SA performs especially well as the size of the instances increases, overcoming the challenges of the QAP's classification as an NP-Hard problem. The classical heuristic has a larger computation time than BNB for the small instances ($n < 7$), but SA is more time-efficient than BNB as the problem size increases (note an instance of size 7 in Figure 3.3).

3.2 Quantum Computing

COP are solved with quantum computers using quantum heuristics. The variational quantum heuristics in this research are hybrid algorithms where classical computers optimise parameters used by quantum computers to solve Hamiltonians. The eigensolutions of these Hamiltonians are optimal solutions to optimisation problems. Hybrid quantum heuristics are the Variational Quantum Eigensolver (VQE) and the Quantum Approximation Optimisation Algorithm (QAOA).

This research obtained the results between 2019 and 2022, over which IBM Quantum updated software and hardware designs on their systems. Quantum computing,

specifically solving COP using gate model NISQ devices like in this research, is a very dynamic field. The structure of the results aims to show the change and discuss its impact on the results. One of the fundamental changes involved the introduction of conditional reset, which used a different set of basis gates. All quantum table headings (Tables A.3 - A.10) and graph captions (Figure 3.3 - 3.12 and 3.14) distinguish between the basis gates concerned. Further analysis of conditional reset and basis gates is in Chapter 2.2.2.

This research includes implementations and applications of the VQE and the QAOA to the QAP. The details of these methods and the pseudo-code are in Chapter 2. A grid search performed on the quantum heuristics and executed on the quantum simulators (Aer backends run locally) finds the best parameters (listed in Table A.11). This grid search, in turn, generates an optimal point of the quantum algorithm - a key parameter for the warm-start. The quantum heuristic is executed a second time on quantum devices with 1 SPSA maximum iteration, and the optimal point found previously as an initial point (note Algorithms 5 and 6). When the quantum heuristic provides a result, it is a series of eigenstates with associated probabilities. The solution is the most probable feasible eigenstate from the 1024 shots at each trial. The number of qubits on a device limits the size of solvable QAP instances (note Table 2.1).

This research evaluates the QAOA and the VQE on many of the same metrics detailed in Chapter 2.2.5. Similar to SA, the success rate is the metric of solution quality (SR99 and SR95) and, similar to BNB, the computational time is a metric of efficiency. There are two computational time measurements in unit seconds - AT is the average time over all 30 trials, and MT is the mean computational time with outliers removed. The plots use the MT value as the outliers distort the data and make it more challenging to compare devices. Unlike the classical heuristic, not all trials are feasible, so the feasibility percentage is a metric. Uncertainty percentage is a test of solution quality that describes the distribution of the eigenvalues over the 1024 shots. The mean uncertainty percentage (in the plots) represents the average of the peaks in the distributions between the 30 trials. The parameter in Table A.11 - number of SPSA maximum iterations - describes the classical optimiser where more maximum iterations require more computational time.

3.2.1 Variational Quantum Eigensolver

The VQE is a hybrid quantum-classical heuristic algorithm where a classical computer calculates an ansatz to minimise the eigenvalue of the Hamiltonian solved with a quantum computer. Results for the VQE are in Tables A.3 and A.4. An additional parameter is an initial circuit ("RA" is Real Amplitudes and "TL" is Two Local). It is unique to the VQE because the user designs the initial variational form.

Comparing the computational time taken to obtain a solution using classical algorithms to that of the VQE algorithm shows the applied classical algorithms had

a faster computational time. Figure 3.3 illustrates that the performance of the various employed quantum devices is consistent in terms of computational time, with the simulators performing better on the whole. There is no distinct correlation between problem size and computational time for quantum devices. However, this claim considers a limited number of QAP instances. There is no CPU time benefit from new basis gates from Figure 3.3

IBM Quantum's NISQ devices have different processors to innovate and improve quantum computers' results. Over time, new devices have more qubits, larger quantum volume and different designs of processors - details explained in Chapter 2.2.1 and Table 2.1. How developments in processors may be effecting the metrics for the VQE are displayed in Figures 3.4, 3.5 and 3.6. Note the processors were developed in the order in Table 2.1 with Penguin as the smallest number of qubits, then Falcon and lastly, Hummingbird. In Figure 3.4, the data is the same as Figure 3.3 but reorganised according to processors. It is evident that Hummingbird devices have the capacity for solving instances with larger problem sizes, and for ibmq_manhattan specifically, the computational time increases exponentially with problem size. There seem to be two groups of time values for the Falcon devices - times between 200 and 600 seconds and between 1000 and 2000 seconds. This bimodal behaviour may identify flaws in the process of removing outliers or the process of recording time taken.

From Figure 3.5, feasibility decreases with problem size. 100% feasibility for all problems of size 3 is a good result for using the VQE to solve small instances of COP. However, since the devices with more qubits have a low feasibility percentage as the size of the QAP instance scales, devices in the future with more qubits might show the same trend. If this trend persists beyond NISQ devices, the VQE will not suit the QAP. For QAP instance of size 4 - only ibmq_johannesburg, ibmq_mumbai and ibmq_montreal have the feasibility of above 50%, and ibmq_mumbai and ibmq_montreal have the highest quantum volume of 128 (Table 2.1). This observation might be a positive indicator for the future of quantum computing solving the QAP. If quantum volume keeps increasing, it may overcome decreasing feasibility with time.

From Figure 3.6, success rate - like feasibility - decreases with increased problem size. Additionally, the success rate is near or at zero for $n > 3$ instances of the QAP. The gap between feasibility and success rate are the trials that got a result larger than 95% of the optimal - perhaps a local minimum or some other sub-optimal result. From Figures 3.6 and 3.2, success rates in SA are higher than or equal to success rates in the VQE for all instances of the QAP, indicating that the SA obtains better solution quality than the VQE. Examples of sizes 6 and 7 perform the worst - with the most operations and most minor time-efficient parameters (note Table A.11).

Simulators of quantum devices are essential tools since NISQ devices experience noise, environmental impact and limited computational power. Although broadly limiting in problem size, simulators can be a window into the future of NISQ devices.

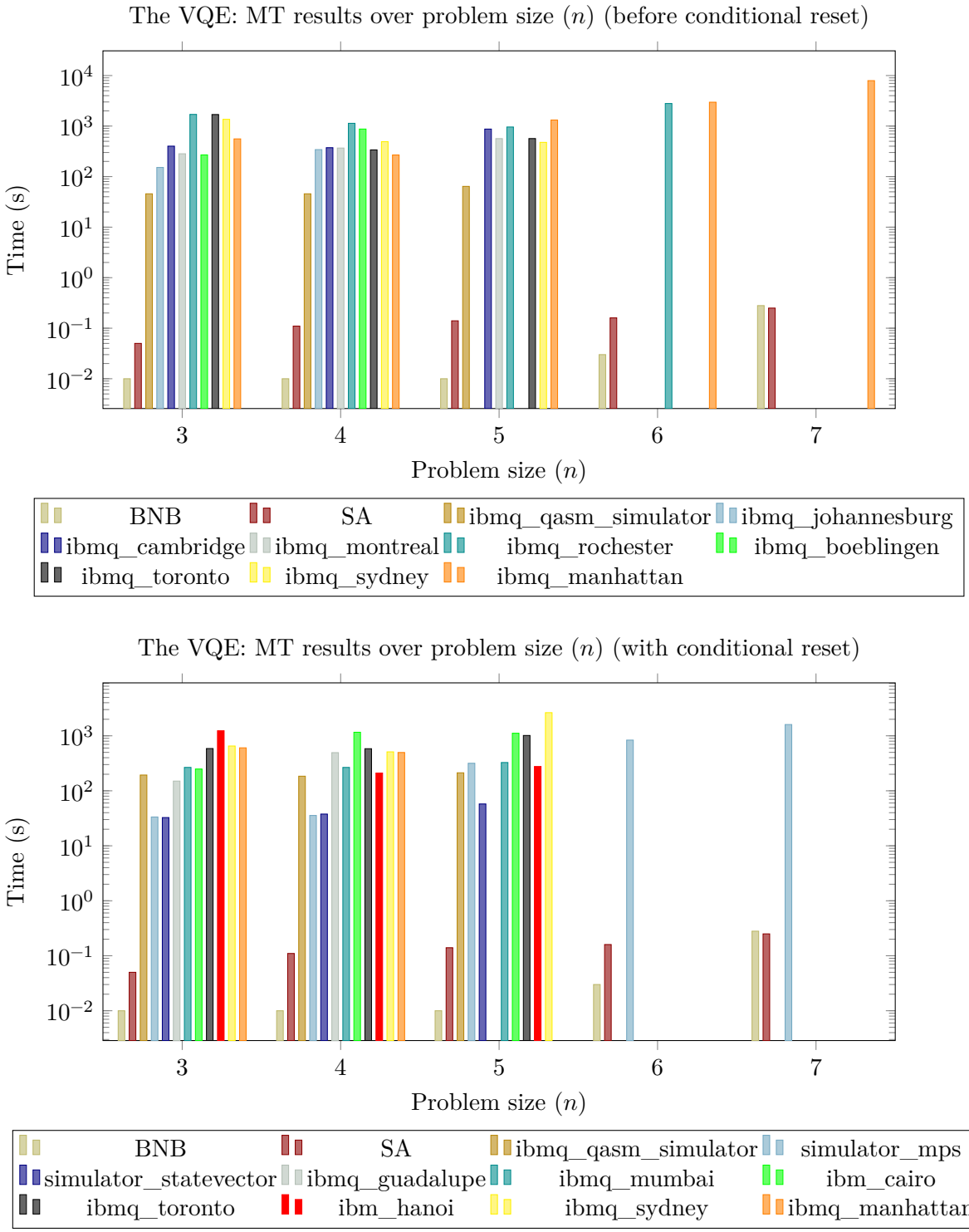


FIGURE 3.3: These plots show the CPU times without outliers (s) for the VQE (Log scale) solving the QAP over problem size (n) before and with conditional reset. Both plots show that classical methods solve the QAP in a shorter computational time than the VQE on all devices. Additionally, no individual device outperforms the others with no clear difference before and with conditional reset.

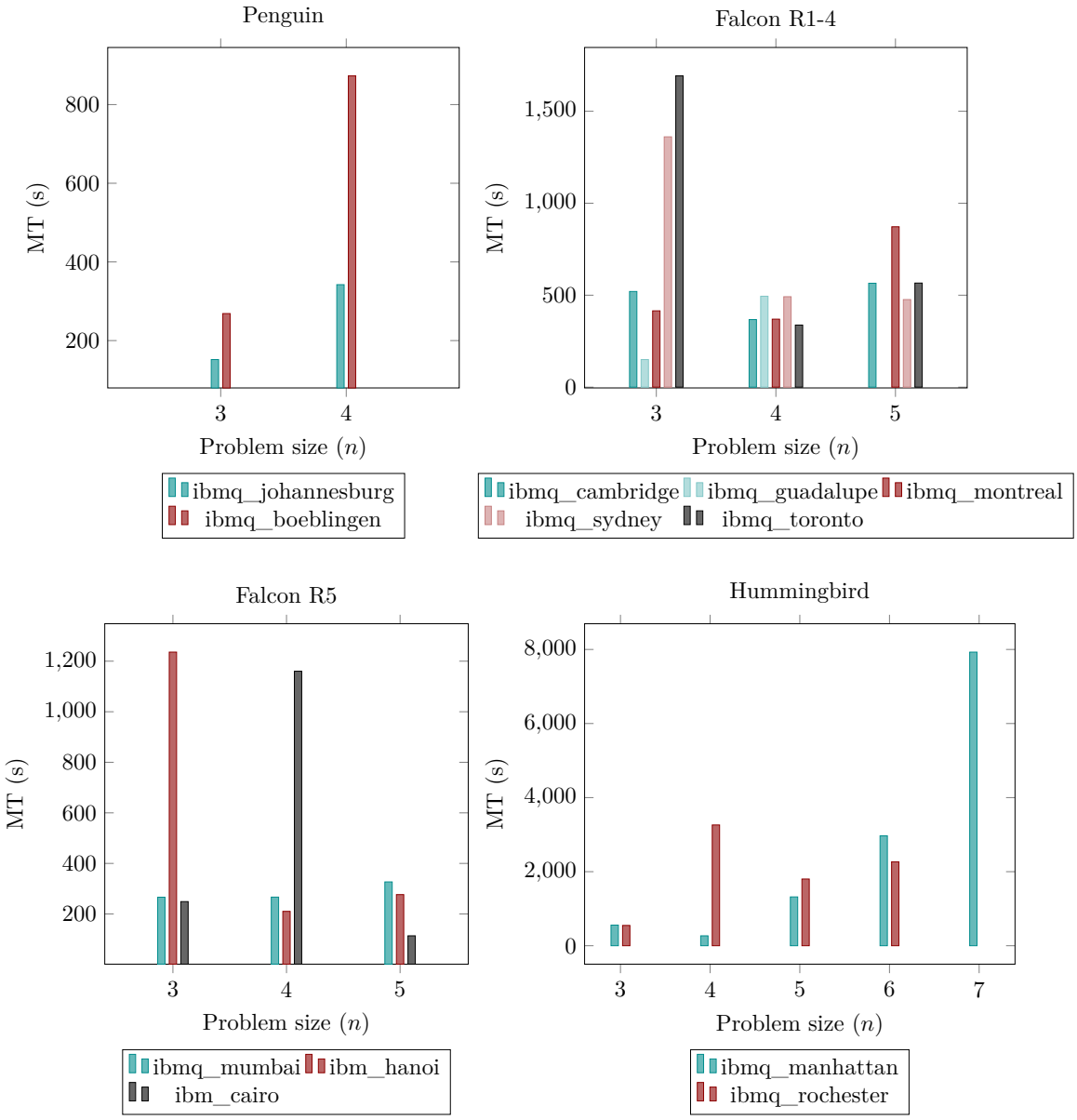


FIGURE 3.4: These plots illustrate the MT (CPU time without outliers in seconds) results for the VQE solving the QAP for different IBM Quantum processors with conditional reset. This figure demonstrates no clear relationship between processor type and computational time.

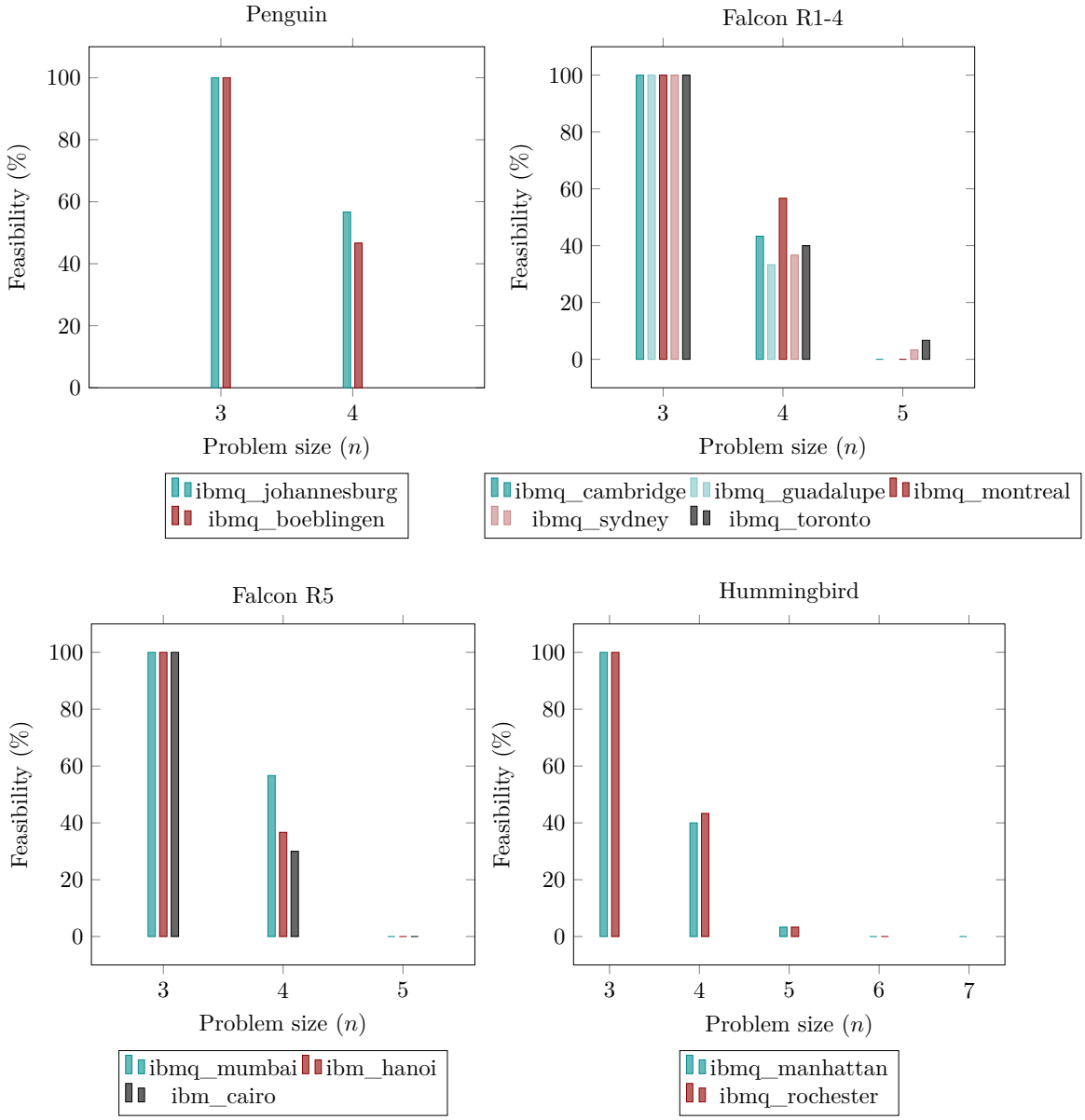


FIGURE 3.5: These plots illustrate the feasibility (as a percentage of the 30 trials) of the VQE results solving the QAP for different IBM Quantum processors with conditional reset. This figure shows that feasibility decreases with increasing problem size (n). Feasibility results are similar between processors for the instances of the same problem size.

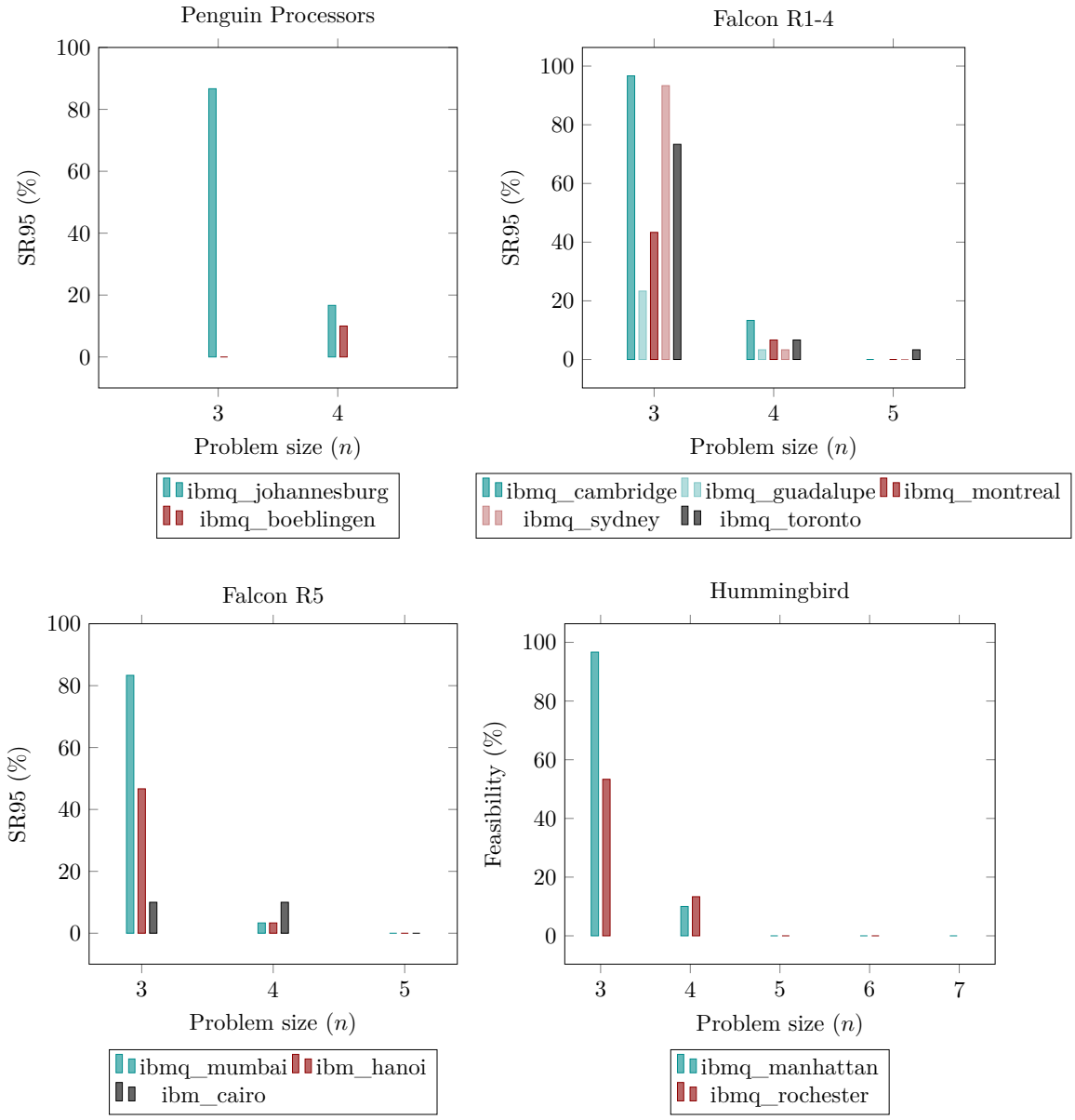


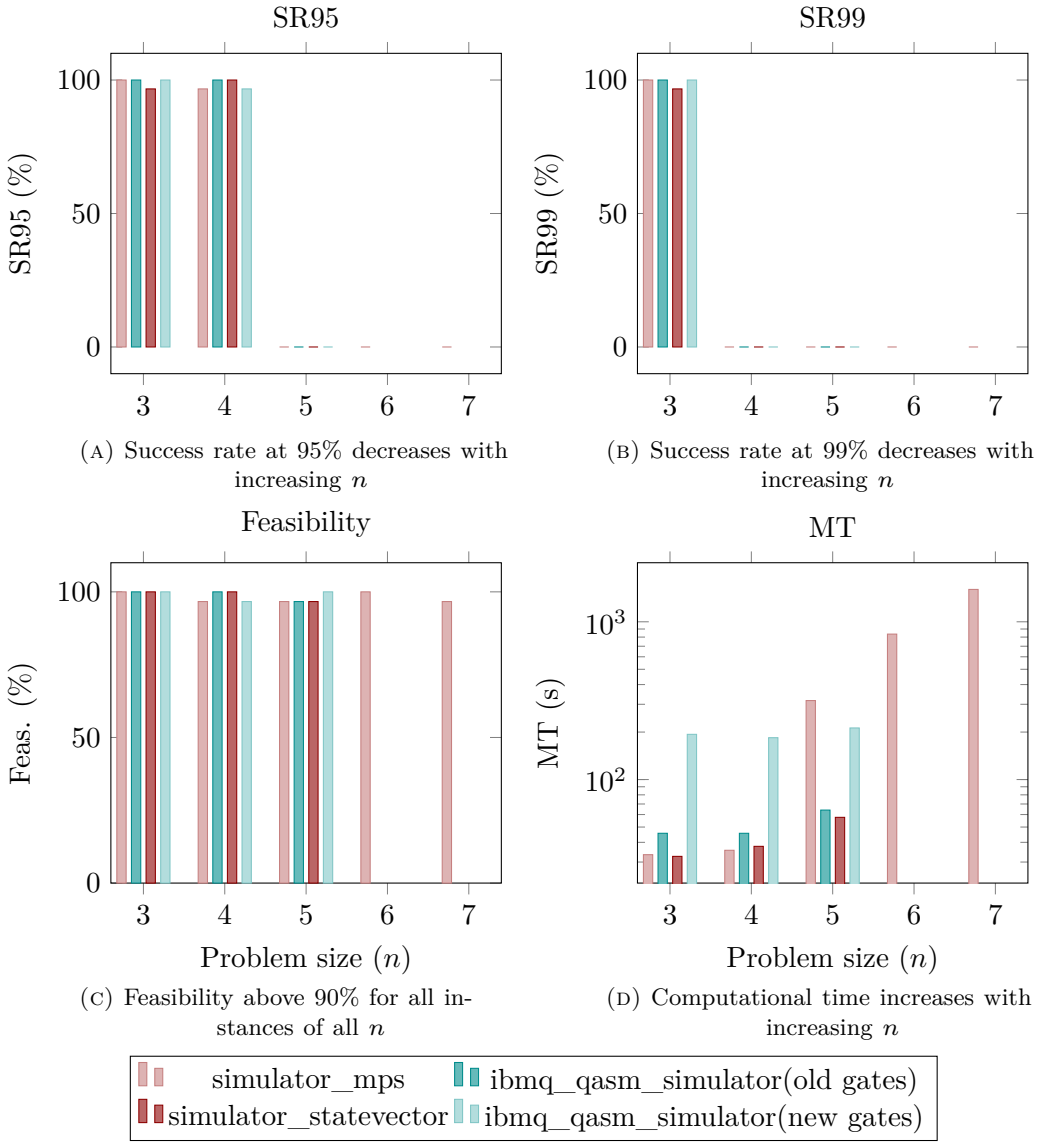
FIGURE 3.6: These plots illustrate the SR95 for the VQE solving the QAP for different IBM Quantum processors with conditional reset. They show, with some exceptions, that the success rate decreases with increased problem size (n) across all processors.

The simulators shown in Figure 3.7 are accessed through the IBM Quantum Lab and not locally on the CPU like the Aer backends used for generating the initial point - note Table 2.2 for details on each simulator. The success rate is bimodal - above 95% for size 3 at SR99 and SR95 and 4 at SR95 and 0% for all the larger instances ($n = 5, 6, 7$). The simulators' success rate results are better than all the quantum devices except ibmq_toronto, which had 3.33% at SR95 and SR99 for size 5. The bimodal success rates and almost unanimous feasibility might suggest that the initial point may be instructive of the results from simulators. The initial points were optimal points for a feasible result but not necessarily a point within 99% or 95% of the optimal. If this phenomenon is evident in the future of quantum devices, feasibility is close to guaranteed with this approach, but improvements in success rate are not. MT increases with problem size (it appears exponential for simulator_mps).

Figure 3.8 and Tables A.5 and A.6 show the uncertainty percentage metric - details are explained in Chapter 2.2.5. Uncertainty percentage describes the quality of the feasible trials obtained using quantum heuristics. Quantum circuits do not return a single answer but a series of eigenstates with corresponding frequencies. The corresponding percentage describes the likelihood of the most probable feasible solution resulting from a given trial. This percentage is unique to the success rate, as it measures the uncertainty of the current trial and not the probability of success for the heuristic. The uncertainty percentage is less than or equal to 1% across all the QAP instances for quantum devices except ibmq_cambridge and ibmq_manhattan for size 3. A low uncertainty percentage indicates that most of the returned eigenstates are infeasible. All the simulators have an uncertainty percentage above 90% (except for simulator_mps at $n = 7$). Therefore, the trend in the results highlights the detriment of noise to solution quality and accentuates the gap in performance between simulators (and the ideal solutions from theoretical quantum computers) and the applied algorithms' performance on NISQ devices. The larger the problem size of a QAP instance, the lower the uncertainty percentage in the results from the VQE. Uncertainty percentage is an average of all feasible trials, so QAP instances without feasible results have no uncertainty percentage. The larger the mean uncertainty percentages appear to have large standard deviations. This phenomenon results from clear peaks at optimal or feasible results instead of the flat distribution where every shot has a different eigenstate (note Chapter 2.2.5).

3.2.2 Quantum Approximation Optimisation Algorithm

The QAOA is a hybrid quantum-classical heuristic algorithm similar to the VQE, but the quantum computer also evaluates the ansatz used by the quantum computer. Chapter 1.1.5 details how QAOA works. QAOA results are tabulated in Tables A.7 and A.8. An important parameter is the p-value - an integer value that dictates circuit depth and is unique to the QAOA. The hypothesis is that the higher the p-value, the higher the solution quality while increasing computational time (note Chapter 1.1.4).



(A) Success rate at 95% decreases with increasing n

(B) Success rate at 99% decreases with increasing n

(C) Feasibility above 90% for all instances of all n

(D) Computational time increases with increasing n

FIGURE 3.7: These plots illustrate the SR95, SR99, feasibility and MT results for the VQE solving the QAP for different IBM Quantum simulators. Results between simulators are very similar except that ibmq_qasm_simulator has a later computational time with conditional reset than before it was introduced.

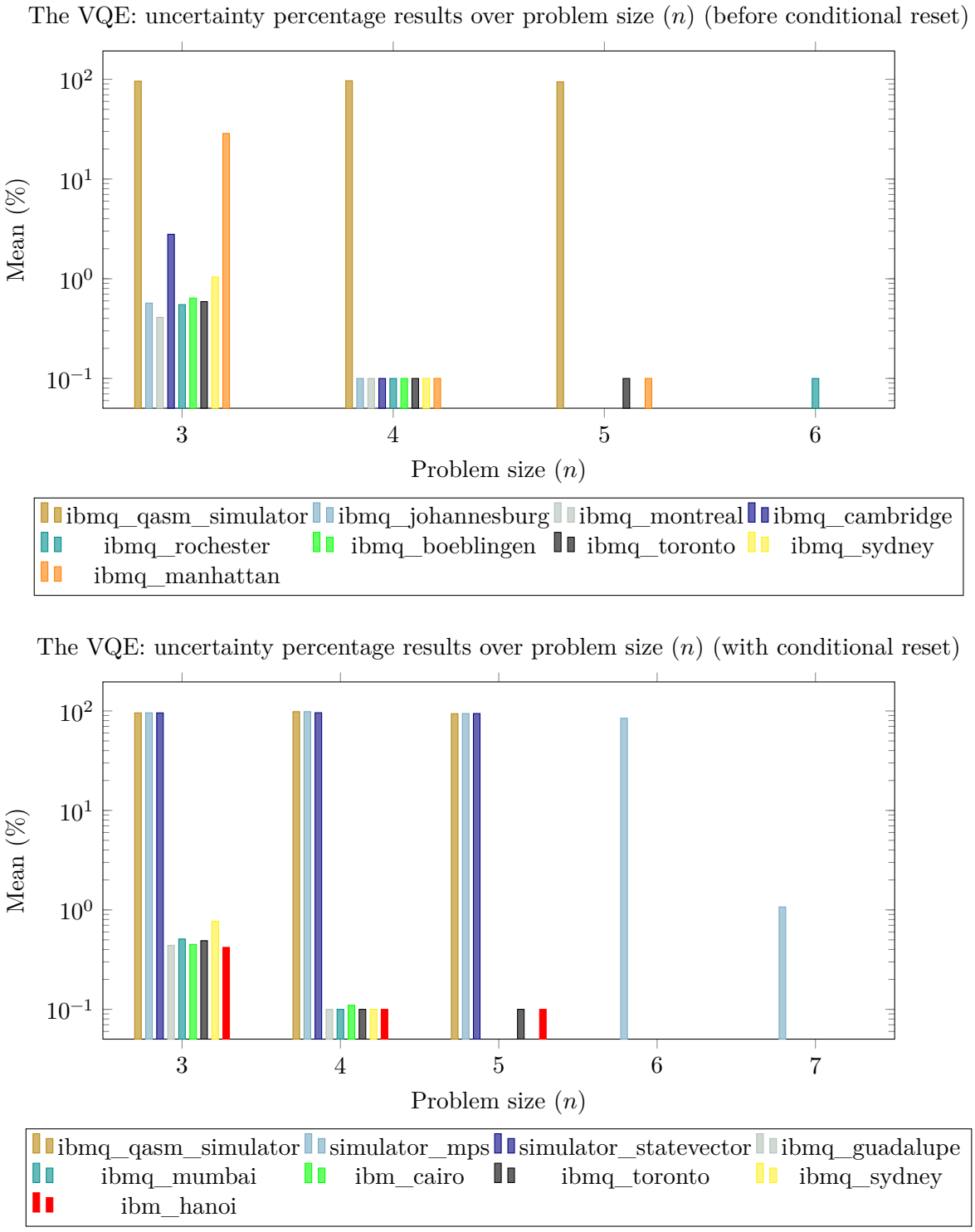
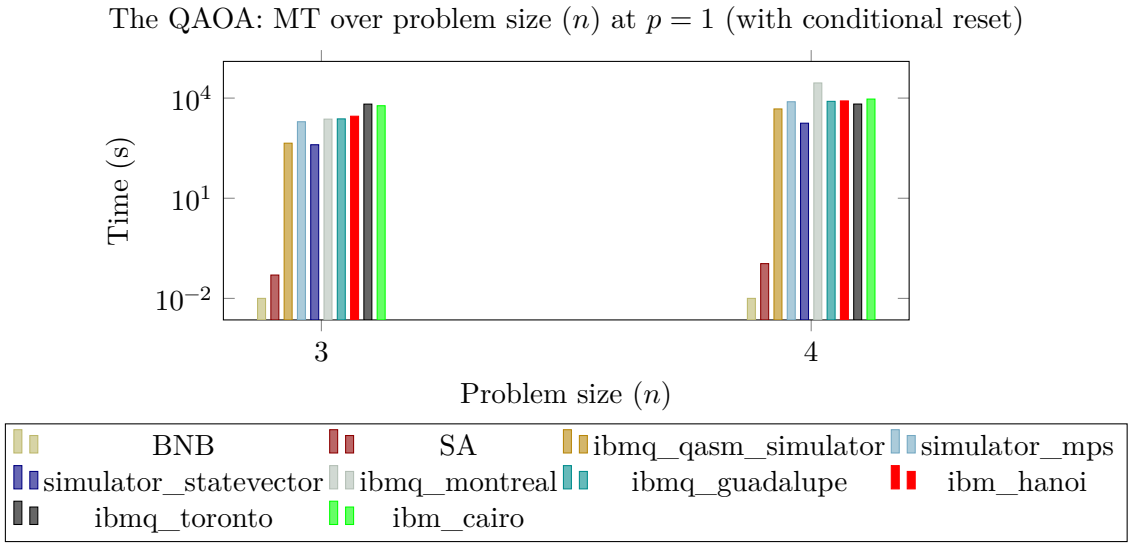


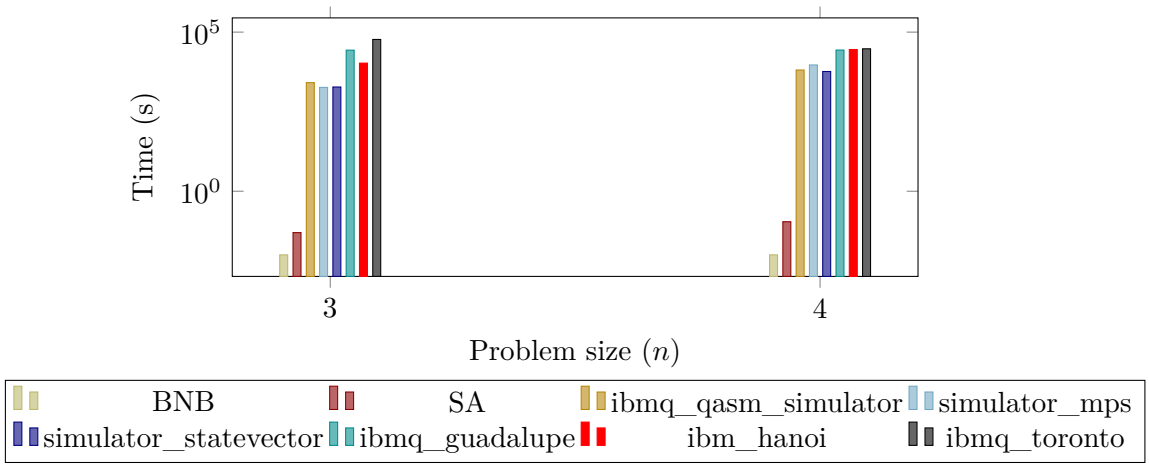
FIGURE 3.8: These plots show the mean uncertainty percentage for the VQE solving the QAP (Log scale). Simulators outperform the quantum devices over problem size (n). If no trials are feasible, there is no uncertainty percentage so some omitted values depict infeasibility.

Classical algorithms' computational time to obtain a solution is faster than the QAOA. Figure 3.9 illustrates that the performance of the various employed quantum devices is consistent in terms of computational time. In contrast to the VQE, the simulators (essentially) perform similarly to the quantum devices in terms of computational time. Simulators have to simulate an additional quantum process (the quantum ansatz) for the QAOA, which explains why the simulators have a longer computational time solving the QAOA than the VQE. The VQE uses the classical computer for this process, making it easier to simulate and explaining why the VQE performs better on the simulators than the QAOA. There is no evidence of a relationship between problem size and computational time over the limited number of tested problem sizes. Most QAP instances increase in CPU time from size 3 to 4. The VQE is faster than the QAOA on average; the VQE CPU times are predominantly between 10^2 and 10^4 seconds, while the QAOA CPU times are between 10^4 and 10^5 seconds (depicted in Figure 3.3, Figure 3.9, Table A.3 and Table A.7). There is no apparent increase in computational time with increased p-value - this contradicts the hypothesis that CPU time is strongly dependent on circuit size.

This research uses the same warm-start method on both the VQE and the QAOA. The warm-start approach means a good quality initial point, found using the classical simulators, is an input when executing the QAOA on the quantum devices. Simulators cannot implement the warm-start for instances of a size larger than 4 because the size of the circuits generated by the QAOA exceeds their capacity (note Table A.11). Therefore results are limited instances of sizes 3 and 4. Additionally, the QAOA's computational time is $10 - 100$ times larger than the VQE's. This computational lag of the QAOA, paired with the rapid development and sudden decommissioning of IBM Quantum systems, means that not all the QAP instances have results for all the p-values for the QAOA. Therefore two devices (one simulator and one quantum device) are plotted over the critical metrics to aid in the discussion of trends in the results in the tables in Appendix A. Shown in Figure 3.10 is ibmq_toronto - the only device with more than nine qubits that could be reserved. Therefore, this device is most accessible because there is no time spent waiting in queues like the other devices. Figure 3.11 displays ibmq_qasm_simulator, which was the first simulator on IBM Quantum Lab, allowing for the most available run time.

For Figure 3.10a, there is no clear relationship between success rate, p-value and problem size. Table A.7 shows no p-value has the highest success rate over the other devices. The results for $p = 1$ are the best of all the other p-values for this subplot. The assumption that a deeper circuit would provide a higher success rate in quantum devices is invalid from Figure 3.10a. For QAP instance size 4, all the QAOA results performed better than the VQE. This trend, shown in ibmq_toronto in Figure 3.10a, continues with other quantum devices shown in Tables A.3, A.4 and A.7. The higher success rates for instances of size 4 may suggest that the QAOA does not have decreasing success rate with the increasing problem size that the VQE has (note with only 2 data points). Therefore, the QAOA may be a better performing





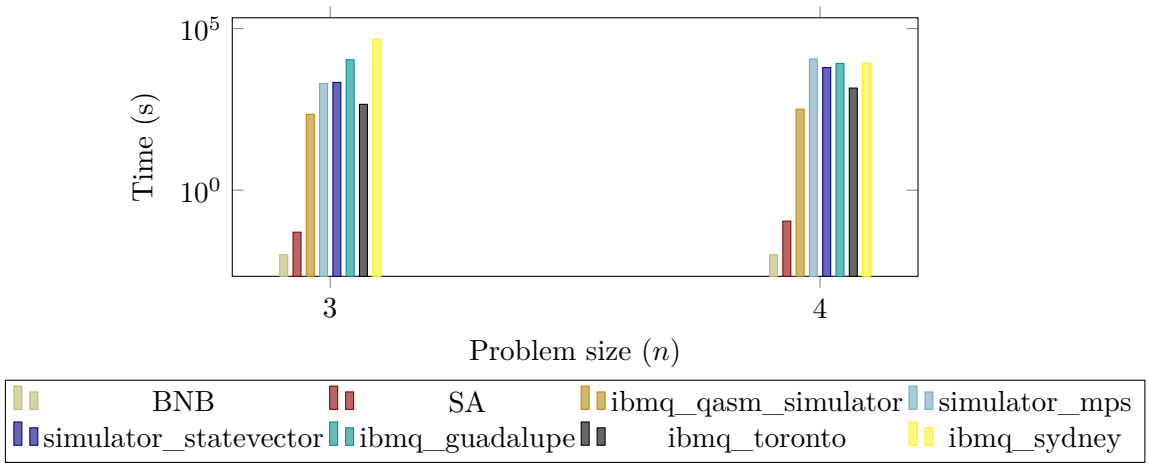


FIGURE 3.9: These plots illustrate the CPU times without outliers (MT measured in log scale seconds) for the QAOA at varying p-values solving for the QAP with conditional reset. These plots show that the QAP is solved with a shorter computational time by classical methods than the QAOA. There are no clear relationships between p-value, problem size and computational time from this Figure.

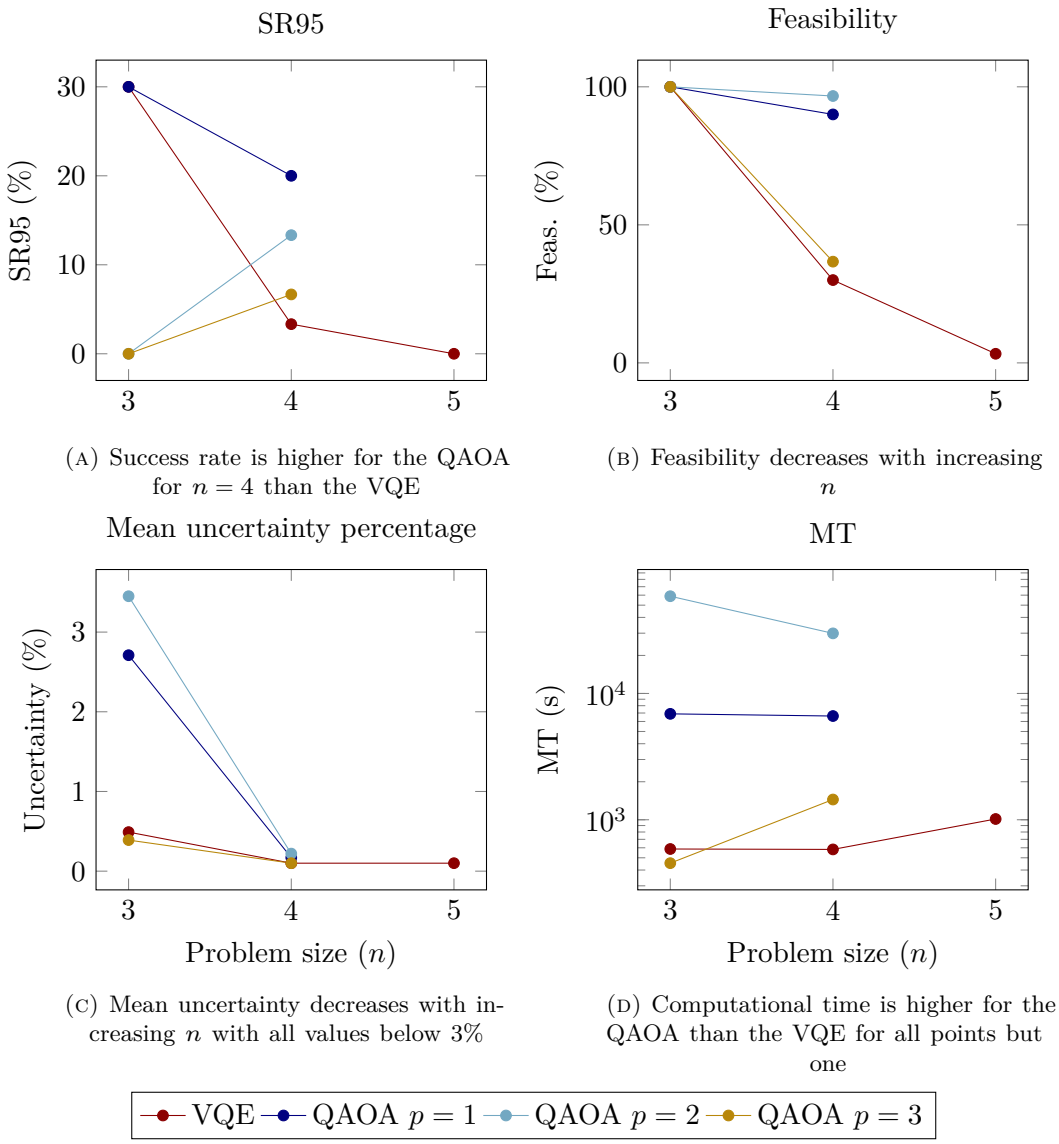
algorithm than the VQE to overcome the challenge of the QAP complexity limiting results as the QAP instance scales. When NISQ devices have a capacity for larger circuits, the hypothesis may show true, but it will be unclear until circuit capacity increases.

For Figure 3.10b, as the problem size increases from size 3 to 4, feasibility decreases. This decrease is evident for all the quantum devices, for all p-values and continues with larger problem sizes for the VQE. However, the QAOA has close to 100% feasibility for many devices for instances of size 4. The results for the QAOA where $p = 3$ had lower feasibility for instances of size 4 than $p = 1$ and $p = 2$. From Figure 3.10b, the QAOA performs better than the VQE for instances of size 4 in terms of feasibility for ibmq_toronto. Tables A.3, A.4 and A.7 show this phenomenon follows for the other quantum devices and may show that the VQE is preferable over the QAOA for maximising feasibility. Note that the QAOA - executed on ideal quantum devices - would return only feasible solutions under the variational principle (note Chapter 1.1.4). Therefore the results of $90 - 100\%$ feasibility show the power of NISQ devices in solving small instances of the QAP using the QAOA ($n = 3$ and $n = 4$).

For Figure 3.10c, uncertainty percentage decreases with increasing problem size from size 3 to 4 for all p-values of the QAOA for ibmq_toronto. From Tables A.5, A.6 and A.8, there is no pattern between p-values for ibmq_toronto or any other quantum device. The mean uncertainty percentage results across all the QAOA and the VQE results are low. Notably, the QAOA has larger uncertainty percentage values than the VQE for most quantum devices over both sizes. If the QAOA has better uncertainty percentages for the QAP than the VQE, the QAOA has a higher peak in its distribution of eigenstates (see Figure 2.1). Uncertainty percentage may be related to feasibility, and the improvement in the QAOA may be a result of this, but this hypothesis would be difficult to prove.

Figure 3.10d shows the values from Figures 3.9 and 3.3 in one plot for ibmq_toronto. Figure 3.10d shows the drawback of working with the QAOA - the computation times are an order of magnitude or higher than the VQE. Given the potential benefits to the other three metrics, they come at the cost of large average computational times. The MT for ibmq_toronto $p = 3$ is an outlier in Figure 3.10d and an outlier in Table A.7. Improvements in NISQ devices might minimise these trade-offs further, but (at the time of writing) these large CPU times make the QAOA the least time-efficient method studied in this research.

For Figure 3.11a, there is no clear relationship between success rate, p-value and problem size for the QAOA results for ibmq_qasm_simulator. Table A.7 shows no p-value presents the best success rate for all three simulators. Similarly to quantum devices, simulators have no clear relationship between circuit depth and success rate. For the QAP instance of size 4, all the QAOA results performed worse than the VQE. This trend, shown in ibmq_qasm_simulator in Figure 3.11a, continues with other two simulators shown in Tables A.3, A.4 and A.7. The QAOA may be more difficult to



(A) Success rate is higher for the QAOA for $n = 4$ than the VQE

(B) Feasibility decreases with increasing n

(C) Mean uncertainty decreases with increasing n with all values below 3%

(D) Computational time is higher for the QAOA than the VQE for all points but one

FIGURE 3.10: These plots are comparison of the VQE and the QAOA on SR95, feasibility, uncertainty percentage solving the QAP on ibmq_toronto (with conditional reset) as an example. The QAOA has a much larger computational time than the VQE (especially $p = 1$ and $p = 2$). The QAOA has a higher feasibility and SR95 for $n = 4$.

simulate because a Hamiltonian calculates the ansatz used to find the eigenstates. However, with only two data points, this is not conclusive. Therefore the VQE may be a more suitable algorithm for simulators in terms of success rate.

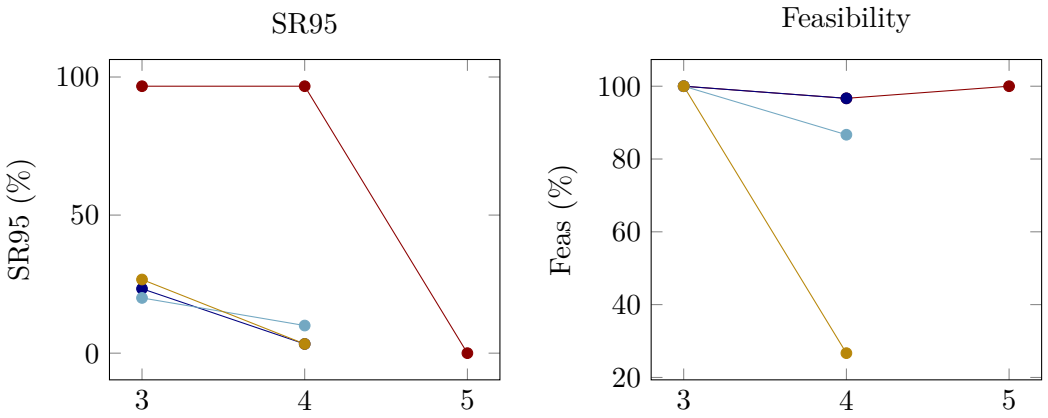
For Figure 3.11b, as the problem size increases from size 3 to 4, feasibility decreases of all three p-values for ibmq_qasm_simulator. For all the simulators, feasibility decreases or stays the same from problem size 3 to 4 for all p-values. Tables A.3, A.4 and A.7 show feasibility is $90-100\%$ for all simulators for both methods (ibmq_qasm_simulator $p=3$ size 4 is an outlier). These high feasibility percentages are similar to the results for the QAOA for quantum devices on size 3 and 4 devices. Therefore the QAOA and the VQE have the results of $90-100\%$ feasibility showing the power of simulators in solving small instances of the QAP.

For Figure 3.11c, uncertainty percentage decreases with increasing problem size from size 3 to 4 for all p-values of the QAOA for ibmq_qasm_simulator. From Tables A.5, A.6 and A.8, there is no pattern between p-values for any of the simulators. The uncertainty percentage results for the simulators across all the QAOA are low, while the VQE results are $90-100\%$. This gap is similar to the success rate where the simulators perform better for the VQE than the QAOA. Notably, the QAOA has larger uncertainty percentages for the simulator than the quantum devices in many cases, but they are all within the same range of $0-5\%$. The QAOA simulators and devices have similar results, while the VQE results diverge between simulators and devices.

Figure 3.11d shows the same values from Figures 3.9 and 3.3 in one plot for ibmq_qasm_simulator. Figure 3.11d reinforces the analysis made earlier - that the QAOA is less time-efficient than the VQE. Tables A.5, A.6 and A.8 reinforce the idea that the QAOA results are similar between the devices and simulators. The MT values for ibmq_qasm_simulator are smaller overall than the other two simulators for the QAOA, making this difference even more pronounced in those devices than in Figure 3.11d. Note a simulator gives a time-efficient way to get results for the VQE - with simulators getting high-quality results in the other three metrics. However, the QAOA has the same CPU time with lower quality success rates and uncertainty percentages when executed on the simulators. Therefore, a warm-start using the simulators is appropriate for the VQE but not necessarily for the QAOA.

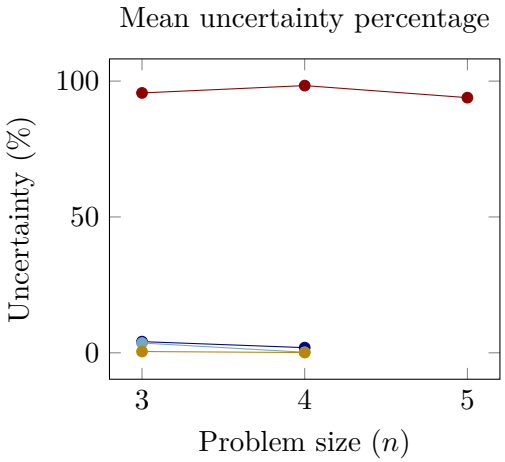
3.2.3 Quantum Hardware Limitations

Table 2.1 describes each IBM Quantum system, including the date of release. From Table 2.1, it is evident that developments in quantum computing are very rapid, with IBM Quantum releasing a new quantum processor every year with more qubits and a larger quantum volume. Similarly, the software continues to be updated and restructured. The VQE and the QAOA execute on Qiskit version 0.26.3, but the research used deprecated libraries and software packages to keep consistency with older results. As many new developments continue in quantum computing, it is also notable that the time of writing is the era of the NISQ devices. As a result, it is

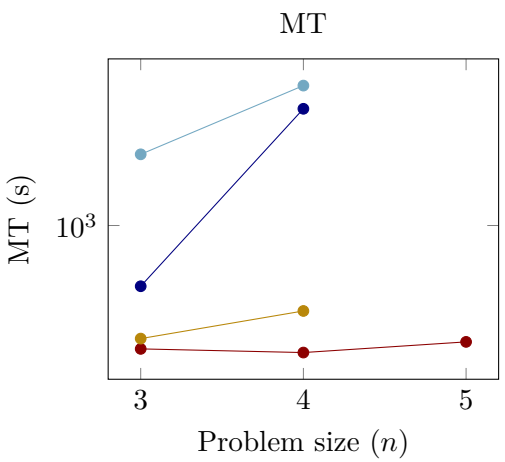


(A) Success rate is higher for the VQE than the QAOA

(B) Feasibility is higher for the VQE than the QAOA



(C) Mean uncertainty is higher for the VQE (near 100%) than the QAOA (near 0%)



(D) Computational time is higher for the QAOA than the VQE for all points

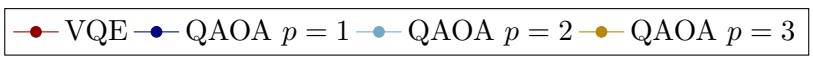


FIGURE 3.11: These plots are a comparison of the VQE and the QAOA on SR95, feasibility, uncertainty percentage and computational time results solving the QAP on ibmq_qasm_simulator (with conditional reset) as an example. The VQE performs best on all four metrics when this simulator executes the QAP. There is no clear relationship between p-values for the QAOA results.

currently impossible to solve extensive circuits or problems requiring hundreds of qubits. This section describes how certain limitations identified affected our results and suggested solutions.

One of the fundamental changes to the software is the conditional reset. A limitation of NISQ devices - circuit size - is also discussed in this section. And lastly, the limit of the simulators may suggest the value of a quantum warm-start whose results this research investigates on the same metrics as the simulator warm-start.

Conditional reset

IBM Quantum has introduced the conditional reset feature to make quantum device calculations more efficient. These gates are described in Table 2.3 and Chapter 2.2.2. All the QAOA results and the VQE results in Tables A.4 and A.6 use the gates (CX, ID, RZ, SX, X) available with the conditional reset. The VQE results in Tables A.3 and A.5 use (CX, ID, U1, U2, U3). Figure 3.12 compares conditional reset's impact on time and average times AT and MT on both devices. Figure 3.12 shows no clear improvement to computational time between results with or without conditional reset. When comparing the success rates in Table A.4 to those on the same devices without conditional reset in Table A.3, there is no improvement in success rate or feasibility. Therefore, the implementation of conditional reset did not impact the results obtained.

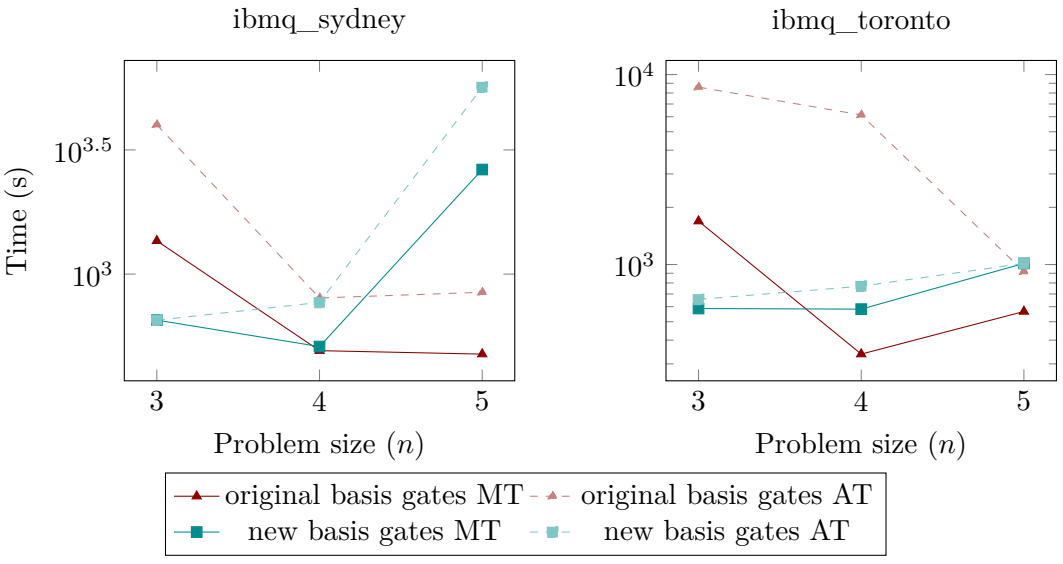


FIGURE 3.12: These plots aim to illustrate the impact on CPU time the new set of basis gates (CX, ID, RZ, SX, X) have on ibmq_sydney and ibmq_toronto for both methods solved with the VQE.

Circuit size as problem size increases

Figure 3.13 shows how the number of operations and depth of circuits depends on problem size. The VQE appears to have a direct exponential relationship between

problem size and circuit size. The QAOA is larger than the VQE for all p-values, with higher p-values having increasing circuit size. Regarding the number of circuit operations, the QAOA values increase more than the VQE in Figure 3.13. For the circuit depth, all $p = 2$ and $p = 3$ QAOA circuits are deeper than size 8 for the VQE in Figure 3.13. This phenomenon limits experiments to small QAP instances (hence the QAOA was executed until size instance 4). Therefore, obtaining an initial point from a simulator for instances of size greater than size 4 is intractable.

The rapid increase in circuit size with problem size is one of many factors that likely inform the increase in computation time with problem size. Since the QAOA generates larger circuits than the VQE, it may inform why the QAOA has a longer computation time. The relationship shown in Figure 3.13 suggests that exponential growth in circuit capacity on devices is needed to allow the QAOA to become more time-efficient like the VQE. Table A.11 shows the values for Figure 3.13 along with qubits and parameters. Besides improving devices, optimising circuits is an alternative way to improve results. Circuit size is independent of SPSA iterations and the type of variational form chosen for the VQE. However, the larger the number of SPSA iterations, the larger the computational time.

At the time of writing, the NISQ devices are limited, by their specifications, from solving the QAP instances considered non-trivial (problem sizes larger than 12). Table 2.1 shows the limited access to qubits that prevents instances of size 8 and larger from being tested. The other specifications in Table 2.1 impact error in the results and the computational time. Quantum volume may show benefits in some results, but this relationship is difficult to measure since quantum volume changes with other parameters. Table A.11 shows how the value of the parameters grow with problem size.

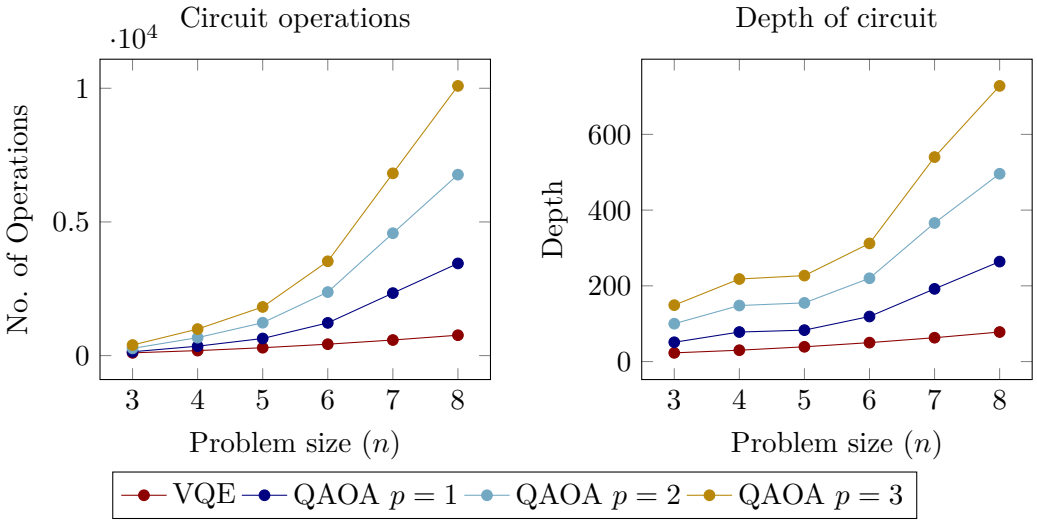


FIGURE 3.13: These plots shows the circuit size for quantum heuristics solving the QAP. Evidently, operations and depth grow exponentially with problem size (n). The VQE has smaller circuits than the QAOA and increasing p-values correlate to larger circuit size for the QAOA solutions to the QAP.

Quantum warm-start

The VQE and the QAOA use a simulator warm-start where an initial point is generated from a simulator locally on the classical computer. Simulating quantum computing on a classical computer can be difficult as the size of the QAP instance scales and the circuits grow. Hence, an attempt to overcome this limitation was to use a quantum computer to generate the initial point.

Quantum warm-start (QWS in Figure 3.14) is a method that executes the quantum algorithm once on a quantum device with a random initial point to generate an optimal point. That optimal point is the initial point for 30 trials on each quantum device and simulator (method detailed in Chapter 2.1.5).

Using a quantum computer to generate an initial point has the limitation that computation time increases, and the user may wait in queues to access the quantum device. The device ibmq_toronto found the initial point because it has reservation privileges, preventing wait times for access for a limited time each month. The limitation of using ibmq_toronto is limited to $n < 6$ because of its number of qubits.

The benefit of a quantum warm-start approach is that the circuit size limitations have less impact than the simulator warm-start approach. Therefore the quantum warm-start can execute instances with size 5 (where simulator warm-start cannot). Notably, the computational time is considerably more significant than the CPU time for size 4.

Figure 3.14 and Tables A.9 and A.10 display a brief investigation of a quantum warm-start. Firstly, the large computation times limit the number of data points collected. Therefore the number of experiments is only 10 trials instead of 30 trials for each device (quantum devices and simulators). The limit makes it difficult to make conclusions on this approach. Figure 3.14 shows results for simulator_statevector for both simulator warm-start and the quantum warm-start for comparison of this simulator as an example.

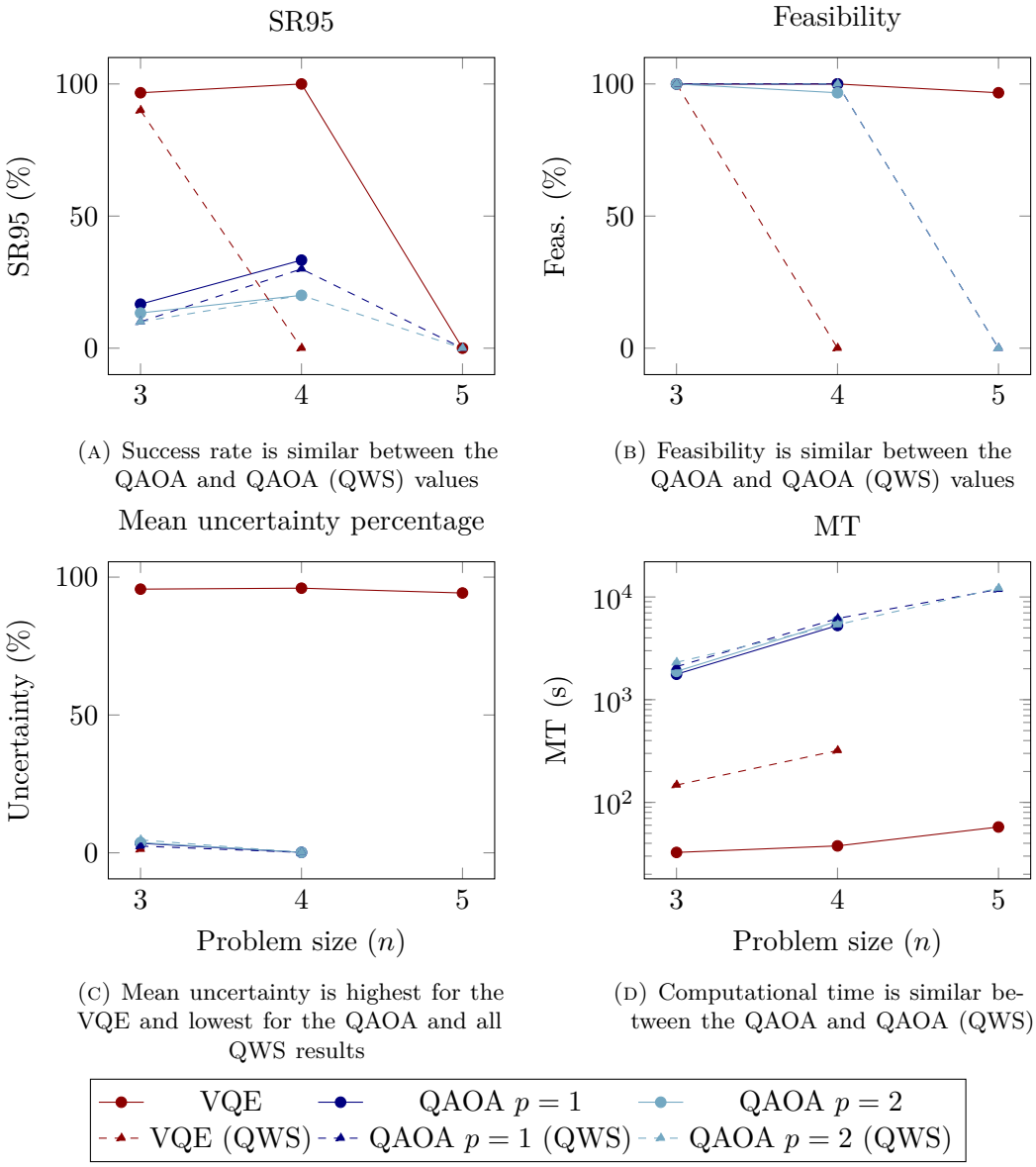
Comparing quantum and simulator warm-start for the VQE in Figure 3.14a shows that the success rates for instances of size 3 are similar but diverge at instance 4. This phenomenon is valid for the other two simulators (no devices have results for size 4 of the VQE because of time constraints mentioned at the start of this section). Feasibility in Figure 3.14b is similar, but the phenomenon is not present for simulator_mps from Table A.10. Figures 3.14c and 3.14d show the uncertainty percentage is much lower for both sizes ($n = 3$ and $n = 4$), and the time is more considerable for both sizes for the VQE. This decreased performance of the VQE in uncertainty percentage and CPU time continues with the other two simulators. The limited number of quantum devices that implemented the VQE quantum warm-start achieved similar results on the tested metrics as the simulator warm-start. Therefore, a quantum warm-start has little impact on the VQE when executed on quantum devices. However, a quantum warm-start performs worse than a simulator warm-start on the VQE when executed on a simulator.

Comparing quantum and simulator warm-start for the QAOA in Figure 3.14 shows that most of the results are very similar. The feasibility is the same between methods for both p-values. The success rate, uncertainty and computational time are similar and show no pattern. This pattern for simulator_statevector is true for all quantum devices and simulators in Tables A.9 and A.10. Since there is no change in key metrics, a quantum warm-start could be suitable for the QAOA since the results are the same and can solve QAP instances larger than size 4. However, since the time taken to find the initial point and queue time is unrecorded, these limitations still exist as disadvantages for a quantum warm-start.

A benefit of the quantum warm-start is that it helps find initial points for instances larger than size 4 for the QAOA. It is difficult to predict from the limited data on the size 5 results; however, solution quality generally decreases with increased problem size for the QAOA. No results are feasible, and therefore there is no uncertainty percentage, and the success rate is 0% on the simulators. The feasibility is higher than zero on most quantum devices and shows that results for size 5 are worse than results for size 4 for the quantum warm-start of the QAOA. The computational times are also more significant. Therefore, solution quality may still be an issue when quantum capacity allows for QAP instances larger than size 4 to be solved by the QAOA. Similarly, the quality of the initial point as the size of the QAP scales may be a concern. Computation time remains a barrier to solving the QAP with the QAOA, and the quantum warm-start does not overcome these limitations.

3.3 Conclusion

The results show the impact of these classical and quantum methods on the QAP. When solution quality is more important than computational time, deterministic methods like BNB are the fastest. BNB is only appropriate for QAP instances that are smaller than size 11. With limited time, SA is a better option than BNB. SA (used with the best parameters) provides a significant speed-up with a reduced loss of solution quality. The Quantum Computing approach in the preliminary results is slower than all methods tested, and the success rate is lower than that of classical heuristics. Comparing the success rate and feasibility between devices shows that no NISQ device has overall superior performance. The QAOA performs better than the VQE in feasibility, success rate and uncertainty percentage on the quantum devices, but with two data points, this is not conclusive. The VQE performs better than the QAOA on the simulators and is more time-efficient on quantum devices and simulators. Using a simulator warm-start method, the VQE has small circuits to solve for QAP instances of larger problem sizes. The conditional reset and quantum warm-start do not show substantial differences in results. Still, more research might be needed to overcome the limitations of increasing circuit size with problem size.



(A) Success rate is similar between the QAOA and QAOA (QWS) values

(B) Feasibility is similar between the QAOA and QAOA (QWS) values

(C) Mean uncertainty is highest for the VQE and lowest for the QAOA and all QWS results

(D) Computational time is similar between the QAOA and QAOA (QWS)

FIGURE 3.14: These plots are a comparison of the VQE and the QAOA on SR95, feasibility, uncertainty percentage and computational time results for the QAP on simulator_statevector (with conditional reset) as an example. The quantum warm-start (QWS) has little difference to the QAOA results but the VQE results are poorer on all 4 metrics. The $n = 5$ results show the highest computational time and 0 % feasibility and success rate.

Chapter 4

Conclusion

The results show two classical methods and two hybrid quantum algorithms - a Deterministic Branch and Bound (BNB), Simulated Annealing (SA), the Variational Quantum Eigensolver (VQE) and the Quantum Approximation Optimisation Algorithm (QAOA) - solving the Quadratic Assignment Assignment Problem (QAP). Each problem instance studied performed best with specific methods. When solution quality (success rate) is more critical than time, BNB is the fastest. BNB is only appropriate for QAP instances that are smaller than size 20. With limited time, SA is more efficient in computational time with a slight loss of solution quality with optimal parameters. The Quantum Computing approach is slower than both classical methods on the limited cases solvable by Noisy Intermediate-Scale Quantum (NISQ) devices ($n = 3, 4, 5, 6, 7$).

In comparison, the QAOA takes a longer computational time on the quantum devices than the VQE due to increased operations and depth. The success rate, feasibility and uncertainty percentage of the QAOA are comparable to that of the VQE for size 3, but the QAOA outperforms the VQE on instances of size 4. Working with NISQ devices has limitations like solving large circuits. Implementing new basis gates with conditional reset showed no change in results. A quantum warm-start, implemented to overcome the limitations of simulators, showed no difference in results and that instances of size 5 of the QAOA perform worse than size 4. It is difficult to make inferences on the QAOA because of the large size of the circuits and the limitations of the simulators and quantum systems. Therefore it is unanswered whether quantum methods can overcome the limits of the QAP's classification as an NP-Hard problem.

4.1 Research Objectives

This research describes quantum computing and its application to solving the QAP using the IBM Quantum systems and simulators. The QAP instances are encoded using Discrete Optimisation CPLEX (DOcplex). DOcplex represents the binary integer programming model of the QAP, which makes a Hamiltonian using tools in the Qiskit library (note Chapter 2.2 for the full method). The quantum heuristics solve for the eigensolution of the Hamiltonian, which is the decision vector of the QAP.

The evolution of the quantum processors shows no apparent difference in performance on the chosen metrics in studying the VQE. Similarly, conditional reset shows no evident improvement in computational time from the VQE.

The experimental results show that classical optimisation algorithms outperform quantum optimisation algorithms in success rate, feasibility and computational time in finding solutions to the QAP. The VQE exceeds the QAOA on quantum systems and simulators in computational time. In contrast, the feasibility and uncertainty percentage of the QAOA exceeds the VQE on quantum devices. The VQE outperforms the QAOA in all the metrics on the simulators. The success rate on the devices has no clear relationship between the VQE and the QAOA.

In answering the research question, when solving the maximum problem size possible on NISQ devices, classical algorithms like BNB and SA may currently outperform quantum algorithms like the VQE and QAOA regarding both solution quality and computational time. However, it's important to note that quantum technologies are still in the early stages of development, and future advancements in hardware and algorithms may lead to improved performance for quantum approaches. While quantum algorithms hold promise for optimization problems like the QAP, their practical applicability and performance on NISQ devices require more exploration, and classical algorithms remain competitive for large-scale instances.

4.2 Recommendations

In terms of future work, IBM Quantum plan to release more quantum devices with more qubits and different specifications. IBM Quantum launched a new 128 qubit device in December 2021 and plans to release more in 2022. Access to more qubits and new devices provides an opportunity to solve more instances of the QAP using the VQE and the QAOA. This opportunity could provide conclusions on the performance of the quantum methods as the instances of the QAP increase with problem size. Access to quantum systems with many qubits would also make a quantum warm start feasible for instances where $n > 7$ is an alternative to using limited simulators. Access to more qubits can also enable the study of real-life applications, like the energy systems described in Chapter 1, which have a larger problem size.

Future work can investigate the warm-start approach further. Measuring the quality of the initial point could provide insight into the current results and improve the solution quality and computation time in future. Optimising the initial point with multiple passes in the quantum heuristic is an alternative to the warm-start. Optimising other parameters is another possible approach to improve results.

Qiskit Runtime is a framework designed to optimise computations with many operations where IBM Quantum reports a 120 times computational speed up [92]. Specific Qiskit Runtime programs are intended for the VQE and the QAOA, reducing computation times. Studying this program and measuring its effect on the QAP would compare the new and old approaches. Measuring the improvement in quantum

computing solution for Combinatorial Optimisation Problems (COP) using the QAP as an example would be a meaningful contribution.

Lastly, future work could include examining the formulation of the QAP. This research uses two formulations that are not convex, and convexity may affect the performance of the QAP on the chosen metrics. There also exist methods that reduce the number of qubits a Quantum Unconstrained Binary Optimisation (QUBO) problem uses. Therefore adjusting the formulation of the QAP could improve performance on NISQ devices or allow for larger instances ($n > 7$) to be solved. Different instances of the QAP have different typologies, and understanding their impact on the results could also extend this research. For example, this research only uses symmetrical input matrices, and perhaps asymmetrical instances of the QAP will garner different results.

The study of solving COP with quantum algorithms on NISQ devices is rapidly changing and evolving, and the developments in hardware and software will determine future work.

Appendix A

Data tables

A.1 Description of selected QAPLIB datasets

Table A.1: Table of Data

Name	n	Best known solution	IG (%)
tai12a	12	224416 (8,1,6,2,11,10,3,5,9,7,12,4)	
tai40a	40	3139370 *	9.43
tai60a	60	7205962 *	12.21
tai80a	80	13499184 *	13.65
tai256c	256	44759294 *	1.48
nug14	14	1014 (9,8,13,2,1,11,7,14,3,4,12,5,6,10)	
nug15	15	1150 (1,2,13,8,9,4,3,14,7,11,10,15,6,5,12)	
nug16a	16	1610 (9,14,2,15,16,3,10,12,8,11,6,5,7,1,4,13)	
nug17	17	1732 (16,15,2,14,9,11,8,12,10,3,4,1,7,6,13,17,5)	
nug20	20	2570(18,14,10,3,9,4,2,12,11,16,19,15,20,8,13,17,5,7,1,6)	
nug21	21	2438 (4,21,3,9,13,2,5,14,18,11,16,10,6,15,20,19,8,7,1,12,17)	
bur26a	26	5426670 (26, 15, 11, 7, 4, 12, 13, 2, 6, 18, 1, 5, 9, 21, 8, 14, 3, 20, 19, 25, 17, 10, 16, 24, 23, 22)	
chr15c	15	9504 (13,2,5,7,8,1,14,6,4,3,15,9,12,11,10)	
chr18a	18	11098 (3,13,6,4,18,12,10,5,1,11,8,7,17,14,9,16,15,2)	
chr20a	20	2192 (3,20,7,18,9,12,19,4,10,11,1,6,15,8,2,5,14,16,13,17)	
chr22a	22	6156 (15,2,21,8,16,1,7,18,14,13,5,17,6,11,3,4,20,19,9,22,10,12)	

A.2 Results

Key points from the results below can be found in their respective subsections in the form of plots. The methods used are Branch and Bound (BNB), Simulated Annealing (SA), Variational Quantum Eigensolver (VQE) and Quantum Approximation Optimisation Algorithm (QAOA).

*The optimal permutation was omitted for ease of reading

TABLE A.2: SA: parameters and success rates over problem size (n)

			Parameters			
n	**SR95**	**SR99**	ϵ^*	$L^\dagger$	$\alpha^\ddagger$	$T^\S$
3	100.0	100.0	1.0	20	0.90	20
4	100.0	100.0	1.0	20	0.90	20
5	100.0	100.0	1.0	20	0.90	20
6	96.67	60.0	1.0	20	0.90	20
7	93.33	93.33	1.0	20	0.90	740
8	80.0	26.67	1.0	20	0.90	1460
9	100.0	53.33	1.0	20	0.92	2540
10	60.0	3.33	1.0	20	0.90	420
11	93.33	40.0	1.0	28	0.97	11 820
12	100.0	63.33	0.1	22	0.99	14 660
14	100.0	30.0	1.0	20	0.92	19 060
15	100.0	80.0	1.0	20	0.99	19 700
16	100.0	56.67	1.0	20	0.92	820
17	100.0	43.33	1.0	20	0.97	220
20	100.0	10.0	1.0	20	0.90	920
21	100.0	43.33	1.0	22	0.97	23 420
26	100.0	100.0	0.1	24	0.97	420
40	100.0	0.0	0.01	26	0.99	6020
60	100.0	0.0	0.0001	26	0.99	4020
80	100.0	0.0	0.001	26	0.95	4020

Find the parameters in Algorithm 4 and their explanation in Chapter 2.1.2
* ϵ: final temperature
† L: Markov chain length
‡ α: cool-down factor
§ T: starting temperature

TABLE A.3: The VQE: parameters, solution quality and computational time results over problem size (n) (before conditional reset)

n	**Parameters**[*]	**SR99**	**SR95**	**Feas.**[†]	**AT(s)**[‡]	**MT(s)**[§]	**Devices**
3	[1000, "TL"]	100.0	100.0	100.0	45.65	45.65	ibmq_qasm_simulator
		86.67	86.67	100.0	1486.68	151.78	ibmq_johannesburg
		43.33	43.33	100.0	479.18	415.05	ibmq_montreal
		96.67	96.67	100.0	414.38	520.24	ibmq_cambridge
		53.33	53.33	100.0	987.22	548.20	ibmq_rochester
		0.0	0.0	100.0	852.69	268.71	ibmq_boeblingen
		93.33	93.33	100.0	3994.88	1360.91	ibmq_sydney
		73.33	73.33	100.0	8586.37	1692.50	ibmq_toronto
		96.67	96.67	100.0	3722.58	557.68	ibmq_manhattan
4	[3000, "TL"]	0.0	100.0	100.0	45.64	45.64	ibmq_qasm_simulator
		3.33	16.67	56.67	495.1	342.20	ibmq_johannesburg
		3.33	6.67	56.67	827.51	369.81	ibmq_montreal
		6.67	13.33	43.33	562.17	367.47	ibmq_cambridge
		6.66	13.33	43.33	4803.45	3260.97	ibmq_rochester
		0.0	10.0	46.67	1397.01	873.04	ibmq_boeblingen
		0.0	3.33	36.67	801.31	491.77	ibmq_sydney
		0.0	6.67	40.0	6133.34	337.68	ibmq_toronto
		0.0	10.0	40.0	298.94	267.96	ibmq_manhattan
5	[5000, "TL"]	0.0	0.0	96.67	64.05	64.05	ibmq_qasm_simulator
		0.0	0.0	0.0	9986.32	872.01	ibmq_montreal
		0.0	0.0	0.0	565.01	565.01	ibmq_cambridge
		0.0	0.0	3.33	2116.71	1801.51	ibmq_rochester
		0.0	0.0	3.33	845.03	476.22	ibmq_sydney
		3.33	3.33	6.67	917.79	566.01	ibmq_toronto
		0.0	0.0	3.33	3020.28	1318.44	ibmq_manhattan
6	[6000, "TL"]	0.0	0.0	0.0	5017.53	2265.98	ibmq_rochester
		0.0	0.0	0.0	24 373.16	2967.89	ibmq_manhattan
7	[12000, "TL"]	0.0	0.0	0.0	18 429.26	7928.29	ibmq_manhattan

Find the parameters in Algorithm 5 and the metrics and explanation in Chapter 2.2.5

[*] Parameters: [number of SPSA iterations, variational form]

[†] Feas.: percentage of trials found feasible

[‡] AT: average CPU time with outliers

[§] MT: average CPU time without outliers

TABLE A.4: The VQE: parameters, solution quality and computational time results over problem size (n) (with conditional reset)

n	**Parameters**[*]	**SR99**	**SR95**	**Feas.**[†]	**AT(s)**[‡]	**MT(s)**[§]	**Devices**
3	[1000, "TL"]	100.0	100.0	100.0	33.41	33.41	simulator_mps
		96.67	96.67	100.0	33.17	32.59	simulator_statevector
		96.67	96.67	100.0	193.49	193.49	ibmq_qasm_simulator
		23.33	23.33	100.0	155.46	150.18	ibmq_guadalupe
		86.67	86.67	100.0	652.57	652.57	ibmq_sydney
		30.0	30.0	100.0	655.30	586.91	ibmq_toronto
		83.33	83.33	100.0	6455.18	266.34	ibmq_mumbai
		10.0	10.0	100.0	415.08	248.95	ibm_cairo
		46.67	46.67	100.0	1441.64	1235.92	ibm_hanoi
		100.0	100.0	100.0	604.24	604.24	ibmq_manhattan
4	[3000, "TL"]	0.0	96.67	96.67	36.09	35.62	simulator_mps
		0.0	100.0	100.0	37.73	37.73	simulator_statevector
		0.0	96.67	96.67	184.27	184.27	ibmq_qasm_simulator
		0.0	3.33	33.33	494.41	494.41	ibmq_guadalupe
		3.33	6.67	36.67	768.95	510.72	ibmq_sydney
		0.0	3.33	30.0	720.66	582.44	ibmq_toronto
		3.33	3.33	56.67	2393.14	266.71	ibmq_mumbai
		3.33	10.0	30.0	1160.29	1160.29	ibm_cairo
		3.33	3.33	36.67	210.36	210.36	ibm_hanoi
		0.0	10.0	36.67	553.22	497.06	ibmq_manhattan
5	[5000, "TL"]	0.0	0.0	96.67	321.44	316.78	simulator_mps
		0.0	0.0	96.67	58.24	57.61	simulator_statevector
		0.0	0.0	100.0	212.39	212.39	ibmq_qasm_simulator
		0.0	0.0	0.0	5659.09	2636.32	ibmq_sydney
		0.0	0.0	3.33	1014.09	1014.09	ibmq_toronto
		0.0	0.0	0.0	626.4	326.84	ibmq_mumbai
		0.0	0.0	0.0	1113.64	1113.64	ibm_cairo
		0.0	0.0	3.33	280.9	276.63	ibm_hanoi
6	[6000, "TL"]	0.0	0.0	100.0	835.59	835.59	simulator_mps
7	[12000, "TL"]	0.0	0.0	96.67	1625.68	1603.99	simulator_mps

Find the parameters in Algorithm 5 and the metrics and explanation in Chapter 2.2.5

[*] Parameters: [number of SPSA iterations, variational form]

[†] Feas.: percentage of trials found feasible

[‡] AT: average CPU time with outliers

[§] MT: average CPU time without outliers

TABLE A.5: The VQE: uncertainty percentage results over problem size (n) (before conditional reset)

n	# feas.*	mean (%)	max (%)	min (%)	std (%)	Devices
3	30	95.65	98.93	0.68	17.64	ibmq_qasm_simulator
	30	0.57	1.07	0.20	0.22	ibmq_johannesburg
	30	0.41	0.78	0.20	0.13	ibmq_montreal
	30	2.79	4.69	0.59	1.22	ibmq_cambridge
	30	0.55	1.07	0.20	0.21	ibmq_rochester
	30	0.64	0.98	0.39	0.18	ibmq_boeblingen
	30	1.04	2.54	0.20	0.48	ibmq_sydney
	30	0.59	0.98	0.20	0.21	ibmq_toronto
	30	28.62	35.25	0.29	5.96	ibmq_manhattan
4	30	96.68	98.34	48.63	8.92	ibmq_qasm_simulator
	17	0.10	0.20	0.10	0.02	ibmq_johannesburg
	17	0.10	0.10	0.10	0.00	ibmq_montreal
	13	0.10	0.10	0.10	0.00	ibmq_cambridge
	19	0.10	0.20	0.10	0.02	ibmq_rochester
	14	0.10	0.10	0.10	0.00	ibmq_boeblingen
	11	0.10	0.10	0.10	0.00	ibmq_sydney
	12	0.10	0.10	0.10	0.00	ibmq_toronto
	12	0.10	0.10	0.10	0.00	ibmq_manhattan
5	29	94.24	94.24	94.24	0.00	ibmq_qasm_simulator
	0	-	-	-	-	ibmq_montreal
	0	-	-	-	-	ibmq_cambridge
	0	-	-	-	-	ibmq_rochester
	0	-	-	-	-	ibmq_sydney
	2	0.10	0.10	0.10	0.0	ibmq_toronto
	1	0.10	0.10	0.10	0.0	ibmq_manhattan
6	1	0.10	0.10	0.10	0.00	ibmq_rochester
	0	-	-	-	-	ibmq_manhattan
7	0	-	-	-	-	ibmq_manhattan

If 0 trials are feasible then there is no metric to measure - find more on this metric in Chapter 2.2.5

* # feas. number of the 30 trials that are feasible

TABLE A.6: The VQE: uncertainty percentage results over problem size (n) (with conditional reset)

n	# feas.*	mean (%)	max (%)	min (%)	std (%)	Devices
3	30	95.80	98.93	5.27	16.81	simulator_mps
	30	95.64	98.93	0.39	17.69	simulator_statevector
	30	95.64	98.93	0.29	17.71	ibmq_qasm_simulator
	30	0.44	0.88	0.20	0.14	ibmq_guadalupe
	30	0.77	2.05	0.20	0.43	ibmq_sydney
	30	0.49	0.88	0.20	0.16	ibmq_toronto
	30	0.51	0.98	0.2	0.19	ibmq_mumbai
	30	0.45	0.68	0.29	0.13	ibm_cairo
	30	0.42	0.78	0.20	0.14	ibm_hanoi
4	29	98.34	98.34	98.34	0.00	simulator_mps
	30	95.99	98.34	27.93	12.64	simulator_statevector
	29	98.34	98.34	98.34	0.0	ibmq_qasm_simulator
	10	0.10	0.10	0.10	0.00	ibmq_guadalupe
	11	0.11	0.20	0.10	0.03	ibmq_sydney
	9	0.10	0.10	0.10	0.00	ibmq_toronto
	17	0.10	0.10	0.10	0.00	ibmq_mumbai
	9	0.11	0.20	0.10	0.03	ibm_cairo
	11	0.10	0.10	0.10	0.00	ibm_hanoi
5	29	94.24	94.34	94.24	0.02	simulator_mps
	29	94.24	94.24	94.24	0.00	simulator_statevector
	30	93.91	94.24	84.38	1.77	ibmq_qasm_simulator
	0	-	-	-	-	ibmq_sydney
	1	0.10	0.10	0.10	0.00	ibmq_toronto
	0	-	-	-	-	ibmq_mumbai
	0	-	-	-	-	ibm_cairo
	1	0.10	0.10	0.10	0.00	ibm_hanoi
6	30	84.41	84.47	82.62	0.33	simulator_mps
7	29	1.07	1.07	1.07	0.00	simulator_mps

If 0 trials are feasible then there is no metric to measure - find more on this metric in Chapter 2.2.5

* # feas. number of the 30 trials that are feasible

TABLE A.7: The QAOA: parameters, solution quality and computational time results over problem size (n) for each p-value (with conditional reset)

p	n	Par.*	SR99	SR95	Feas.†	AT(s)‡	MT(s)§	Devices
1	3	1	23.33	23.33	100.0	453.34	445.47	ibmq_qasm_simulator
			16.67	16.67	100.0	1768.80	1768.80	simulator_statevector
			10.0	10.0	100.0	1956.79	1956.79	simulator_mps
			16.67	16.67	100.0	5176.16	2383.20	ibmq_guadalupe
			30.0	30.0	100.0	12 697.07	6595.37	ibmq_toronto
			60.0	60.0	100.0	12 288.01	2342.49	ibmq_montreal
			6.67	6.67	100.0	13 778.79	2863.72	ibm_hanoi
			10.0	10.0	100.0	18 589.98	5904.43	ibm_cairo
			16.67	16.67	100.0	22 811.06	12 141.47	ibmq_manhattan
	4	1	0.0	3.33	96.67	4924.15	4729.84	ibmq_qasm_simulator
			16.67	33.33	100.0	5278.72	5278.72	simulator_statevector
			10.0	30.0	96.67	7862.57	7768.21	simulator_mps
			6.67	20.0	100.0	19 596.96	7983.68	ibmq_guadalupe
			10.0	20.0	90.0	19 877.39	6626.54	ibmq_toronto
			20.0	43.33	100.0	34 163.30	28 407.03	ibmq_montreal
			10.0	23.33	96.67	9743.14	8224.30	ibm_hanoi
			20.0	30.0	96.67	61 429.97	9342.21	ibm_cairo
2	3	1	20.0	20.0	100.0	2579.95	2579.95	ibmq_qasm_simulator
			36.67	36.67	100.0	1833.80	1833.80	simulator_mps
			13.33	13.33	100.0	1885.27	1885.27	simulator_statevector
			0.0	0.0	100.0	58 853.24	58 853.24	ibmq_toronto
			40.0	40.0	100.0	31 754.84	10 608.88	ibm_hanoi
			6.67	6.67	100.0	31 795.21	27 097.91	ibmq_guadalupe
	4	1	3.33	10.0	86.67	6658.05	6441.94	ibmq_qasm_simulator
			3.33	16.67	90.0	9512.24	9290.07	simulator_mps
			3.33	20.0	96.67	5961.19	5788.60	simulator_statevector
			10.0	20.0	100.0	28 250.70	28 250.70	ibm_hanoi
			3.33	16.67	100.0	32 319.71	27 322.74	ibmq_guadalupe
			3.33	13.33	96.67	41 250.59	29 899.28	ibmq_toronto
3	3	100	26.67	26.67	100.0	221.89	221.89	ibmq_qasm_simulator
			16.67	16.67	100.0	1990.63	1990.63	simulator_mps
			16.67	16.67	100.0	2430.14	2160.71	simulator_statevector
			20.0	20.0	100.0	86 327.60	46 532.08	ibmq_sydney
			0.0	0.0	100.0	552.26	453.37	ibmq_toronto
			3.33	3.33	100.0	14 588.49	10 835.97	ibmq_guadalupe
	4	50	3.33	3.33	26.67	320.58	320.58	ibmq_qasm_simulator
			0.0	13.33	96.67	11 700.57	11 483.13	simulator_mps
			13.33	26.67	93.33	6528.40	6221.85	simulator_statevector
			6.67	16.67	60.0	47 035.14	8325.71	ibmq_sydney
			0.0	6.67	36.67	6979.90	1446.55	ibmq_toronto
			3.33	30.0	96.67	12 145.58	8347.45	ibmq_guadalupe

Find the parameters in Algorithm 6 and the metrics and explanation in Chapter 2.2.5

* Par.: number of SPSA iterations (parameter)

† Feas.: percentage of trials found feasible

‡ AT: average CPU time with outliers

§ MT: average CPU time without outliers

TABLE A.8: The QAOA: uncertainty percentage results over problem size (n) for each p-value (with conditional reset)

p	n	# feas.†	mean (%)	max (%)	min (%)	std (%)	Devices
1	3	30	4.12	12.21	1.56	2.92	ibmq_qasm_simulator
		30	3.46	12.21	1.56	2.57	simulator_statevector
		30	3.87	7.32	1.46	1.87	simulator_mps
		30	2.71	4.69	1.86	0.58	ibmq_toronto
		30	2.62	3.32	2.15	0.32	ibmq_guadalupe
		30	2.36	3.42	1.76	0.40	ibmq_montreal
		30	3.65	6.05	3.12	0.65	ibm_hanoi
		30	2.51	3.81	1.76	0.47	ibm_cairo
		30	2.89	4.00	1.56	0.68	ibm_manhattan
	4	29	1.91	3.81	0.10	1.18	ibmq_qasm_simulator
		30	0.17	0.59	0.10	0.14	simulator_statevector
		29	0.12	0.20	0.10	0.04	simulator_mps
		30	0.21	0.39	0.10	0.07	ibmq_guadalupe
		27	0.17	0.29	0.10	0.06	ibmq_toronto
		30	0.18	0.29	0.10	0.07	ibmq_montreal
		29	0.17	0.29	0.10	0.07	ibm_hanoi
		29	0.17	0.39	0.10	0.07	ibm_cairo
2	3	30	3.61	11.52	1.27	2.35	ibmq_qasm_simulator
		30	3.41	6.15	0.29	1.49	simulator_mps
		30	3.71	9.18	0.29	2.25	simulator_statevector
		30	3.45	5.96	1.66	1.02	ibmq_toronto
		30	2.71	3.52	2.25	0.36	ibm_hanoi
		30	2.78	3.52	2.25	0.33	ibmq_guadalupe
	4	26	0.21	0.78	0.10	0.16	ibmq_qasm_simulator
		27	0.21	0.78	0.10	0.19	simulator_mps
		29	0.18	1.17	0.10	0.20	simulator_statevector
		29	0.22	0.49	0.10	0.10	ibmq_toronto
		30	0.15	0.29	0.10	0.07	ibm_hanoi
		30	0.24	0.39	0.10	0.08	ibmq_guadalupe
3	3	30	0.47	0.98	0.1	0.22	ibmq_qasm_simulator
		30	3.97	9.38	1.46	2.02	simulator_mps
		30	3.82	10.16	1.27	1.85	simulator_statevector
		30	0.41	0.78	0.20	0.15	ibmq_sydney
		30	0.39	0.68	0.20	0.13	ibmq_toronto
		30	2.2	3.52	1.56	0.46	ibmq_guadalupe
	4	8	0.10	0.10	0.10	0.0	ibmq_qasm_simulator
		29	0.12	0.29	0.1	0.06	simulator_mps
		28	0.12	0.2	0.10	0.04	simulator_statevector
		18	0.10	0.20	0.10	0.02	ibmq_sydney
		11	0.10	0.10	0.10	0.00	ibmq_toronto
		29	0.20	0.39	0.10	0.09	ibmq_guadalupe

† Find the explanation of this metric in Chapter 2.2.5

* # feas. number of the 30 trials that are feasible

TABLE A.9: Quantum warm-start: the VQE and the QAOA parameters, solution quality and computational time results over problem size (n) (with conditional reset)

n	p	Par.*	SR99	SR95	Feas.†	AT(s)‡	MT(s)§	Devices	Method
3	1	1	0.0	0.0	100.0	1686.9	1610.1	ibmq_qasm_simulator	QAOA
			10.0	10.0	100.0	2095.3	2095.3	simulator_statevector	
			40.0	40.0	100.0	5837.6	4509.0	ibm_cairo	
	2	1	20.0	20.0	100.0	2070.0	2070.0	simulator_mps	
			10.0	10.0	100.0	2311.5	2311.5	simulator_statevector	
			10.0	10.0	100.0	21 044.7	3373.5	ibm_cairo	
			10.0	10.0	100.0	17 869.7	17 869.7	ibm_hanoi	
	-	[1,"TL"]	100.0	100.0	100.0	179.8	153.9	ibmq_qasm_simulator	VQE
			90.0	90.0	100.0	147.6	147.6	simulator_statevector	
			90.0	90.0	100.0	243.8	243.8	simulator_mps	
			70.0	70.0	100.0	30 890.2	3889.9	ibmq_guadalupe	
			40.0	40.0	100.0	1267.7	734.8	ibmq_sydney	
4	1	1	10.0	20.0	100.0	5394.7	5394.7	ibmq_qasm_simulator	QAOA
			20.0	30.0	100.0	6180.4	6180.4	simulator_statevector	
			0.0	20.0	100.0	29 356.5	5989.0	ibm_cairo	
	2	1	10.0	30.0	100.0	8266.4	8266.4	simulator_mps	
			0.0	20.0	100.0	5461.0	5371.5	simulator_statevector	
			30.0	60.0	100.0	10 594.6	6857.3	ibm_cairo	
			10.0	30.0	100.0	18 568.7	18 568.7	ibm_hanoi	
	-	[1,"TL"]	0.0	0.0	0.0	238.6	238.6	ibmq_qasm_simulator	VQE
			0.0	0.0	0.0	320.0	320.0	simulator_statevector	
			0.0	0.0	90.0	296.2	273.7	simulator_mps	
5	1	5	0.0	0.0	0.0	13 411.6	13 411.6	ibmq_qasm_simulator	QAOA
			0.0	0.0	0.0	11 862.3	11 862.3	simulator_statevector	
			0.0	0.0	10.0	121 376.2	58 487.8	ibm_cairo	
			0.0	0.0	0.0	54 655.9	54 655.9	ibmq_mumbai	
	2	1	0.0	0.0	0.0	12 177.8	12 177.8	simulator_statevector	
			0.0	0.0	40.0	26 373.9	17 670.8	ibm_cairo	
			0.0	0.0	10.0	45 564.3	45 564.3	ibm_hanoi	

The quantum warm-start uses 10 trials and not 30 - find the explanation in Chapter 2.1.5
Find the parameters in Algorithms 5 and 6, the metrics in Chapter 2.2.5

* Par.: (parameter) number of SPSA iterations for the QAOA

* Par.: (parameters) [number of SPSA iterations, variational form] for the VQE

† Feas.: percentage of trials found feasible

‡ AT: average CPU time with outliers

§ MT: average CPU time without outliers

TABLE A.10: Quantum warm-start: the VQE and the QAOA uncertainty percentage results over problem size (n) (with conditional reset)

n	p	# feas.*	mean (%)	max (%)	min (%)	std (%)	Devices	Methods
3	1	10	7.85	13.87	3.22	3.72	ibmq_qasm_simulator	QAOA
		10	2.45	4.88	1.17	1.09	simulator_statevector	
		10	2.17	2.93	1.46	0.44	ibm_cairo	
	2	10	4.52	9.18	2.05	2.43	simulator_mps	
		10	4.69	9.18	2.05	2.46	simulator_statevector	
		10	2.23	2.44	1.95	0.17	ibm_cairo	
		10	2.10	2.34	1.76	0.19	ibm_hanoi	
	-	10	1.16	1.27	0.2	0.32	ibmq_qasm_simulator	VQE
		10	1.29	1.46	1.27	0.06	simulator_statevector	
		10	1.62	1.76	0.39	0.41	simulator_mps	
		10	0.42	0.78	0.29	0.14	ibmq_guadalupe	
		10	0.39	0.78	0.20	0.15	ibmq_sydney	
4	1	10	0.15	0.2	0.1	0.05	ibmq_qasm_simulator	QAOA
		10	0.14	0.20	0.10	0.05	simulator_statevector	
		10	0.14	0.20	0.10	0.05	ibm_cairo	
	2	10	0.12	0.20	0.10	0.04	simulator_mps	
		10	0.16	0.20	0.10	0.05	simulator_statevector	
		10	0.13	0.20	0.10	0.04	ibm_cairo	
		10	0.11	0.20	0.10	0.03	ibm_hanoi	
	-	0	-	-	-	-	ibmq_qasm_simulator	VQE
		0	-	-	-	-	simulator_statevector	
		9	0.10	0.10	0.10	0.00	simulator_mps	
5	1	0	-	-	-	-	ibmq_qasm_simulator	QAOA
		0	-	-	-	-	simulator_statevector	
		0	-	-	-	-	ibmq_mumbai	
		1	0.10	0.10	0.10	0.00	ibm_cairo	
	2	0	-	-	-	-	simulator_statevector	
		4	0.10	0.10	0.10	0.00	ibm_cairo	
		1	0.10	0.10	0.10	0.00	ibm_hanoi	

* If 0 trials are feasible then there is no metric to measure - find more on this metric in Chapter 2.2.5
# feas. number of the 10 trials that are feasible

TABLE A.11: Specifications of the circuits before transpiling on the quantum devices

n	**Qubits**	p	**Parameters**[*]	**Operations**	**Depth**	**Method**
3	9	-	[1000, "TL"]	103	23	VQE
		1	[1]	144	51	QAOA
		2	[1]	270	100	
		3	[100]	396	149	
4	16	-	[3000, "TL"]	187	30	VQE
		1	[1]	352	78	QAOA
		2	[1]	672	148	
		3	[50]	992	218	
5	25	-	[5000, "TL"]	295	39	VQE
		1	-	640	83	QAOA
		2	-	1230	155	
		3	-	1820	227	
6	36	-	[6000, "TL"]	427	50	VQE
		1	-	1224	119	QAOA
		2	-	2376	220	
		3	-	3528	312	
7	49	-	[12000, "TL"]	583	63	VQE
		1	-	2338	192	QAOA
		2	-	4578	366	
		3	-	6818	540	
8	64	-	-	763	78	VQE
		1	-	3448	264	QAOA
		2	-	6768	496	
		3	-	10088	728	

[†] Parameters: [number of SPSA iterations, variational form] for the VQE
Parameter: [number of SPSA iterations] for the QAOA

Appendix B

Published work

B.1 Conference publication

The conference paper featured this research on the Quadratic Assignment Problem with another Combinatorial Optimisation Problem (COP) - the Travelling Salesmen Problem (TSP) in a conference paper in the 2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI) [77]. This conference paper shows many of the Variational Quantum Eigensolver (VQE) results garnered before introducing the conditional reset used in this research. Additionally, it establishes Branch and Bound (BNB) and Simulated Annealing (SA) as classical benchmark methods for comparison to the VQE, which this research continues. The conference paper concludes that Noisy Intermediate-Scale Quantum (NISQ) devices perform better on all metrics than the quantum methods - this conclusion is shared with this research.

B.1.1 Abstract

The prevalence of NP-Hard COP requires a continuous search for powerful heuristic techniques to efficiently solve problems of this nature due to the exponentially growing CPU times faced by deterministic methods. This paper focuses on two COP, namely the TSP and the QAP. In particular, the computational performance of classical optimisation techniques, BNB and SA, are compared to the VQE algorithm. These algorithms are executed respectively on classical and Noisy Intermediate-Scale Quantum (NISQ) computers, in this case, a host of IBM quantum devices. The experimental results resemble and extend on previously reported results found in the literature. Furthermore, this research critically analyses, compares and comments on the performance of quantum computing techniques to that of classical techniques. The results highlight the current shortcomings of NISQ devices, as classical optimisation techniques significantly outperform the VQE algorithm when run on state of the art IBM Quantum devices.

B.2 Preprint journal paper

This preprint [82] is an extension of the conference publication that remains under review. It includes new metrics - namely feasibility and uncertainty percentage (both

examined in this research). The preprint paper has conclusions on the basis gates included in the results of this research. A few of the Quantum Approximate Optimisation Algorithm (QAOA) results in this research (where $p = 3$) are also in the preprint. The conclusion on the QAOA are more detailed and vary somewhat from the findings in the preprint.

B.2.1 Abstract

The exponential increase in CPU time taken to deterministically solve NP-Hard COP, as the problem size scales, has resulted in a search for non-deterministic optimisation solution techniques to obtain solutions to COP efficiently. This paper juxtaposes classical and quantum optimisation algorithms' performance to solve two common COP, the TSP and QAP. The two classical optimisation techniques applied are BNB and SA, and the two quantum optimisation methods used are the VQE algorithm and QAOA. These algorithms are respectively executed on classical and IBM's suite of NISQ computers. Our experimental results resemble and extend on previously reported results in the literature. Extensions include critically analysing, comparing and commenting on the performance of quantum optimisation computing techniques to classical techniques, with respect to computational time and additional metrics used to measure solution quality. Furthermore, a comparison of the impact of a new set of basis gates on the quantum optimisation techniques was investigated; the results did not reflect any consistent impact on results. The VQE algorithm and QAOA executed on state of the art IBM quantum devices are outperformed by classical optimisation techniques, highlighting the shortcomings of NISQ devices.

References

[1] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, p. 5, 2014.

[2] A. Ajagekar and F. You, "Quantum computing for energy systems optimization: Challenges and opportunities," *Energy*, vol. 179, pp. 76–89, 2019.

[3] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, no. 1, pp. 53–76, 1957, ISSN: 00129682, 14680262. [Online]. Available: http://www.jstor.org/stable/1907742.

[4] M. Abdel-Basset, G. Manogaran, H. Rashad, and A. N. H. Zaied, "A comprehensive review of quadratic assignment problem: Variants, hybrids and applications," *Journal of Ambient Intelligence and Humanized Computing*, 2018, ISSN: 1868-5145. DOI: 10.1007/s12652-018-0917-x. [Online]. Available: https://doi.org/10.1007/s12652-018-0917-x.

[5] C. Schellewald, S. Roth, and C. Schnörr, "Evaluation of a convex relaxation to a quadratic assignment matching approach for relational object views," *Image and Vision Computing*, vol. 25, no. 8, pp. 1301–1314, 2007, ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2006.08.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885606002678.

[6] O. Nissfolk, R. Pörn, T. Westerlund, and F. Jansson, "A mixed integer quadratic reformulation of the quadratic assignment problem with rank-1 matrix," in *11th International Symposium on Process Systems Engineering*, ser. Computer Aided Chemical Engineering, I. A. Karimi and R. Srinivasan, Eds., vol. 31, Elsevier, 2012, pp. 360–364. DOI: https://doi.org/10.1016/B978-0-444-59507-2.50064-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444595072500640.

[7] D. J. White, "A convex form of the quadratic assignment problem," *European Journal of Operational Research*, vol. 65, no. 3, pp. 407–416, 1993, ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(93)90120-C. [Online]. Available: https://www.sciencedirect.com/science/article/pii/037722179390120C.

[8] E. Lawler, "The quadratic assignment problem: A brief review," in *Combinatorial programming: Methods and applications*, Springer, 1995, pp. 351–360.

[9] S. Ishii and M.-a. Sato, "Constrained neural approaches to quadratic assignment problems," *Neural Networks*, vol. 11, no. 6, pp. 1073–1082, 1998.

[10] S. Gabrielsson, *A parallel tabu search alglorithm for the quadratic assignment problem*, 2008.

[11] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the ACM (JACM)*, vol. 23, no. 3, pp. 555–565, 1976.

[12] R. E. Burkard, E. Cela, P. M. Pardalos, and L. S. Pitsoulis, "The quadratic assignment problem," in *Handbook of combinatorial optimization*, Springer, 1998, pp. 1713–1809.

[13] R. E. Burkard and J. Offermann, "Entwurf von schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme," *Zeitschrift für Operations Research*, vol. 21, no. 4, B121–B132, 1977.

[14] A. N. Elshafei, "Hospital layout as a quadratic assignment problem," *Journal of the Operational Research Society*, vol. 28, no. 1, pp. 167–179, 1977.

[15] J. Dickey and J. Hopkins, "Campus building arrangement using topaz," *Transportation Research*, vol. 6, no. 1, pp. 59–68, 1972.

[16] T. Stützle and M. Dorigo, "Local search and metaheuristics for the quadratic assignment problem," Citeseer, 2001.

[17] J. Clausen, "Branch and bound algorithms-principles and examples," *Department of Computer Science, University of Copenhagen*, pp. 1–30, 1999.

[18] C. Edwards, "A branch and bound algorithm for the koopmans-beckmann quadratic assignment problem," in *Combinatorial optimization II*, Springer, 1980, pp. 35–52.

[19] R. E. Burkard, S. E. Karisch, and F. Rendl, "Qaplib–a quadratic assignment problem library," *Journal of Global optimization*, vol. 10, no. 4, pp. 391–403, 1997.

[20] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Transactions on computers*, no. 9, pp. 917–922, 1977.

[21] P. C. Gilmore, "Optimal and suboptimal algorithms for the quadratic assignment problem," *Journal of the society for industrial and applied mathematics*, vol. 10, no. 2, pp. 305–313, 1962.

[22] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," in *50 Years of Integer Programming 1958-2008*, Springer, 2010, pp. 105–132.

[23] E. D. Taillard, "Comparison of iterative searches for the quadratic assignment problem," *Location science*, vol. 3, no. 2, pp. 87–105, 1995.

[24] Z. Drezner, "Finding a cluster of points and the grey pattern quadratic assignment problem," *OR Spectrum*, vol. 28, no. 3, pp. 417–436, 2006.

[25] K. M. Anstreicher and N. W. Brixius, "Solving quadratic assignment problems using convex quadratic programming relaxations," *Optimization Methods and Software*, vol. 16, no. 1-4, pp. 49–68, 2001.

[26] P. M. Pardalos, H. Wolkowicz, *et al.*, *Quadratic Assignment and Related Problems: DIMACS Workshop, May 20-21, 1993*. American Mathematical Soc., 1994, vol. 16.

[27] T. Mautor and C. Roucairol, "Difficulties of exact methods for solving the quadratic assignment problem," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 16, pp. 263–274, 1994.

[28] W. Zhang, A. Maleki, M. A. Rosen, and J. Liu, "Optimization with a simulated annealing algorithm of a hybrid system for renewable energy including battery and hydrogen storage," *Energy*, vol. 163, pp. 191–207, 2018, ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2018.08.112. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544218316487.

[29] A. Silverman and J. Adler, "Animated simulated annealing," *Computers in Physics*, vol. 6, no. 3, pp. 277–281, 1992.

[30] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[31] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," 1970.

[32] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[33] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*, Springer, 1987, pp. 7–15.

[34] M. R. Wilhelm and T. L. Ward, "Solving quadratic assignment problems by 'simulated annealing'," *IIE transactions*, vol. 19, no. 1, pp. 107–119, 1987.

[35] R. E. Burkard and F. Rendl, "A thermodynamically motivated simulation procedure for combinatorial optimization problems," *European Journal of Operational Research*, vol. 17, no. 2, pp. 169–174, 1984.

[36] U. Thonemann and A Bolte, "An improved simulated annealing algorithm for the quadratic assignment problem," in *Working paper, School of Business, Department of Production and Operations Research*, 1994.

[37] J. Gruska, *Quantum computing*. Citeseer, 1999, vol. 2005.

[38] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[39] V. N. Smelyanskiy, E. G. Rieffel, S. I. Knysh, *et al.*, "A near-term quantum computing approach for hard computational problems in space exploration," *arXiv preprint arXiv:1204.2821*, 2012.

[40] D. J. Griffiths and D. F. Schroeter, *Introduction to quantum mechanics*. Cambridge University Press, 2018.

[41] V. S. Denchev, S. Boixo, S. V. Isakov, *et al.*, "What is the computational value of finite-range tunneling?" *Physical Review X*, vol. 6, no. 3, p. 031 015, 2016.

[42] A. Asfaw, L. Bello, Y. Ben-Haim, *et al.*, *Learn Quantum Computation Using Qiskit*. 2020. [Online]. Available: `http://community.qiskit.org/textbook`.

[43] E. Ising, "Beitrag zur theorie des ferromagnetismus," *Zeitschrift für Physik*, vol. 31, no. 1, pp. 253–258, 1925.

[44] B. A. Cipra, "The ising model is np-complete," *SIAM News*, vol. 33, no. 6, pp. 1–3, 2000.

[45] N. Moll, P. Barkoutsos, L. S. Bishop, *et al.*, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Science and Technology*, vol. 3, no. 3, p. 030 503, 2018.

[46] L. Braine, D. J. Egger, J. Glick, and S. Woerner, "Quantum algorithms for mixed binary optimization applied to transaction settlement," *arXiv preprint arXiv:1910.05788*, 2019.

[47] H. Ushijima-Mwesigwa, R. Shaydulin, C. F. Negre, S. M. Mniszewski, Y. Alexeev, and I. Safro, "Multilevel combinatorial optimization across quantum architectures," *arXiv preprint arXiv:1910.09985*, 2019.

[48] M. S. A. et al., *Qiskit: An open-source framework for quantum computing*, 2021. DOI: `10.5281/zenodo.2573505`.

[49] A. Kandala, A. Mezzacapo, K. Temme, *et al.*, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, 2017.

[50] D. Wang, O. Higgott, and S. Brierley, "Accelerated variational quantum eigensolver," *Physical review letters*, vol. 122, no. 14, p. 140 504, 2019.

[51] A. Peruzzo, J. McClean, P. Shadbolt, *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, p. 4213, 2014.

[52] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, "Quantum computational chemistry," *Reviews of Modern Physics*, vol. 92, no. 1, p. 015 003, 2020.

[53] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[54] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, "Benchmarking the quantum approximate optimization algorithm," *Quantum Information Processing*, vol. 19, pp. 1–24, 2020.

[55] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Physical Review X*, vol. 10, no. 2, p. 021 067, 2020.

[56] M. Streif and M. Leib, "Training the quantum approximate optimization algorithm without access to a quantum processing unit," *Quantum Science and Technology*, vol. 5, no. 3, p. 034 008, 2020.

[57] J. C. Spall, "A stochastic approximation technique for generating maximum likelihood parameter estimates," in *1987 American control conference*, IEEE, 1987, pp. 1161–1167.

[58] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992. DOI: 10.1109/9.119632.

[59] S. Sheldon, L. S. Bishop, E. Magesan, S. Filipp, J. M. Chow, and J. M. Gambetta, "Characterizing errors on qubit operations via iterative randomized benchmarking," *Physical Review A*, vol. 93, no. 1, p. 012 301, 2016.

[60] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, pp. 1–6, 2018.

[61] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, "An initialization strategy for addressing barren plateaus in parametrized quantum circuits," *Quantum*, vol. 3, p. 214, 2019.

[62] A. G. Rattew, S. Hu, M. Pistoia, R. Chen, and S. P. Wood, "A domain-agnostic, noise-resistant, hardware-efficient evolutionary variational quantum eigensolver," *arXiv: Quantum Physics*, 2019.

[63] A. Church, "Am turing. on computable numbers, with an application to the entscheidungs problcm. proceedings of the london mathematical society, 2 s. vol. 42 (1936–1937), pp. 230–265.," *The journal of symbolic logic*, vol. 2, no. 1, pp. 42–43, 1937.

[64] L. Fortnow and S. Homer, "Computational complexity," in *Handbook of the History of Logic*, vol. 9, Elsevier, 2014, pp. 495–521.

[65] V. S. Denchev, N. Ding, S. Vishwanathan, and H. Neven, "Robust classification with adiabatic quantum optimization," *arXiv preprint arXiv:1205.1148*, 2012.

[66] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM Journal on computing*, vol. 26, no. 5, pp. 1411–1473, 1997.

[67] J. Watrous, "Quantum computational complexity," *arXiv preprint arXiv:0804.3401*, 2008.

[68] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM journal on Computing*, vol. 26, no. 5, pp. 1510–1523, 1997.

[69] A. Montanaro, "Quantum algorithms: An overview," *npj Quantum Information*, vol. 2, no. 1, pp. 1–8, 2016.

[70] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Physical review letters*, vol. 79, no. 2, p. 325, 1997.

[71] T. F. Rønnow, Z. Wang, J. Job, *et al.*, "Defining and detecting quantum speedup," *Science*, vol. 345, no. 6195, pp. 420–424, 2014.

[72] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[73] W. Chmiel and J. Kwiecień, "Quantum-inspired evolutionary approach for the quadratic assignment problem," *Entropy*, vol. 20, no. 10, p. 781, 2018, ISSN: 1099-4300. DOI: 10.3390/e20100781. [Online]. Available: http://dx.doi.org/10.3390/e20100781.

[74] J. Choo, "Investigating the feasibility of solving the quadratic assignment problem using quantum computing,"

[75] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, "Improving variational quantum optimization using cvar," *Quantum*, vol. 4, p. 256, 2020.

[76] V. Verteletskyi, T.-C. Yen, and A. F. Izmaylov, "Measurement optimization in the variational quantum eigensolver using a minimum clique cover," *The Journal of chemical physics*, vol. 152, no. 12, p. 124 114, 2020.

[77] H. Chieza, M. Khumalo, K Prag, and M Woolway, "On the computational performance of ibm quantum devices applied to combinatorial optimisation problems," in *2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, IEEE, 2020, pp. 260–264.

[78] G. E. Crooks, "Performance of the quantum approximate optimization algorithm on the maximum cut problem," *arXiv preprint arXiv:1811.08419*, 2018.

[79] M. P. Harrigan, K. J. Sung, M. Neeley, *et al.*, "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," *Nature Physics*, pp. 1–5, 2021.

[80] E. Farhi, J. Goldstone, S. Gutmann, and H. Neven, "Quantum algorithms for fixed qubit architectures," *arXiv preprint arXiv:1703.06199*, 2017.

[81] E. Farhi, D. Gamarnik, and S. Gutmann, "The quantum approximate optimization algorithm needs to see the whole graph: A typical case," *arXiv preprint arXiv:2004.09002*, 2020.

[82] M. T. Khumalo, H. A. Chieza, K. Prag, and M. Woolway, "An investigation of ibm quantum computing device performance on combinatorial optimisation problems," *arXiv preprint arXiv:2107.03638*, 2021.

[83] G. G. Guerreschi and A. Y. Matsuura, "Qaoa for max-cut requires hundreds of qubits for quantum speed-up," *Scientific reports*, vol. 9, no. 1, pp. 1–7, 2019.

[84] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, "Matrix product state representations," *arXiv preprint quant-ph/0608197*, 2006.

[85] R. Tao, Y. Shi, J. Yao, J. Hui, F. T. Chong, and R. Gu, *Gleipnir: Toward practical error analysis for quantum programs (extended version)*, 2021. arXiv: 2104.06349 [cs.PL].

[86] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Physical Review A*, vol. 100, no. 3, p. 032 328, 2019.

[87] M. Takita, A. W. Cross, A. Córcoles, J. M. Chow, and J. M. Gambetta, "Experimental demonstration of fault-tolerant state preparation with superconducting qubits," *Physical review letters*, vol. 119, no. 18, p. 180 501, 2017.

[88] J. B. Hertzberg, E. J. Zhang, S. Rosenblatt, *et al.*, "Laser-annealing josephson junctions for yielding scaled-up superconducting quantum processors," *npj Quantum Information*, vol. 7, no. 1, pp. 1–8, 2021.

[89] E. Zahedinejad and A. Zaribafiyan, "Combinatorial optimization on gate model quantum computers: A survey," *arXiv preprint arXiv:1708.05294*, 2017.

[90] S. van den Heever, "Decision optimization: The key differentiator for next-generation analytics," 2015.

[91] O. Goldreich, *P, NP, and NP-Completeness: The basics of computational complexity*. Cambridge University Press, 2010.

[92] J. Gacon, C. Zoufal, G. Carleo, and S. Woerner, "Simultaneous perturbation stochastic approximation of the quantum fisher information," *Quantum*, vol. 5, p. 567, 2021.