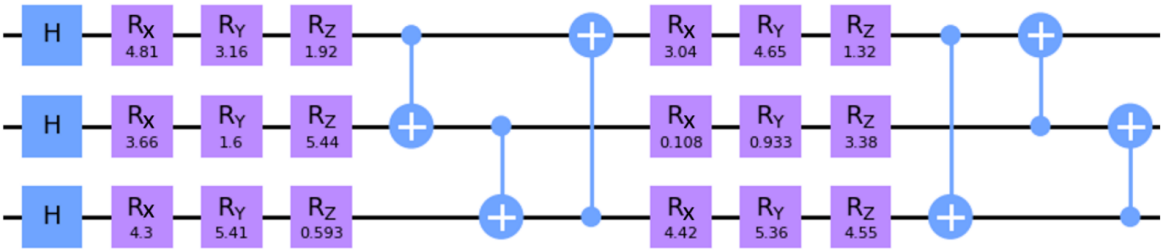
TÉCNICO LISBOA



Development of a new variational quantum algorithm for MaxCut: QAOA and QEMC hybrid algorithm

Afonso Sequeira Azenha

Thesis to obtain the Master of Science Degree in

Engineering Physics

Supervisor(s): Prof. Yasser Rashid Revez Omar
Prof. Joao Carlos Carvalho de Sá Seixas

Examination Committee

Chairperson: Prof. Full Name 1
Supervisor: Prof. Full Name 2
Member of the Committee: Prof. Full Name 3

June 2024

Dedicated to someone special...

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

A few words about the university, financial support, research advisor, dissertation readers, faculty or other professors, lab mates, other friends and family...

Resumo

Inserir o resumo em Português aqui com um máximo de 250 palavras e acompanhado de 4 a 6 palavras-chave.

Palavras-chave: palavra-chave1, palavra-chave2, palavra-chave3,...

Abstract

Insert your abstract here with a maximum of 250 words, followed by 4 to 6 keywords.

Keywords: keyword1, keyword2, keyword3,...

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

$|\psi\rangle$ Arbitrary pure state

ρ Density matrix or performance guarantee

σ_x, σ_y, σ_z Pauli matrices

Roman symbols

$\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ Pauli matrices (alternative notation)

$R_x(\theta), R_y(\theta), R_z(\theta)$ Rotation (of angle θ) gates around the x, y and z axes, on the Bloch sphere.

S_n n-dimensional unit sphere

Subscripts

x, y, z Cartesian components

Superscripts

† Hermitian adjoint

T Transpose

List of acronyms

AR	Approximation Ratio
GW	Goemans-Williamson Algorithm
HQCC	Hybrid Quantum-Classical Computing
iQAQE	Interpolated QAOA/QEMC Hybrid Algorithm
MaxCut	Maximum Cut Problem
NISQ	Noisy Intermediate-Scale Quantum
PQC	Parameterized Quantum Circuit
QAOA	Quantum Approximate Optimization Algorithm
QEMC	Qubit-Efficient MaxCut Heuristic Algorithm
QKD	Quantum Key Distribution
SAT	Boolean Satisfiability Problem
TSP	Traveling Salesman Problem
VQA	Variational Quantum Algorithm

Chapter 1

Introduction

Over the last few decades, significant progress has been achieved in the field of quantum computing. This is an entirely new computing paradigm, based on exploiting the fundamental principles of quantum mechanics to our advantage. Although the concept of a quantum computer has been around for a little over 40 years [1], only rather recently, in the present century, have we successfully engineered elementary prototypes for such cutting-edge devices. Scientists and engineers worldwide are now working towards the development of practical quantum computers, which are expected to revolutionize the way we solve complex problems in various fields, such as cryptography, optimization, drug discovery, material science, machine learning, and many more.

However, with the current generation of quantum machines, so-called Noisy Intermediate Scale Quantum (NISQ) devices, we are still far from achieving the promised quantum advantage. This refers to the point at which a quantum computer can outperform its classical counterpart by solving specific problems considerably faster or more efficiently. It's the "holy grail" of quantum computing and what drives the research in this field forwards.

Presently, the most promising candidates for achieving meaningful quantum advantage are anchored in what has become known as "Hybrid Quantum-Classical Computing" (HQCC). This approach aims to merge the strengths of both worlds: the computational power of quantum and the stability of classical computing. In this context, Variational Quantum Algorithms (VQA) have been at the forefront of research, as they are particularly well-suited for near-term quantum devices, requiring fewer qubits and featuring shallower circuit depths. These algorithms are hybrid by design, using a classical optimizer to adjust the parameters of a Parameterized Quantum Circuit (PQC).

Amid the many prospective applications of quantum computing, notable advances have been made in recent years in solving combinatorial optimization problems. VQAs like the Quantum Approximate Optimization Algorithm (QAOA) [2] have demonstrated potential for tackling the Maximum Cut (MaxCut) problem, a challenging graph-based NP-hard problem in computer science. Despite its promise, however, QAOA has a significant drawback: it requires a large number of qubits, exceeding the capacity of current quantum devices, when scaled to larger problem instances. This constraint has prompted the development of alternative approaches such as the Qubit-Efficient MaxCut Heuristic Algorithm (QEMC)

[3], designed to address the MaxCut problem using fewer qubits. Meanwhile, the search for better algorithms to solve this problem continues, and the development of new hybrid quantum-classical methods is crucial to achieving quantum advantage.

1.1 Motivation

The search for efficient algorithms to solve combinatorial optimization problems is critical in numerous fields, such as logistics, finance, and telecommunications. The MaxCut problem, for example, has broad applications in areas like machine learning [4], statistical physics [5], circuit design [5], and data clustering [6]. Creating more efficient algorithms to solve this problem can improve the performance of these applications, driving substantial progress in their respective fields.

Moreover, there is a strong interest from the computer science community in terms of computational complexity. The MaxCut problem is recognized as NP-hard, and the search for efficient algorithms to solve it may offer valuable insights into the boundaries between classical and quantum computing. It might even contribute to unraveling one of the most perplexing questions in theoretical computer science: the $P = NP$ problem. Imagining a world where the $P = NP$ conjecture is proven true, albeit improbable, is intriguing. It would mean that every problem in NP could be solved in polynomial time, including MaxCut. This would also extend to problems like the Traveling Salesman Problem (TSP) and the Knapsack Problem, among others. The implications would be profound, as this would dramatically increase our ability to solve previously difficult optimization problems, with direct applications in areas like vehicle routing, job scheduling, and broader logistics. Additionally, the impact on cryptography would be as significant – if not more so – since many cryptographic techniques rely on the complexity of NP problems. For example, integer factorization is a key component of RSA (Rivest-Shamir-Adleman) encryption, a popular asymmetric encryption algorithm used extensively in public-key cryptography for secure message transmission over the internet. If $P = NP$, RSA encryption could easily be broken, leading to major security risks. Hence, the importance of studying these problems to fully understand their complexity.

The aforementioned considerations drive our efforts to develop a new algorithm for solving the MaxCut problem with greater efficiency and accuracy. This algorithm, the Interpolated QAOA/QEMC Hybrid Algorithm (iQAQE), will be the focus of this thesis.

1.2 Topic Overview

In this project, we propose a new VQA, the Interpolated QAOA/QEMC Hybrid Algorithm (iQAQE). This algorithm combines the strengths of two existing VQAs, QAOA and QEMC, for improved performance in solving the MaxCut problem. As previously mentioned, QAOA requires a qubit for each graph vertex, making it difficult to scale. In contrast, QEMC uses exponentially fewer qubits by assigning one basis state to each graph node, requiring only $\log_2(n)$ qubits (for n graph vertices). However, this compression leads to limitations in QEMC's results. By interpolating both VQAs, we aim to create an

algorithm that utilizes fewer qubits than QAOA and performs better than QAOA and QEMC. The new algorithm, iQAQE, assigns multiple basis states to each node, unlike QEMC's single basis state approach. This design tentatively allows for a more practical implementation on present-day NISQ devices, thanks to its reduced qubit requirements compared to QAOA, fewer measurement shots than QEMC, and potentially greater trainability than QAOA.

1.3 Objectives and Deliverables

The primary objective of this thesis is to develop and analyze the iQAQE algorithm. The algorithm will be implemented and tested using classical simulations of quantum machines. The deliverables for this project include the iQAQE algorithm's code, the results obtained from the simulations, and the analysis of these results. The expected outcomes are improvements in the performance of the iQAQE algorithm compared to QAOA and QEMC, with a focus on accuracy, efficiency, and scalability.

1.4 Thesis Outline

This thesis is structured as follows: Chapter 2 provides an overview of the background concepts related to quantum computing, variational quantum algorithms, and the MaxCut problem. Chapter 3 introduces the base hybrid quantum-classical algorithm, iQAQE, explaining its design, advantages, and potential applications. Chapter 4 details the numerical implementation of the iQAQE algorithm, including the individual algorithms QAOA and QEMC. It also describes the benchmarking and testing methods used to evaluate the algorithm's performance. Chapter 5 presents the schemes and results obtained from the simulations, comparing iQAQE with QAOA and QEMC. It also outlines the various iQAQE variations that were explored. Finally, Chapter 6 concludes the thesis, summarizing the main findings and suggesting future research directions.

Chapter 2

Theoretical background

This chapter covers the key concepts and theoretical background necessary to understand the work presented in this thesis. It begins by describing the MaxCut problem and discussing elements of computational complexity, reinforcing the rationale behind this research. The state-of-the-art algorithms for solving the MaxCut problem, both classical and quantum, are then examined. After a brief introduction to quantum computing, the chapter explores hybrid quantum-classical computing and variational quantum algorithms in greater depth. Specifically, the Quantum Approximate Optimization Algorithm (QAOA) and the Qubit-Efficient MaxCut Heuristic (QEMC) are analyzed, as they play a pivotal role in this study.

2.1 The Maximum Cut Problem & Computational Complexity

The MaxCut problem, a fundamental problem in graph theory and combinatorial optimization, involves partitioning a graph $G = (V, E)$ into two disjoint subsets, S_1 and S_2, such that the number of edges connecting vertices from different subsets is maximized. Formally, the objective is to find a partition (S_1, S_2) that maximizes:

$$\text{Cut}(S_1, S_2) = \sum_{(u,v)\in E} \chi(u, v), \tag{2.1}$$

where $\chi(u, v) = 1$ if u and v belong to different subsets, and $\chi(u, v) = 0$ if they belong to the same subset. This quantity is referred to as the ”cut” of the partition (S_1, S_2). Pictorially, this can be represented as cutting the edges of the graph (Figure 2.1), hence the name MaxCut. What we describe here is the undirected, un-weighted MaxCut problem. A more general formulation would involve the specific graph's adjacency matrix, W_{ij}.

Since the MaxCut problem is NP-hard, finding an optimal solution efficiently is a significant computational challenge, especially as the graph size grows. However, researchers have developed various approximation algorithms and heuristics to approach near-optimal solutions in a reasonable time, including both classical and quantum approaches (cf. subsections 2.1.1 and 2.1.2). These methods are designed to tackle the intrinsic complexity of the problem, providing practical solutions that have real-

world applications. As mentioned earlier, the MaxCut problem finds use in a variety of fields, including machine learning [4], statistical physics [5], circuit design [5], and data clustering [6].

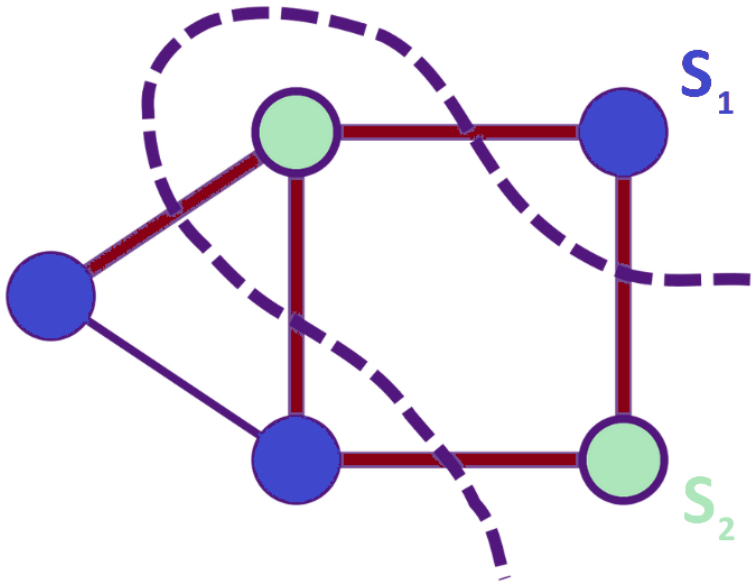


Figure 2.1: An example of a graph with a partition that maximizes the number of cut edges (red). Note that the MaxCut partition might not be unique.

We previously mentioned that the MaxCut problem is well-known to be NP-hard. (Indeed, it [decision version] even appears on Karp's original list of NP-complete problems, from 1972 [7].) Let's explore what this means in more detail. The computational complexity class of a problem is defined by the computational resources required to solve it, such as time or memory. Although there are many complexity classes (see the Complexity Zoo), the following four are key to our discussion:

1. **P (Polynomial time)**: Problems that can be solved in polynomial time by a deterministic Turing machine, indicating they are computationally efficient.
2. **NP (Non-deterministic Polynomial time)**: Problems that can be verified in polynomial time if given a solution, but finding the solution may not be as straightforward. Therefore, if you guess a solution, you can confirm its correctness quickly.
3. **NP-hard (Non-deterministic Polynomial time hard)**: Problems that are at least as challenging as the most difficult problems in NP. If you could solve an NP-hard problem in polynomial time, you would be able to solve all problems in NP efficiently.
4. **NP-complete (Non-deterministic Polynomial time complete)**: Problems that belong to NP but are also NP-hard. If you can solve one NP-complete problem efficiently, all problems in NP would become efficiently solvable.

The MaxCut problem in its **optimization form** – "Given a graph G, find the maximum cut" – is NP-hard. This implies that if you could find an efficient solution to MaxCut, you could also solve all other NP

problems reliably. That would include, e.g., the boolean SATisfiability problem (SAT), knapsack problem, decision version of the traveling salesman problem, clique problem, and more [8]. This potential for broad application drives interest in developing high-performance algorithms for MaxCut. Alternatively, the MaxCut problem can be formulated as a **decision problem** – "Given a graph G and an integer k, determine if there is a cut of size at least k in G." This formulation is NP-complete. NP-complete status indicates that a problem is both in NP, allowing polynomial-time verification, and at least as hard as any other problem in NP, making it central to computational theory and offering crucial insights into the boundary between efficiently solvable and intractable problems. Although our focus is on the NP-hard version of the problem, it's important to understand the significance of the NP-complete formulation.

2.1.1 Classical State-of-the-Art Algorithms for MaxCut

Over the years, a variety of algorithms have been proposed to solve the MaxCut problem, ranging from exact solutions to various approximations, including heuristics and other techniques. In this work, we focus on scaling these algorithms to handle bigger graphs, which is why exact algorithms are not discussed – they quickly become impractical for large instances. Instead, our attention is on approximation algorithms, which are more suitable for larger scales[1].

Approximation algorithms aim to find solutions that are close to optimal, often with a guaranteed Approximation Ratio (AR). This is a measure of the performance of an approximation algorithm, representing the worst-case deviation between the algorithm's solution and the optimal solution. Formally, this can be expressed as: (For the Goemans-Williamson (GW) Algorithm [9], e.g.)[2]

$$r_A^{GW} = \frac{\text{MaxCut}_{\text{GW}}}{\text{MaxCut}_{\text{Opt.}}} \approx 0.87856,$$

where $\text{MaxCut}_{\text{GW}}$ is the worst-case MaxCut value obtained by the Goemans-Williamson Algorithm, and $\text{MaxCut}_{\text{Opt.}}$ is the optimal MaxCut value. The AR is a crucial metric for evaluating the quality of an approximation algorithm, providing insight into its effectiveness. The closer the AR is to 1, the better the algorithm's performance. In literature, this worst-case approximation ratio might also be referred to as the *performance guarantee* ρ, often appearing alongside designations such as ρ-approximation algorithms. Using this nomenclature, the Goemans-Williamson Algorithm is a 0.87856-approximation algorithm for MaxCut (i.e., $\rho = 0.87856$).

Building on this, two commonly used **approximation algorithms** are:

- **Goemans-Williamson Algorithm** [9]: This popular approximation algorithm achieves a performance guarantee of approximately 0.87856, leveraging semidefinite programming and a randomized rounding technique. (This will be further developed below, in subsubsection Goemans-Williamson Algorithm.)
- **Spectral Methods** [10]: These techniques employ eigenvalues and eigenvectors of the adjacency matrix to identify effective cuts. While spectral methods may offer quicker computational speeds

[1] In this work, "larger/bigger graphs" will often refer to graphs with many hundreds to a few thousand nodes.
[2] The actual value is $\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos\theta} > 0.878$.

compared to other approximation strategies (e.g. GW), they may not consistently provide the optimal approximation ratios. Performance-wise, the leading spectral algorithm, Trevisan's algorithm, using Soto's refined analysis [11], features a worst-case approximation ratio of 0.614.

Heuristic and **metaheuristic algorithms** offer another approach, focusing on finding solutions more quickly, albeit without guaranteed approximation ratios. They are typically employed when computational efficiency is prioritized over achieving the best possible approximation ratio. These algorithms include:

- **Greedy Algorithms** (Described in [10]): These build solutions incrementally, following a greedy approach. Mathematically, we start with $S_1, S_2 = \emptyset$. In each step of the algorithm, we choose a vertex $v \in V$ yet to be assigned to S_1 or S_2. Then, we add v to S_1 or S_2 by choosing the larger of the two cuts $(S_1 \cup \{v\}, S_2)$ and $(S_1, S_2 \cup \{v\})$ at this step. Repeat until all vertices are assigned to a set.
- **Simulated Annealing** [12]: Employs a probabilistic optimization approach, inspired by metallurgical annealing, to iteratively explore the solution space, gradually reducing the acceptance of worse solutions ("temperature") over time to converge towards near-optimal solutions.
- **Genetic Algorithms** [13]: These algorithms, rooted in evolutionary principles, employ selection, crossover, and mutation techniques to iteratively generate solutions for the MaxCut problem, evolving from an initial population of diverse partitions towards improved solutions.

Reiterating, these approximation and heuristic methods offer scalable solutions for the MaxCut problem, making them more suitable for larger graphs where exact methods are not feasible.

Of all the mentioned approaches, we will now elaborate on the Goemans-Williamson algorithm, which currently stands as the most effective approximation method for MaxCut, boasting the best-known performance guarantee to date (0.87856). Its detailed explanation is particularly relevant as we plan to utilize it extensively in benchmarking our proposed algorithms later on. We opt for GW over other options for one obvious reason: Our aim is to push the boundaries of MaxCut algorithms, thus it is logical to compare against the top-performing method. (We do not use any of the heuristic/metaheuristic algorithms, as they do not provide performance guarantees, and we are interested in having a benchmark to compare against.)

2.1.1.1 Goemans-Williamson Algorithm

In this subsubsection, we shall present the Goemans-Williamson Algorithm in greater detail. This approximation algorithm, developed by Michel X. Goemans and David P. Williamson in 1995 [9], is a cornerstone in the field of combinatorial optimization, particularly for the MaxCut problem. The algorithm leverages semidefinite programming to construct a solution that is then rounded to provide a near-optimal cut. Semidefinite programming involves optimizing a linear function of a symmetric matrix while adhering to linear equality constraints and the requirement that the matrix be positive semidefinite. This programming paradigm finds utility across diverse fields such as control theory, nonlinear programming,

geometry, and combinatorial optimization, showcasing its versatility and applicability. (These applications are detailed in [9] and references therein.)

Formally, the Goemans-Williamson Algorithm can be described as follows. (We will use the original formulation, from [9], presenting a distilled version of the most important aspects of their description.) Given a graph, with vertex set $V = \{1, ..., n\}$ and non-negative weights $w_{ij} = w_{ji}$ for each pair of vertices i and j, the weight of the maximum cut $w(S_1, S_2)$ is given by the following integer quadratic program[3]:

$$\begin{aligned} & \text{Maximize } \frac{1}{2}\sum_{i<j} w_{ij}(1 - y_i y_j) \\ (Q) \qquad & \text{Subject to: } y_i \in \{-1, 1\} \qquad \forall i \in V. \end{aligned} \tag{2.2}$$

It is clear that the sets $S_1 = \{i | y_i = 1\}$ and $S_2 = \{i | y_i = -1\}$ correspond to a cut of weight $w(S_1, S_2) = \frac{1}{2}\sum_{i<j} w_{ij}\,(1 - y_i y_j)$. This represents the same quantity as the aforementioned Cut$(S_1, S_2) = \sum_{(u,v)\in E} \chi(u, v)$ (Eq. 2.1), in the un-weighted case. Explicitly, in words, the sum in Eq. 2.2 is to be taken over all the graph's edges (i, j), with y_i representing node-i's associated set: $y_i = +1$, if node-$i \in S_1$, or $y_i = -1$, when node-$i \in S_2, \forall i \in V$.

Given that solving this integer quadratic program is NP-complete [9], we explore relaxations of (Q). These are obtained by relaxing some of the constraints of (Q), and extending the objective function to the larger space. Consequently, all solutions of (Q) are feasible for the relaxation, and the optimal value of the relaxation serves as an upper bound for (Q)'s optimal value. We interpret (Q) as constraining y_i to be a 1-dimensional unit vector. A number of possible relaxations involve allowing y_i to be a multidimensional vector v_i of unit Euclidean norm. As the linear space spaned by the vectors v_i has dimension at most n (number of graph nodes), we can assume that these vectors belong to $\mathbb{R}^n$. More precisely, they belong to the n-dimensional unit sphere, S_n. At this point, so as to make sure that the resulting optimization problem is, indeed, a relaxation, we must define the cost function such that it reduces to $\frac{1}{2}\sum_{i<j} w_{ij}(1 - y_i y_j)$ for vectors lying in a 1-dimensional space. One way to do this is to replace $y_i y_j$ with the inner product $v_i \cdot v_j$ in the expression for $w(S_1, S_2)$. The resulting relaxation, (P), reads:

$$\begin{aligned} & \text{Maximize } \frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i \cdot v_j) \\ (P) \qquad & \text{Subject to: } v_i \in S_n \qquad \forall i \in V. \end{aligned} \tag{2.3}$$

In the context of this semidefinite programming relaxation, the GW algorithm follows these steps:

1. Solve (P), thus obtaining an optimal set of vectors v_i;
2. Let r be a vector uniformly distributed on the unit sphere, S_n;
3. Set $S_1 = \{i | v_i \cdot r \geq 0\}$ and $S_2 = \{i | v_i \cdot r < 0\}$.

[3]Note that we use the general formulation here, contemplating the possibility of weighted graphs, i.e., we do not restrict ourselves to un-weighted graphs.

In other words, we choose a random hyperplane in n dimensions, crossing through the origin, and partition the vertices according to whether the vectors v_i lie "above" ($v_i \cdot r \geq 0$) or "below" ($v_i \cdot r < 0$) this hyperplane. Those above the hyperplane are assigned to S_1, and those below to S_2.

Furthermore, it can be shown that the GW algorithm has a performance guarantee of:

$$\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos\theta} > 0.878. \tag{2.4}$$

A detailed proof elucidating the appearance of this 0.878 value is present in section 3 of [9].

Computationally, one defines the positive semidefinite matrix $X \in \mathbb{R}^{n \times n}$, with $X_{ii} = 1$ and $X_{ij} = v_i \cdot v_j = v_i^T v_j, \forall i, j \in V$. Then, we can solve the following semidefinite program:

$$\begin{aligned} &\text{Maximize } \frac{1}{2} \sum_{\text{Edges}} w_{ij}(1 - X_{ij}) \\ (N) \qquad &\text{Subject to: } X_{ii} = 1, X \succeq 0. \end{aligned} \tag{2.5}$$

We do possess effective tools and techniques for solving semidefinite programs, enabling the efficient determination of the optimal X matrix. From there, we can extract the vectors v_i, by performing the square root operation on the matrix X. The resulting matrix's columns, or lines – X is symmetric – will correspond to the desired vectors v_i. This outlines the process for solving (P) as listed earlier.

2.1.2 Hybrid Quantum-Classical State-of-the-Art Algorithms for MaxCut

A number of different hybrid quantum-classical algorithms have been proposed to solve the MaxCut problem, leveraging the unique properties of quantum computing to potentially outperform purely classical algorithms. They are designed to exploit quantum superposition and entanglement to explore the solution space more efficiently. In this section, we will briefly mention the state-of-the-art hybrid quantum-classical algorithms for MaxCut, focusing on the Quantum Approximate Optimization Algorithm (QAOA) and the Qubit-Efficient MaxCut Heuristic Algorithm (QEMC). Due to their variational nature, these algorithms do not have a theoretical performance guarantee, but they have shown promising results in practice.

Presently, the Quantum Approximate Optimization Algorithm (QAOA), initially introduced in [2], is one of the top candidates for achieving meaningful quantum advantage with near-term quantum devices. Although, MaxCut-wise, it has a substantial limitation: it requires a number of qubits equal to the number of graph nodes. This is a significant drawback, as it severely limits the algorithm's scalability to bigger graphs. Hence, why it is believed that "QAOA for Max-Cut requires hundreds of qubits for quantum speed-up" [14]. After all, our classical algorithms work just fine for small graphs, which is why we do not present performance metrics for basic QAOA. Alternatively, different QAOA variations have been proposed to address the issue of high qubit requirements, such as QAOA-in-QAOA [15], which partitions a large graph into many smaller subgraphs, each of which is easily solved using a separate QAOA instance. Afterwards, these results are joined together by working through another, this time **weighted**, MaxCut problem. This approach in particular has proven to yield competitive or even better performance

over the best known classical algorithms, i.e. GW, for graphs up to 2000 nodes.

A separate problem one might face when implementing QAOA, that also hinders its application to larger graphs, has to do with qubit connectivity[4], as QAOA might require specific connectivity patterns that are not natively available in present-day NISQ devices. To implement these, we are often required to utilize a number of 2-qubit SWAP gates, which degrade the algorithm's performance by introducing extra noise in the system, and hence errors in our results. Therefore, a parallel line of work has been to develop QAOA variations that have lesser connectivity requirements, in the sense that they do not depend on arbitrary qubit connectivity. For example, Parity-QAOA [16, 17] was designed for quantum chips with planar lattice geometry (Figure 2.2), requiring only nearest-neighbour connectivity. Although it necessitates more qubits, it entirely solves this problem.

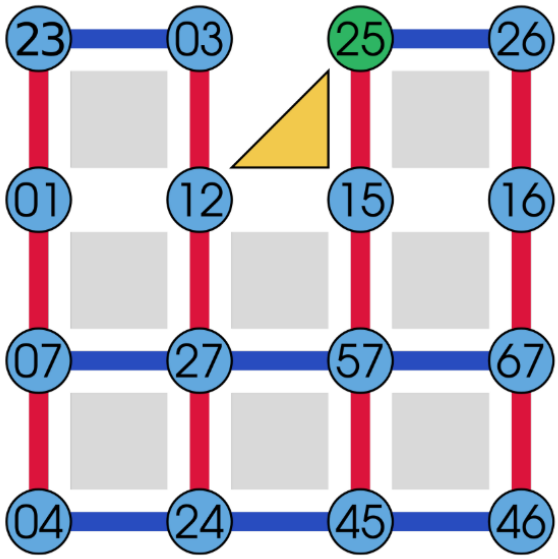


Figure 2.2: Planar lattice geometry illustration: the circular nodes represent the physical qubits, each of which can interact directly with its nearest neighbours. The lines represent terms of the driver Hamiltonian, from Parity-QAOA [16]. The numbers refer to edges in the original graph. Image reproduced from [16].

Furthermore, the Qubit-Efficient MaxCut Heuristic Algorithm (QEMC), introduced in [3], is a quantum-classical hybrid algorithm that aims to solve the MaxCut problem using fewer qubits than QAOA, therefore extending its applicability to larger graphs. It has shown cutting-edge performance in practice, surpassing the classical state-of-the-art, i.e. GW, for graphs up to 2048 nodes. The novelty of this algorithm is that it opens up the door for quantum devices to be used for large MaxCut instances, previously unfeaseble due to qubit number limitations, which rendered QAOA computations for such large graphs impossible. As was already mentioned, QEMC allows for an exponential compression in the number of needed qubits, enabling its utilization on today's quantum hardware for considerably larger graphs than what would be possible with traditional QAOA. However, this exponential compression also makes it efficiently simulable classically, which effectively defeats its purpose as a quantum algorithm. For this

[4]Qubit connectivity refers to the pattern or arrangement of connections between qubits in a quantum computing system, determining which qubits can interact directly with each other.

reason, QEMC is termed a quantum-inspired classical algorithm and, by definition, will not produce any quantum advantage. Either way, we include it in this subsection, since it can always be implemented as a VQA on current-day NISQ devices.

More recently, different algorithms have been proposed in an attempt to reduce the number of qubits required for larger graphs, without falling into the trap of efficient classical simulability. (After all, we are looking for quantum advantage.) Polynomial compression-based algorithms, such as that proposed in [18], have too shown promising results. This one in specific [18] is, to the best of our knowledge, boasting the highest quality attained experimentally on sizes up to 7000 graph nodes, competitive with state-of-the-art classical solvers. Not only that, but their specific qubit-efficient encoding brings in a super-polynomial mitigation of barren plateaus[5] as a built-in feature, which constitutes a significant advantage over other VQAs. Curiously enough, our initial idea for the iQAQE algorithm is quite similar to what is explored in [18]. However, we believe our algorithm to be somewhat more general, by not assuming *a priori* a polynomial compression in the number of qubits.

2.2 Quantum Computing Primer

In this section, we present a brief introduction to quantum computing, focusing on its fundamental ideas and principles. We begin with an overview of the general concepts, followed by a discussion of quantum gates and circuits, crucial for understanding quantum algorithms. This groundwork will prepare us to delve into the specifics of variational quantum algorithms and their application to the MaxCut problem.

General concepts and mathematical formalism

(This description is inspired by [19].)

Quantum computing makes use of the foundational principles of quantum mechanics in an attempt to extract some sort of quantum advantage from them. Instead of using classical binary digits, quantum computers use their quantum analog, qubits. In practice, qubits can be any two-state quantum system. Said states are, unequivocally, associated with the states $|0\rangle$ and $|1\rangle$, conventionally, very much like in classical computing when one uses bits. The primary distinction lies in the fact that a qubit, being a quantum system, can exist in a superposition of both states generally denoted as:

$$\mathcal{H}^2 \ni |\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.6}$$

where $\alpha, \beta \in \mathbb{C}$, with $|\alpha|^2 + |\beta|^2 = 1$. This ensures probability normalization to 1, as, according to the famous Born rule, the probability of measuring the qubit in state $|0\rangle$ is $|\langle 0|\psi\rangle|^2 = |\alpha|^2$, and in state $|1\rangle$ is $|\langle 1|\psi\rangle|^2 = |\beta|^2$.

In the same vein, one can compose (represented by the tensor product) n qubits to obtain an n-qubit

[5] In VQA analysis, barren plateaus are characterized by a vanishing expectation value of $\nabla\mathcal{L}$ over random parameter initializations and an exponential decay (in n) of its variance. Note that $\mathcal{L}$ refers to the loss function. These are, thus, flat regions of the parameter space, impeding optimization.

state:

$$|\psi\rangle = \bigotimes_{i=0}^{n-1}(\alpha_i |0\rangle \ + \beta_i |1\rangle). \tag{2.7}$$

More generally, we can express an arbitrary pure state $|\psi\rangle$ by a normalized linear combination of the computational basis states $|b_0 b_1 ... b_{n-1} b_n\rangle \coloneqq |b_0\rangle \otimes |b_1\rangle \otimes ... \otimes |b_{n-1}\rangle \otimes |b_n\rangle$:

$$|\psi\rangle = \sum_{\boldsymbol{b}\in\{0,1\}^{\otimes n}} \alpha_{\boldsymbol{b}} |b_0 b_1 ... b_{n-1} b_n\rangle , \tag{2.8}$$

where $\sum_{\boldsymbol{b}\in\{0,1\}^{\otimes n}} |\alpha_{\boldsymbol{b}}|^2 = 1$. This superposition phenomenon (mathematically characterized by the linear combination of basis states), utterly impossible in classical physics, brings unprecedented computational power under certain scenarios, as it introduces a kind of built-in parallelism in quantum computing, a highly desirable feature for any computational scheme.

In addition to this, quantum computing also exploits entanglement as a resource. Entanglement allows for intricate correlations between qubits, which have no classical counterpart and are believed to offer computational speed-ups, although it is yet unclear exactly how. An entangled state, by definition, is one that cannot be expressed as the tensor product of individual qubit states, i.e., it cannot be represented in the form of Eq. 2.7. The most famous example of entangled states are the Bell states, representing what are known as two-qubit (or bipartite) maximally entangled states. Such Bell states are reproduced below:

$$\begin{aligned} \left|\Phi^+\right\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \quad \left|\Phi^-\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle), \\ \left|\Psi^+\right\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \quad \left|\Psi^-\right\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle). \end{aligned} \tag{2.9}$$

Clearly, these states cannot be expressed as a tensor product of two individual qubit states. From a practical standpoint, Bell states frequently serve as foundational elements in advanced techniques like quantum teleportation and quantum cryptography. Quantum Key Distribution (QKD) protocols, for example, frequently rely on entangled Bell pairs to distribute secure keys for encrypting and decrypting messages.

Another key topic in quantum information theory is the distinction between pure and mixed states. So far, we have only been treating pure states, which are represented by a single ket vector in some Hilbert space. Mixed states, on the other hand, are represented by density matrices, which are positive semidefinite matrices with unit trace. These matrices are used to describe a statistical ensemble of quantum states. The density matrix ρ of a quantum state $|\psi\rangle$ is defined as:

$$\rho = |\psi\rangle\langle\psi| \tag{2.10}$$

In the more general case of a statistical ensemble of states $|\psi_i\rangle$, each with probability p_i, the density matrix ρ is given by:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| . \tag{2.11}$$

It's crucial to avoid confusing this with superposition, as a mixed state comprises a statistical ensemble of pure states, while superposition pertains to a single state that is a linear combination of other states. For instance, the state $|\psi_S\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ represents a superposition of $|0\rangle$ and $|1\rangle$, whereas the state $\rho_M = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|)$ denotes a mixed state, signifying that half the states in the ensemble are in state $|0\rangle$ and the other half in state $|1\rangle$. Unlike the probabilistic mixture (ρ_M), the superposition (ψ_S) can exhibit quantum interference, hence why it is important to distinguish between the two. If desired, this distinction can also be observed in the off-diagonal terms of the density matrix, referred to as coherences, which are responsible for quantum interference effects: the off-diagonal terms of ρ_M are zero, whereas those of $\rho_S = |\psi_S\rangle\langle\psi_S| = \frac{1}{4}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| + |1\rangle\langle 1|)$ are not.

Currently, numerous quantum computing architectures are in competition and the ultimate victor remains uncertain. Nevertheless, the most prominent architectures have qubits composed of either photons [20, 21], neutral atoms (using Rydberg states) [22, 23], trapped ions [24], semiconductors (still in their infancy), or, most notably, superconducting circuits [25, 26] – employed by giants like IBM and Google in their quantum processors. Looping back to the definition of a qubit, considering a photonic quantum computer, it becomes rather straightforward and natural to map the photons' vertical and horizontal polarizations to the $|0\rangle$ and $|1\rangle$ states, respectively. Once again, this mapping is analogous to how classical computing maps the current/no current states to 1 and 0.

The next step is to understand how these so-called quantum circuits act on our qubits, and how they can be used to perform quantum computations. This is where quantum gates come into play.

Quantum gates and circuits

Presently, quantum computers are being designed predominantly with a circuit-based architecture. These circuits serve as the quantum equivalent of classical logic circuits, however, instead of the typical AND, OR, and NOT gates, quantum circuits feature quantum gates. These gates apply specific transformations to qubits, based on their definitions, taking an initial state $|\psi\rangle$ to an output state $|\phi\rangle = U|\psi\rangle$, for some gate U, **designed to be unitary**, i.e., $U^\dagger U = UU^\dagger = I$, where I is the identity matrix. Unitary transformations preserve the normalization of quantum states and the inner product between states, which are essential properties in quantum mechanics. Additionally, they ensure that quantum operations are reversible, meaning that information is not lost during computation. This [unitary gates] can also be seen as a natural consequence of the Schrödinger equation, governing the time evolution of quantum systems, requiring said evolution to be described by a unitary operator.

The most frequently used quantum gates consist of rotations around each of the three Cartesian axes, represented as complex exponentials of the Pauli x, y and z matrices (σ_x, σ_y and σ_z). The Pauli matrices are defined as: (Note the alternative notation, $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$, for the sake of clarity.)

$$\boldsymbol{X} \coloneqq \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \boldsymbol{Y} \coloneqq \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \boldsymbol{Z} \coloneqq \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.12}$$

Furthermore, when we mention rotations, these are to be seen in the Bloch sphere, which is a geomet-

rical representation of the pure state space of a two-level quantum system, an illustration of which can be seen in Figure 2.3.

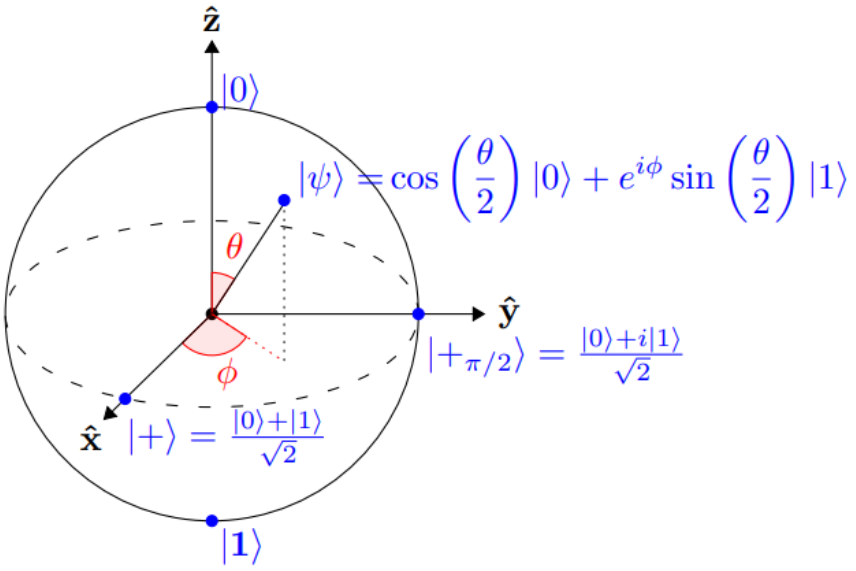


Figure 2.3: Bloch sphere representation. The sphere's north and south poles correspond to states $|0\rangle$ and $|1\rangle$, respectively. Reproduced from [27].

The aforementioned rotation gates can be written as: $R_x(\theta) = e^{-i\theta\sigma_x/2}$, $R_y(\theta) = e^{-i\theta\sigma_y/2}$, and $R_z(\theta) = e^{-i\theta\sigma_z/2}$. In matrix form, this reads:

$$R_x(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \quad R_y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \quad R_z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}, \tag{2.13}$$

These will frequently appear in quantum circuits, as they are the most basic quantum gates, in the sense that all other single-qubit gates can be re-constructed from them. I.e., for any unitary single-qubit gate U, one can always find a decomposition $U = e^{i\phi}R_Z(\gamma)R_X(\beta)R_Z(\alpha)$. (Proven in [28].)

For illustrative purposes, to elucidate how single-qubit gates act on qubits, it is quite simple to verify that the σ_x matrix (alternative symbol, $\boldsymbol{X}$) behaves exactly like a NOT gate. This correspondence is established through a $180°$ rotation around the $x-$axis. Mathematically,

$$\boldsymbol{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{2.14}$$

such that:

$$\boldsymbol{X}|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle\,. \tag{2.15}$$

Another essential gate found in most quantum algorithms is the Hadamard gate. It can be represented as the following linear combination of $\boldsymbol{X}$ and $\boldsymbol{Z}$ matrices: $H = \frac{1}{\sqrt{2}}(\boldsymbol{X} + \boldsymbol{Z})$, and is particularly useful for

creating superpositions, as it maps the state $|0\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) := |+\rangle$, and $|1\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) := |-\rangle$.

In addition to these simple one-qubit quantum gates, there are also gates designed for multiple qubits. For instance, consider the controlled-NOT gate (CNOT), which does exactly what its name suggests: if the control qubit is in state $|1\rangle$, it applies NOT to the test qubit, whereas if it is in state $|0\rangle$ instead, nothing happens. The CNOT gate is also quite prevalent in quantum circuits, playing a vital role in creating entanglement between qubits, which is a crucial resource in quantum computing. The circuit representation and equivalent matrix form of the CNOT gate are presented below, in Figure 2.6.

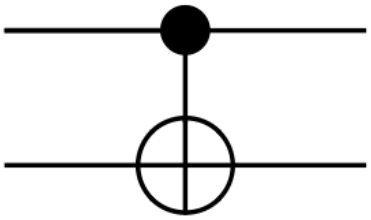


Figure 2.4: CNOT gate – Circuit representation. The black circle identifies the control qubit. The other qubit is the test qubit.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.5: CNOT gate – Equivalent matrix form.

Figure 2.6: Two-qubit gate example – The controlled-NOT gate.

More generally, within a quantum circuit, an intricate network of interconnected qubits and gates collaborates to execute a specific algorithm. For example, the circuit depicted in Figure 2.7, below, is designed for the implementation of the renowned Grover's search algorithm [29].

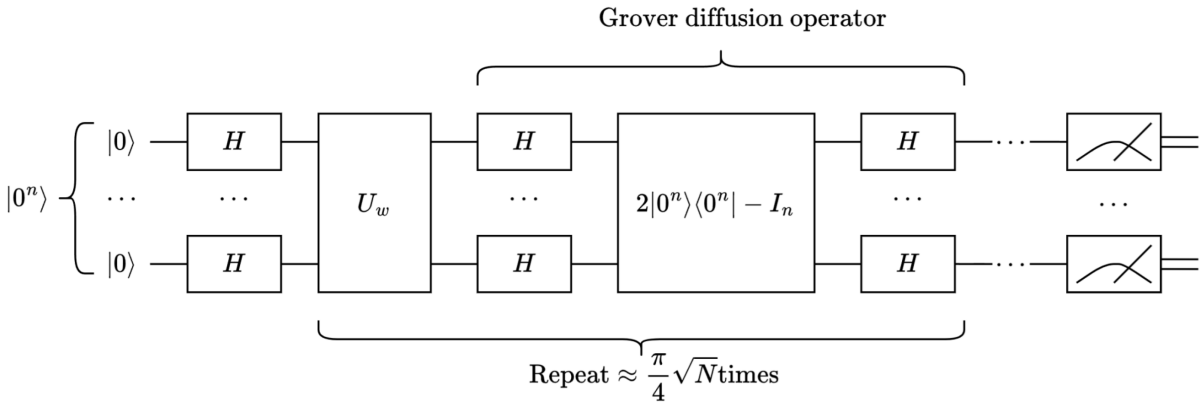


Figure 2.7: Quantum circuit utilized in the famous Grover's search algorithm [29]. Integraly reproduced from [30].

This is a purely quantum algorithm, different from what we intend to explore in this work. Nevertheless, it serves as a good example of how quantum circuits can be used to implement complex quantum algorithms.

An additional point of interest is the potential for classical simulation of quantum circuits. Quantum gates, being akin to mathematical operators, can always be expressed in matrix form. Consequently, analyzing the impact of a series of quantum gates, i.e., a quantum circuit, on a specific quantum state is achievable by sequentially applying the relevant operators, as they appear in the circuit. At the end of this

process, we arrive at a quantum state typically distinct from the initial state. This entire simulation can be executed classically. However, its scalability is severely limited, as the number of states, constituting our computational basis, representable with n qubits grows exponentially, at a rate of 2^n. As one might anticipate, this very quickly becomes unmanageable. Eventually, either memory constraints impede the storage of such extensive states, or the computations slow down to a point where the endeavor loses practicality. Effectively, simulating quantum computers becomes challenging beyond a few dozen qubits. Some tabletop calculations suggest that simulating 28 qubits would necessitate approximately 13 GB of memory, reaching the theoretical maximum capacity of a modern standard 16 GB personal computer. It's important to note that for each additional qubit, memory requirements double. For instance, simulating 50 qubits would demand about 54 Petabytes of memory, and for 51 qubits, approximately 108 PB, and so forth. These estimations are based on the assumption that a complex number is stored as two Python floats, each requiring 24 bytes of memory, thus a complex amplitude requires 48 bytes of memory to be stored. Then, one such amplitude is required for each of the 2^n basis states, for n qubits.

2.3 Hybrid Quantum-Classical Computing

Hybrid quantum-classical computing refers to a computational approach that combines elements of both classical and quantum computing paradigms to leverage the strengths of each. In this model, classical processors and quantum processors work in tandem to solve complex problems more efficiently than either could achieve alone. This collaborative strategy aims to harness quantum computing's unique capabilities while mitigating the challenges and limitations associated with quantum systems, such as error correction and decoherence. As of today, the synergy between classical and quantum elements holds promise for addressing complex real-world problems in areas like optimization, machine learning, and cryptography.

2.3.1 Variational Quantum Algorithms

The hallmark of HQCC is what are called variational quantum algorithms (VQAs), which correspond to hybrid-quantum algorithms, typically realized through a parameterized quantum circuit. Additionally, as part of their implementation, the parameters are subjected to training, in order to achieve the desired outcomes (By minimizing the value of a cost function). As such, VQAs can be thought of as the quantum analogue of highly successful machine-learning methods, such as neural networks. Moreover, since they outsource the circuit parameters' optimization to a classical optimizer, exterior to the quantum processor, VQAs leverage the full toolbox of classical optimization. As a whole, this approach has the added advantage of keeping the quantum circuit depth shallow, which, ultimately, helps in mitigating the overall noise level. This is crucial, since it allows for these algorithms to be implemented in the current NISQ (Noisy Intermediate Scale Quantum) era, not requiring complete fault-tolerance to produce reasonable results. A wide array of possible applications has been examined for VQAs, essentially covering all the use cases envisioned by researchers for quantum computers (See Figure 2.8). Despite all this, it is

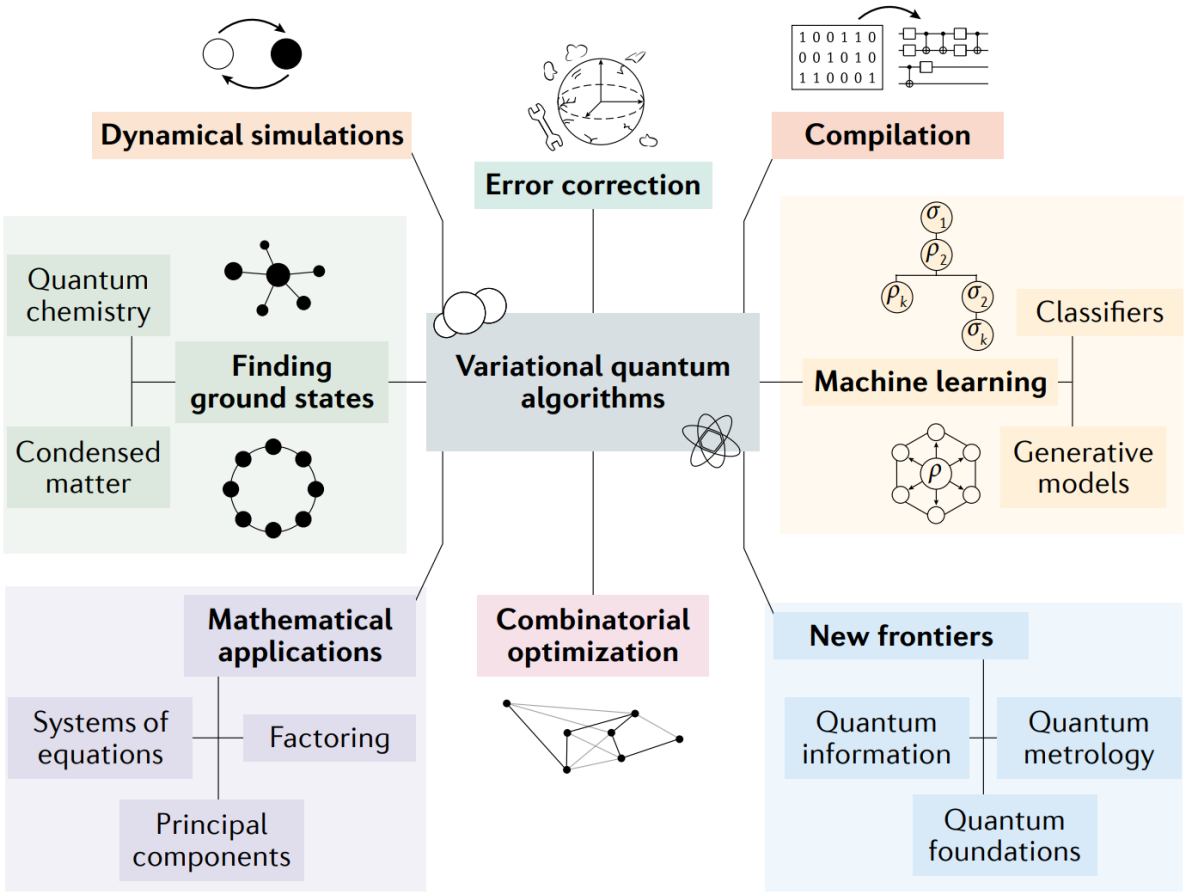


Figure 2.8: Applications of variational quantum algorithms. Sourced from Ref. [31], in its entirety.

important to note that VQAs are not without fault. There is still a lot of work to be done regarding their trainability, accuracy and efficiency. Either way, they are undoubtedly the most promising candidates for achieving useful quantum advantage in the near future, which is exactly why they have come under the spotlight, drawing the attention of numerous scientists over the past few years.

2.3.1.1 Quantum Approximate Optimization Algorithm (QAOA) Review

Describe QAOA - Explain how it works, its advantages and disadvantages, and its potential applications.

There's a mistake in eq. (5), in the PIC2 report, that I should correct, once I include it here. (It's the coefficients γ and β!)

I should also change the labels of the problem and mixer Hamiltonians, so they always match the figures. (Sometimes, it's H_P and H_M, sometimes it's H_C and H_B.)

Also, mention that $|\psi_0\rangle$ is the uniform superposition state, and that it's the initial state entering the ansatz.

Specify that $U_{H_{P_l}} = \exp\{-iH_P\gamma_l\}$. Or something like that.

I also feel like I don't mention that we use a number of qubits equal to the number of nodes. I should do that. And, then, explain how to interpret the results. (0 - One partition; 1 - The other partition.)

I should present an explicit (with the actual CNOTs) QAOA ansatz, maybe for the usual 8-node graph. Or, for the smaller, trivial 4-node graph. This elucidates the idea of the ansatz. This would also require me to explain how $\exp\{-i\gamma Z_i Z_j\}$ is transpiled/represented as a quantum circuit. (Look at Bence's TeX file: there's an example there, for $\exp\{-i\gamma Z_i Z_j Z_k Z_l\}$, I believe.)

2.3.1.2 Variational Qubit-Efficient MaxCut Heuristic Algorithm (QEMC) Review

Describe QEMC - Explain how it works, its advantages and disadvantages, and its potential applications.

The reason why we can efficiently iterate through the values of B is that we have a reduced number of qubits (exponential compression). [I think so, at least.] We always say the set with B blue nodes is the smallest. Aka., $B \leq N/2$, where N is the graph's number of nodes.

Is there any reason for this, though? What would happen if we said $B > N/2$? (Aka., the blue set is the one with the most nodes.)

Mention that the problem-agnostic "Strongly-Entangling-Layers" ansatz might result in "too much entanglement", thus hindering the results.

Go over the last paragraph of the QEMC section again. Something about the discussion on "shot number" is bugging me.

Chapter 3

Base iQAQE Algorithm

In this chapter, we shall describe the base hybrid quantum-classical algorithm that we're proposing in this work. This algorithm is based on the Quantum Approximate Optimization Algorithm (QAOA) and the Qubit-Efficient MaxCut Heuristic (QEMC) algorithm. It is called: Interpolated QAOA/QEMC Hybrid Algorithm (iQAQE).

3.1 Interpolated QAOA/QEMC Hybrid Algorithm (iQAQE)

Describe iQAQE - Explain how it works, its advantages and disadvantages, and its potential applications. Also, mention how iQAQE has many degrees of freedom that can be tunes, which led to the development of many variations of iQAQE, springing from the original idea. These will be described in the iQAQE Schemes and Results chapter.

In the PIC2 report, I sort of contradict myself in the end when I mention that "[...] it is somewhat inefficient to iterate over the possible B values [...]". I should correct this (just remove this sentence).

Sub-lists' cardinalities interval: $[1, 2^N - 1]$. In the PIC2 report, it states $[1, 2^{N-1}]$, which is, indeed, what I used for the simulations. However, there's no reason why we shouldn't be allowed to consider $2^N - 1$ maximum number of basis states per list. I should put this disclaimer somewhere in here (Thesis).

In the "Motivation behind this work" section in the PIC2 report, there's a number of things that need to be re-written, before being included here! Remember to do this! (Inefficient B iterations, better trainability - fewer parameters?, etc.)

There's also a bit of redundancy in this chapter.

Chapter 4

Implementation details

Insert your chapter material here. - In this chapter, we should describe numerical aspects of the algorithms' implementations. This goes for all three of QAOA, QEMC and iQAQE. Mention Pennylane, the developed code/models, the GitHub repository, and any other relevant information. Mention, also, that we're always doing numerical simulations of quantum systems, on a classical computer!

4.1 Individual Algorithms

Description of the numerical implementation of the models explained in Chapter 2.

If needed, pseudo-codes can be included as exemplified in Algorithm 1.

Algorithm 1 Euclid's algorithm

```
1: procedure EUCLID(a, b)                          ▷ The g.c.d. of a and b
2:     r ← a mod b
3:     while r ≠ 0 do                              ▷ We have the answer if r is 0
4:         a ← b
5:         b ← r
6:         r ← a mod b
7:     end while
8:     return b                                    ▷ The gcd is b
9: end procedure
```

4.1.1 Quantum Approximate Optimization Algorithm (QAOA)

Description of the numerical implementation of the QAOA model.

4.1.2 Qubit-Efficient MaxCut Heuristic (QEMC)

Description of the numerical implementation of the QEMC model.

4.1.3 Interpolated QAOA/QEMC Hybrid Algorithm (iQAQE)

Description of the numerical implementation of the iQAQE model.

4.2 Benchmarking and Testing methods

How do we benchmark our models? This should be described here: mention the Avg. BSF metric, and how it was ”wrong”, initially, and how it was ”fixed”. Also mention any other possible metrics that could be used to compare the models: Grid-searches, etc.

Basic test cases to compare the implemented model against other numerical tools (verification) and experimental data (validation).

I should also introduce the utilized score metrics: Best-so-far average and median, etc. Maybe, mention the difference between the before and after of the ”BSF Correction”.

Chapter 5

iQAQE Schemes and Results

Insert your chapter material here. - In this chapter, we should present all the many schemes that we've come up with, and the results of the numerical simulations of these schemes. We should also compare the results of the different schemes, and discuss the potential of each one. We should also mention the importance of the results and the potential applications of the schemes. This will, surely, be the largest chapter of the thesis.

Here, I should present each realization of iQAQE, theoretically, as well as the results of the numerical simulations of each realization. I should also mention that this work is, essentially, a collection/compilation of different schemes/ideas, all inspired by the base-iQAQE algorithm, described above. As such, some schemes will appear disconnected from others, but they all serve the same purpose, ultimately. (This is how I should justify the existence of so many schemes, and the lack of a clear connection between them. Also, this is how one should look at this.)

I might re-order these a little.

5.1 Base iQAQE (Random)

Base iQAQE schemes and their results.

5.2 Polynomial Compression-type Encodings

Polynomial Compression-type Encodings and their results.

5.2.1 Basic Polynomial Compression-type iQAQE

Basic Polynomial Compression-type iQAQE schemes and their results.

5.2.2 Parity-like QAOA

Parity-like QAOA schemes and their results.

5.2.3 Correlation-based iQAQE

Correlation-based iQAQE schemes and their results.

5.2.4 Fixed-Parity iQAQE

Fixed-Parity iQAQE schemes and their results.

5.3 Parity-like QAOA

Parity-like QAOA schemes and their results.

5.4 Other Exploratory Ideas

Other exploratory ideas and their results.

5.4.1 Tranche-based Oracle colouring

Tranche-based Oracle colouring schemes and their results.

5.5 Extended-QEMC

Extended-QEMC scheme and variations thereof, and their results.

5.5.1 Vanilla Extended-QEMC

Vanilla Extended-QEMC scheme and variations thereof, and their results.

5.5.2 Cardinality $= k$ Extended-QEMC

Cardinality $= k$ Extended-QEMC scheme and variations thereof, and their results.

5.6 Goemans-Williamson and Bigger Graphs

Goemans-Williamson scheme and its results on bigger graphs.

5.7 Alternative ansatzë

Alternative ansatzë and their results: More specifically, Non-deterministic CNOTs. Could also feature some discussion on problem-inspired *vs.* problem-agnostic ansatzë. Mention how we've been

using problem-agnostic ansatzë, in iQAQE, and how this hinders the results. Explain how/why problem-inspired is better, and how we could use it in the future. (Could refer to the QML review paper, here.)

5.8 Average Best-so-Far correction

Average Best-so-Far correction and its results. Mention how it was ”wrong”, initially, and how it was ”fixed”. I’m not sure where to include this, though.

5.9 Randomized iQAQE benchmarking

Talk about the randomized benchmarking that we’ve been doing, and how it’s been helping us to understand the performance of the different schemes. Type 1 and 2 variables, etc. This could have many subsections.

5.9.1 Figures

Insert your section material and possibly a few figures.

Make sure all figures presented are referenced in the text!

The caption should appear below the figure.

5.9.1.1 Images

By default, this document supports file types *.png,.pdf,.jpg,,.jpeg*.

See the documentation of package *graphicx* https://www.ctan.org/tex-archive/macros/latex/required/graphics/ for other extensions support.

When referencing a figure, use the abbreviation Fig., unless it is the beginning of a sentence.

Figure 5.1 is an example and so is Fig. 5.2.

Figure 5.1: Caption for figure.

It is possible to include subfigures. Figure 5.2 is composed of three subfigures: Fig. 5.2a, 5.2b and 5.2c.

Most aircraft have wings with large aspect ratios ($\mathcal{AR}$= 8 – 15) for higher aerodynamic efficiency.

]fig1a

(a) Airbus A320.

(b) Bombardier CRJ200.

(c) Airbus A350.

Figure 5.2: Examples of aircraft.

5.9.1.2 Drawings

Insert your subsection material and for instance a few drawings.

The schematic illustrated in Fig. 5.3 can represent some sort of algorithm.

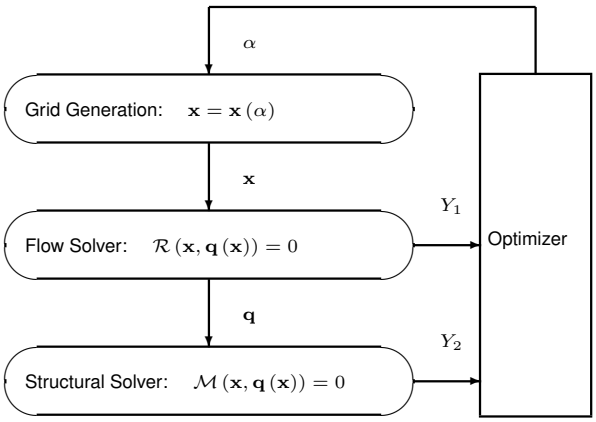


Figure 5.3: Schematic of some algorithm.

5.9.2 Equations

Equations can be inserted in different ways.

The simplest way is in a separate line as

$$\frac{\mathrm{d}q_{ijk}}{\mathrm{d}t} + \mathcal{R}_{ijk}(\boldsymbol{q}) = 0\,, \tag{5.1}$$

where each variable must properly defined.

If the equation is to be embedded in the text, it can be done like $\partial\mathcal{R}/\partial\boldsymbol{q} = 0$.

It may also be split in different lines like

$$\begin{aligned} \text{Minimize} \quad & Y(\boldsymbol{\alpha}, \boldsymbol{q}(\boldsymbol{\alpha})) \\ \text{with respect to} \quad & \boldsymbol{\alpha} \\ \text{subject to} \quad & \mathcal{R}(\boldsymbol{\alpha}, \boldsymbol{q}(\boldsymbol{\alpha})) = 0 \\ & C(\boldsymbol{\alpha}, \boldsymbol{q}(\boldsymbol{\alpha})) = 0\,. \end{aligned} \tag{5.2}$$

It is also possible to use subequations.

$$\frac{\partial\rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j) = 0\,, \tag{5.3a}$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j + p\delta_{ij} - \tau_{ji}) = 0, \quad i = 1, 2, 3\,, \tag{5.3b}$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}(\rho E u_j + p u_j - u_i\tau_{ij} + q_j) = 0\,. \tag{5.3c}$$

Notice that the equations should be punctuated as they are part of sentences, so a comma or a period should be put at the end of each of them, as exemplified in all the previous equations.

When referencing an equation, use the abbreviation Eq., unless it is the beginning of a sentence. The number of the equation should always be in parenthesis.

Equations (5.3a), (5.3b) and (5.3c) form the Navier–Stokes equations (Eq. (5.3)).

5.9.3 Tables

Insert your subsection material and for instance a few tables.

Make sure all tables presented are referenced in the text!

The caption should appear above the table.

Follow some guidelines when making tables:

- Avoid vertical lines;
- Avoid “boxing up” cells, usually 3 horizontal lines are enough: above, below, and after heading;
- Avoid double horizontal lines;
- Add enough space between rows.

Table 5.1: Table caption.

Model	C_L	C_D	C_{My}
Euler	0.083	0.021	-0.110
Navier–Stokes	0.078	0.023	-0.101

When referencing a table, use the abbreviation Tab., unless it is the beginning of a sentence.

Tables 5.2 and 5.3 are examples of tables with merging columns:

Table 5.2: Memory usage comparison (in MB).

	Virtual memory [MB]	
	Euler	Navier–Stokes
Wing only	1,000	2,000
Aircraft	5,000	10,000
(ratio)	$5.0\times$	$5.0\times$

An example with merging rows can be seen in Tab. 5.4.

If a table has too many columns, it can be scaled to fit the text width, as in Tab. 5.5.

5.9.4 Mixing

If necessary, a figure and a table can be put side-by-side as in Fig. 5.4

Table 5.3: Another table caption.

	$w = 2$			$w = 4$		
	$t = 0$	$t = 1$	$t = 2$	$t = 0$	$t = 1$	$t = 2$
$dir = 1$						
c	0.07	0.16	0.29	0.36	0.71	3.18
c	-0.86	50.04	5.93	-9.07	29.09	46.21
c	14.27	-50.96	-14.27	12.22	-63.54	-381.09
$dir = 0$						
c	0.03	1.24	0.21	0.35	-0.27	2.14
c	-17.90	-37.11	8.85	-30.73	-9.59	-3.00
c	105.55	23.11	-94.73	100.24	41.27	-25.73

Table 5.4: Yet another table caption.

ABC	header			
	1.1	2.2	3.3	4.4
IJK	group		0.5	0.6
			0.7	1.2

Table 5.5: Very wide table.

Variable	a	b	c	d	e	f	g	h	i	j
Test 1	10,000	20,000	30,000	40,000	50,000	60,000	70,000	80,000	90,000	100,000
Test 2	20,000	40,000	60,000	80,000	100,000	120,000	140,000	160,000	180,000	200,000

Legend		
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Figure 5.4: Figure and table side-by-side.

Chapter 6

Conclusions

Insert your chapter material here.

6.1 Achievements

The major achievements of the present work.

6.2 Future Work

A few ideas for future work.

Bibliography

[1] J. Preskill. Quantum computing 40 years later, 2023.

[2] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm, 2014.

[3] Y. Tene-Cohen, T. Kelman, O. Lev, and A. Makmal. A variational qubit-efficient maxcut heuristic algorithm, 2023.

[4] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 105–112 vol.1, 2001. doi: 10.1109/ICCV.2001.937505.

[5] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988. doi: 10.1287/opre.36.3.493.

[6] J. Poland and T. Zeugmann. Clustering pairwise distances with missing data: Maximum cuts versus normalized cuts. In L. Todorovski, N. Lavrač, and K. P. Jantke, editors, *Discovery Science*, pages 197–208, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46493-8.

[7] R. M. Karp. *Reducibility Among Combinatorial Problems*, pages 219–241. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-540-68279-0. doi: 10.1007/978-3-540-68279-0_8. URL https://doi.org/10.1007/978-3-540-68279-0_8.

[8] A. Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2, 2014. ISSN 2296-424X. doi: 10.3389/fphy.2014.00005. URL https://www.frontiersin.org/articles/10.3389/fphy.2014.00005.

[9] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, nov 1995. ISSN 0004-5411. doi: 10.1145/227683.227684. URL https://doi.org/10.1145/227683.227684.

[10] R. Mirka and D. P. Williamson. An experimental evaluation of semidefinite programming and spectral algorithms for max cut. *ACM J. Exp. Algorithmics*, 28, aug 2023. ISSN 1084-6654. doi: 10.1145/3609426. URL https://doi.org/10.1145/3609426.

[11] J. A. Soto. Improved analysis of a max-cut algorithm based on spectral partitioning. *SIAM Journal on Discrete Mathematics*, 29(1):259–268, 2015. doi: 10.1137/14099098X. URL https://doi.org/10.1137/14099098X.

[12] S. Sen. Simulated annealing approach to the max cut problem. In U. M. Fayyad and R. Uthurusamy, editors, *Applications of Artificial Intelligence 1993: Knowledge-Based Systems in Aerospace and Industry*, volume 1963 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 61–66, Mar. 1993. doi: 10.1117/12.141755.

[13] Y.-H. Kim, Y. Yoon, and Z. W. Geem. A comparison study of harmony search and genetic algorithm for the max-cut problem. *Swarm and evolutionary computation*, 44:130–135, 2019.

[14] G. G. Guerreschi and A. Y. Matsuura. Qaoa for max-cut requires hundreds of qubits for quantum speed-up. *Scientific Reports*, 9(1):6903, May 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-43176-9. URL https://doi.org/10.1038/s41598-019-43176-9.

[15] Z. Zhou, Y. Du, X. Tian, and D. Tao. Qaoa-in-qaoa: solving large-scale maxcut problems on small quantum machines, 2022.

[16] K. Ender, A. Messinger, M. Fellner, C. Dlaska, and W. Lechner. Modular parity quantum approximate optimization. *PRX Quantum*, 3(3):030304, 2022.

[17] K. Ender, R. ter Hoeven, B. E. Niehoff, M. Drieb-Schön, and W. Lechner. Parity Quantum Optimization: Compiler. *Quantum*, 7:950, Mar. 2023. ISSN 2521-327X. doi: 10.22331/q-2023-03-17-950. URL https://doi.org/10.22331/q-2023-03-17-950.

[18] M. Sciorilli, L. Borges, T. L. Patti, D. García-Martín, G. Camilo, A. Anandkumar, and L. Aolita. Towards large-scale quantum optimization solvers with few qubits, 2024.

[19] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. ISBN 9781139495486. URL https://books.google.pt/books?id=-s4DEy7o-a0C.

[20] S. Slussarenko and G. J. Pryde. Photonic quantum information processing: A concise review. *Applied Physics Reviews*, 6(4), 2019.

[21] L. Madsen, F. Laudenbach, M. Askarani, F. Rortais, T. Vincent, J. Bulmer, F. Miatto, L. Neuhaus, L. Helt, M. Collins, A. Lita, T. Gerrits, S. Nam, V. Vaidya, M. Menotti, I. Dhand, Z. Vernon, N. Quesada, and J. Lavoie. Quantum computational advantage with a programmable photonic processor. *Nature*, 606:75–81, 06 2022. doi: 10.1038/s41586-022-04725-x.

[22] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, Sept. 2020. ISSN 2521-327X. doi: 10.22331/q-2020-09-21-327. URL https://doi.org/10.22331/q-2020-09-21-327.

[23] X. Wu, X. Liang, Y. Tian, F. Yang, C. Chen, Y.-C. Liu, M. K. Tey, and L. You. A concise review of rydberg atom based quantum computation and quantum simulation*. *Chinese Physics B*, 30(2): 020305, Feb. 2021. ISSN 1674-1056. doi: 10.1088/1674-1056/abd76f. URL http://dx.doi.org/10.1088/1674-1056/abd76f.

[24] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage. Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, 6(2), 2019.

[25] H.-L. Huang, D. Wu, D. Fan, and X. Zhu. Superconducting quantum computing: a review. *Science China Information Sciences*, 63(8), July 2020. ISSN 1869-1919. doi: 10.1007/s11432-020-2881-9. URL http://dx.doi.org/10.1007/s11432-020-2881-9.

[26] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11:369–395, 2020. ISSN 1947-5462. doi: https://doi.org/10.1146/annurev-conmatphys-031119-050605. URL https://www.annualreviews.org/content/journals/10.1146/annurev-conmatphys-031119-050605.

[27] A. S. Cacciapuoti, M. Caleffi, R. Van Meter, and L. Hanzo. When entanglement meets classical communications: Quantum teleportation for the quantum internet. *IEEE Transactions on Communications*, 68(6):3808–3833, June 2020. ISSN 1558-0857. doi: 10.1109/tcomm.2020.2978071. URL http://dx.doi.org/10.1109/TCOMM.2020.2978071.

[28] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5): 3457–3467, Nov. 1995. ISSN 1094-1622. doi: 10.1103/physreva.52.3457. URL http://dx.doi.org/10.1103/PhysRevA.52.3457.

[29] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. doi: 10.1145/237814.237866. URL https://doi.org/10.1145/237814.237866.

[30] H. Wu, X. Feng, and J. Zhang. Quantum implementation of the sand algorithm and its quantum resource estimation for brute-force attack. *Entropy*, 26(3), 2024. ISSN 1099-4300. doi: 10.3390/e26030216. URL https://www.mdpi.com/1099-4300/26/3/216.

[31] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9): 625–644, Aug. 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00348-9. URL http://dx.doi.org/10.1038/s42254-021-00348-9.

Appendix A

Vector calculus

In case an appendix if deemed necessary, the whole document cannot exceed a total of 100 pages (in arabic page numbering).

Some definitions and vector identities are listed in the section below.

A.1 Vector identities

$$\nabla \times (\nabla \phi) = 0 \tag{A.1}$$

$$\nabla \cdot (\nabla \times \mathbf{u}) = 0 \tag{A.2}$$

Appendix B

Technical Datasheets

It is possible to add PDF files to the document, such as technical sheets of some equipment used in the work.

B.1 Some Datasheet

See more options to include PDF files in https://www.ctan.org/pkg/pdfpages

Lightweight scanning lidar

Features

- *Application:* *Obstacle detection and navigation for small autonomous vehicles and drones*
- *Key features:* *Small and lightweight*
  *Upgradable through the **LightWare Studio** application*
- *Measuring range:* *0.2 ... 50 m (80% reflective, large target)*
- *Size:* *53 mm x 44 mm x 37 mm*
- *Weight:* *48.3 grams*
- *Measuring speed:* *Up to 20,000 points per second (configurable)*
- *Interfaces:* *Serial, I2C and USB*
- *Integration:* *User APIs, **LightWare Studio***
- *Safety:* *Eye safe laser emission Class 1M*
- *Environmental:* *Open frame, no IP rating*

bsi. ISO 9001:2015 Quality Management
FM 654831

Table of contents

Product ordering codes

Model family	Model name	Model description
SF45	SF45/B (50 m)	Open frame scanning lidar sensor, max 50 m

Disclaimer

Information found in this document is used entirely at the reader's own risk and whilst every effort has been made to ensure its validity, neither LightWare Optoelectronics (Pty) Ltd nor its representatives make any warranties with respect to the accuracy of the information contained herein.

1. Overview

The SF45/B is a small, lightweight scanning lidar ideal for obstacle detection by small autonomous vehicles. The horizontal field of view can be adjusted from a few degrees up to 320 degrees to suit the application. Objects up to 50m away can be detected and avoided by finding clear pathways using simple navigation commands. The SF45/B is tolerant to changes in background lighting conditions, wind and noise.

The following capabilities are included in the SF45/B as standard:

- Streaming of live readings.
- Alarms when an obstacle is detected.
- Configurable update rate and scanning angle.
- Internal status monitoring.

Additional features may be added through ***LightWare Studio***:

- Servo driver for a second axis of motion.
- Measurement to the nearest detected surface (first return).
- Measurement to the farthest detected surface (last return).
- Selectable filters to adjust the dynamic response to moving targets.
- Navigation tools.
- Custom features.

The following communication interfaces are available:

- A micro USB port that connects to a PC running the ***LightWare Studio*** application for visualisation of results, to make configuration changes and for upgrading the firmware.
- A serial port (3.3V logic level) with configurable baud rate to connect to a host controller.
- An I2C serial bus (3.3V logic level, external pull up resistors required) with configurable address as an alternative to the serial port when multiple devices are connected on a common bus.
- Two general purpose outputs.

Application software support is available from the LightWare ***API*** repository.

The SF45/B scanning lidar is rated laser Class 1M eye safe. Do not view the laser with magnifying optics such as microscopes, binoculars or telescopes.

2. Specifications

Performance	
Range	0.2 ... 50 m (white wall in daylight conditions)
Linear Resolution	1 cm
Angular Resolution	< 0.2 deg
Update rate	Up to 20,000 readings per second and 5 sweeps per second.
Accuracy	±10 cm
Connections	
Power supply voltage	4.5 V ... 5.5 V
Power supply current	300 mA (typical)
Outputs & interfaces	Serial and I2C (3.3 V), micro USB, general purpose outputs
Mechanical	
Dimensions	53 mm x 44 mm x 37 mm
Weight	48g (excluding cables)
Optical	
Laser safety	Class 1M (refer to www.lightware.co.za/safety for full details)
Optical aperture	28 mm x 15 mm
Beam divergence	< 0.5°
Environmental	
Operating temperature	-10 ... +50°C
Approvals	FDA: 1710193-000 (2019/08)
Enclosure rating	N/A
Accessories	
Main cable	7 way - individual wires, unterminated
USB cable	USB cable - DigiKey AE10418-ND
Default settings	
Serial port settings	115200 baud, 8 data bits, 1 stop bit, no parity, no handshaking
I2C address	0x66 (Hex), 102 (Dec)
Update rate	388 readings per second
Main cable connections	
1	GPIO / LED driver
2	GPIO / servo driver
3	TXD/SDA - serial data transmit or I2C data
4	RXD/SCL - serial data receive or I2C clock
5	GND - power supply negative
6	GND - power supply negative
7	+ 5 V - power supply positive (4.5 V to 5.5 V at 500 mA)