

Blind Oracular Quantum Computation: from Concept to Physical Implementation

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades einer
Doktorin der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Cica Gustiani, M.Sc.
aus
Bandung, Indonesien

Berichter: Prof. Dr. David P. DiVincenzo
Prof. Dr. Fabian Hassler

Tag der mündlichen Prüfung: 26 November 2020

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek verfügbar.

ABSTRACT

Recent years have seen much excitement for application of quantum computing, triggered by substantial — and ongoing — advances in experimental realizations of quantum computing. It is widely believed that client-server is the setting for quantum computers that will prevail in the future, where privacy becomes crucial in the application. Moreover, a recent experiment by Barz et al. [1] successfully demonstrated a blind quantum computation scheme: a client-server quantum computation in which a client with limited quantum power controls the execution of a quantum computation on a powerful server, without revealing valuable details of the computation.

In this thesis, we discuss the Blind Oracular Quantum Computation (BOQC) scheme, a blind quantum computing scheme in which a third party (the oracle) with limited quantum power, assists the execution of a client’s oracular quantum computations by implementing oracle evaluations. In BOQC, a client with limited quantum power and without the capacity to construct the oracle, can delegate her oracular quantum computations to a powerful yet untrustworthy server, with the help of the oracle. We show that BOQC is provably *blind* within a *composable* definition such that the server cannot learn about the clients’ computation. We provide a realization of BOQC in a physical setting, particularly in a diamond nitrogen-vacancy (NV) center platform. In BOQC, the server has a One-Way Quantum Computer (1WQC) that is resource-demanding. To lower the resource-requirements, we develop the BOQCo protocol, a BOQC that employs *lazy* 1WQC to minimize the number of qubits needed. We also provide systematic numerical optimization to find resource states that are *BOQC-compatible* by admitting BOQC security criteria. Finally, we give explicit oracular quantum algorithms that are BOQC-compatible to be executed on the NV-center platform. The algorithms include 2-qubit Grover’s algorithm using three qubits, 3-qubit exact Grover’s algorithm using four qubits, 2-qubit Simon’s algorithm with a *useless* oracle using four qubits, and Deutsch’s algorithm using three physical qubits. We hope that these BOQC algorithms intrigue some experimentalists to try to implement them.

ZUSAMMENFASSUNG

Durch erhebliche — und anhaltende — Fortschritte bei den experimentellen Realisierungen des Quantencomputers in den vergangenen Jahren haben potentielle Anwendungen eines Quantencomputers zunehmend an Interesse gewonnen. Es wird allgemein angenommen, dass das *Client-Server-Modell* den Rahmen für Quantencomputer bildet, der sich in Zukunft durchsetzen wird, wobei die Garantie des Datenschutzes in der Implementation von entscheidender Bedeutung sein wird. Darüber hinaus hat ein kürzlich durchgeführtes Experiment von Barz et al. [1] erfolgreich ein blindes Quantenberechnungsschema demonstriert: eine Client-Server-Quantenberechnung, bei der ein Client mit begrenzter Quantenleistung die Ausführung einer Quantenberechnung auf einem leistungsstarken Server steuert, ohne wertvolle Details der Berechnung preiszugeben.

In dieser Arbeit diskutieren wir das Blind Oracular Quantum Computation (BOQC) Schema, ein Blind Quantum Computing-Schema, bei dem eine dritte Partei (das Orakel) mit begrenzter Quantenleistung die Ausführung der orakulären Quantenberechnungen eines Kunden durch die Implementierung des Orakels unterstützt. In BOQC kann ein Client mit begrenzter Quantenleistung und ohne die Fähigkeit, das Orakel selbst zu implementieren, seine Orakel-quantenberechnungen an einen leistungsstarken, aber nicht vertrauenswürdigen Server delegieren. Wir zeigen, dass BOQC innerhalb einer *komponierbaren* Definition beweisbar *blind* ist, sodass der Server nichts über die Berechnung der Klienten erfahren kann. Wir geben hier eine Realisierung von BOQC in einem physikalischen System, insbesondere in einer NV-Plattform (Diamond Nitrogen Vacancy). In BOQC verfügt der Server über einen Einweg-Quantencomputer (1WQC), der eine Großzahl an Ressourcen beansprucht. Um den Ressourcenbedarf zu senken, entwickeln wir das BOQCo-Protokoll, ein BOQC, welches *Lazy 1WQC* verwendet. Wir entwickeln auch ein systematisches numerisches Optimierungsverfahren, um Ressourcenzustände zu finden, die BOQC-kompatibel sind und die BOQC-Sicherheitskriterien erfüllen. Schlussendlich geben wir explizite orakuläre Quantenalgorithmen, die BOQC-kompatibel sind und auf der NV-Center-Plattform implementiert werden können. Diese Algorithmen umfassen den 2-Qubit-Grover-Algorithmus mit drei Qubits, den 3-Qubit-exakten Grover-Algorithmus mit vier Qubits, den 2-Qubit-Simon-Algorithmus mit einem *useless* Orakel mit vier Qubits und den Deutsch-Algorithmus mit drei physischen Qubits. Wir hoffen, dass diese BOQC-Algorithmen einige Experimentatoren dazu veranlassen, zu versuchen sie zu implementieren.

ACKNOWLEDGEMENT

It is impossible to complete this thesis without all the help and support I received from the people surrounding me. I am incredibly grateful for that.

First and foremost, I would like to express my very great appreciation to David DiVincenzo for your guidance in this four valuable years of journeying through quantum information, for giving me a chance to join your group, and for believing in me. I deeply admire your original ideas, your amazingly broad knowledge, sharp-wittedness, efficiency, and kindness. Thank you for always finding the time for discussions, remembering every detail of my research progress, and helping me in bureaucratic matters. It has been a lot of joy working with you; our discussions have always been stimulating and greatly helpful. Also, thank you so much for proofreading — pretty much all of my writing, giving me valuable and constructive feedback in my writings. Your lessons will always stay with me for the rest of my life.

Secondly, I would like to thank Fabian Hassler for agreeing to be my second reviewer. Also, thank you for making an English tutorial class for me in Theo-0, it certainly helped a lot.

Thirdly, I would like to express my gratitude to Tal Mor and Rotem Liss, for the fruitful discussions on QKD, security, and valuable feedback on the BOQC protocol. I would also like to thank Christopher Portmann for helping me understand Abstract Cryptography, proofreading on BOQC security proofs, and the valuable feedback on the security parts. I want to thank Elham Kashefi for a fruitful discussion on UBQC. I want to offer my special thanks to Tim Tamiñiau and Slava Dobrovitski, for productive and helpful discussions on experimental NV-centers. I want to thank Barbara Terhal for the interactive proof discussion and for pointing out the right people to discuss NV-centers.

Fourthly, I would like to thank every past and present member of IQI, for making the office feel like my second home. Special thanks to Helene Barton, thank you for helping me in bureaucratic matters, making my life easier. Special thanks to Joel Pommerening; it has always been pleasant to share an office with you. It was a lot of fun managing the quantum information exercise class with you, especially since you are an amazingly smart and quick-thinking person. A special thanks to

Daniel Zeuch, thank you for being an awesome office mate and friend, writing the reference letter for me, and proofreading on Chapter 4 of this thesis. I also had a lot of fun discussing physics and exercises with you; you are a cool teacher. Special thanks to Manuel Rispler, Fatemeh Hajiloo, Stefano Bosco, and Alessandro Ciani, for being good friends, for the soccer games, the espressos, and the parties. It has always been a lot of fun. I also thank Rijul Sachdeva; it was a great pleasure working with you. I thank Veit Langrock for physics and quantum information discussions, also for your idea to draw an illustration for the BOQC scheme. I want to thank Lisa Arndt for inviting me to your birthday party and for your help in the Theo-0. I thank Lucia Gonzalez Rosado for introducing me to your lovely beast Blues.

Fifthly, I'm grateful for the freedom in the software community; I want to thank the community for answering my technical questions. My works mainly involved free(dom) software developed by people that somehow spare their time putting effort into it. My special thanks go to Evgenii Sovetkin. Thank you for helping me a lot in the first years of my PhD, I learned so much about the beauty of math, systematic proofs, Arch Linux, cryptocurrencies, and human kindness from you. I admire your constant passion towards solving math problems for fun. Thank you for many fruitful discussions in math and programming, for sharing codes that improve productivity, proofreading this thesis — especially on the math and definitions part, and thank you for always being a kind and reliable friend. I want to thank my family, although (I'm very sure) you will not read it: buat Susan dan Sipan, terimakasih buat selalu mendukung dan mendoakan Ica.

Last but not least, I would like to thank my partner Jonathan Conrad. Thank you for always being a sweet, supportive, and helpful partner, especially during my thesis writing. I enjoy so much our fruitful discussions in physics and error correction. I admire your work ethic, passion for physics, language skills, and social skills; I learned these things from you. Thank you for proofreading this thesis, your help with German translations, your constructive feedback, and improving my dyslexic-ish writing. I could not have made it without you.

CONTENTS

1	INTRODUCTION	1
2	MEASUREMENT-BASED QUANTUM COMPUTATION	7
2.1	One-Way Quantum Computer	7
2.2	From graph theory to graph states	14
2.3	Measurement Calculus	22
2.4	Determinism in the 1WQC: flow and gflow	24
2.5	Lazy 1WQC	32
3	DELEGATED QUANTUM COMPUTATION AND PRIVACY	49
3.1	Universal Blind Quantum Computation	50
3.2	Abstract Cryptography	58
4	ORACULAR QUANTUM ALGORITHMS	65
4.1	Complexity classes and oracle models	65
4.2	Deutsch and Jozsa's algorithm	72
4.3	Grover search algorithms	74
4.3.1	Grover's algorithm	75
4.3.2	Høyer's algorithm	79
4.4	The exact Grover-Høyer search algorithm	83
4.5	Simon's algorithm	86
5	BLIND ORACULAR QUANTUM COMPUTATION	91
5.1	The BOQC protocol	93
5.2	Security of the BOQC	101
5.2.1	The BOQC-compatible graphs	109
5.3	BOQC with classical input-output and Quantum power	110
5.4	BOQCo is BOQC on solid-state qubits	113
5.4.1	Security of the BOQCo	117
6	BOQC ALGORITHMS: A QUEST FOR THE MOST ECONOMICAL RESULTS	121
6.1	DiVincenzo and Smolin's scheme: extended	122
6.2	Enumerating unique gate topologies	126
6.3	Exhaustive search for the most economical (blind) graph states	130
6.4	Blind Grover-Høyer algorithm	137
6.5	Blind Simon's algorithm	148
6.6	Blind Deutsch-Jozsa algorithm	152
7	NITROGEN-VACANCY CENTERS AS BOQC PLATFORM	155
7.1	The nitrogen-Vacancy center in Diamond	156

7.1.1	Characterization, initialization and control of electron and nuclear spins	156
7.1.2	Quantum network with NV-center nodes	163
7.2	BOQC algorithms on NV-center platform	165
8	SUMMARY AND OUTLOOK	171
	List of Tables	179
	List of Figures	179
	Nomenclature	181
	Appendices	187
A	SOFTWARE OVERVIEW	189
A.1	MBQC	189
A.1.1	Listing open graphs with flow	189
A.1.2	Loose bound cases	192
A.2	cgthesis_supplement	194
A.2.1	uniquedb	194
A.3	Quantum	194
B	mathematica CODES	196
B.1	Grover's algorithm	196
B.2	Grover's algorithm with arbitrary database choice	198
B.3	Exact Grover-Høyer algorithm	201
B.4	Proof of Example 6.5	205
B.5	Simon's algorithm with useless oracles	207

INTRODUCTION

The notion of the *quantum computer*, a computing machine whose behavior is governed by quantum mechanics, has appeared already more than four decades ago. In 1980, Benioff published the first work that incorporates quantum mechanics into computing [2], which was developed further in [3, 4]. In Benioff's works, he introduced a Hamiltonian for a Turing Machine, showing that reversible unitary evolutions can realize Turing Machines without dissipating energy as they evolve. His works were based on Bennett's reversible classical Turing Machine [5] in 1973. While Benioff showed that quantum computers are at least as powerful as (classical) Turing Machines, Feynman was the first to suggest that quantum computer might even be computationally more powerful [6, 7]. Feynman introduced the notion of a universal simulator that simulates physics (quantum systems) exactly. Soon after that, Deutsch pointed out a compelling feature of a quantum computer: the possibility of massive parallelization in a single device [8] — extended in [9], which is well-known as quantum superposition. He initially introduced an oracle problem (the first quantum algorithm) in a setting where quantum superposition becomes necessary, causing the impossibility of any classical Turing Machines to solve the problem. Plenty of follow-up studies emerged afterwards — to name a few [10–20], forming a robust formalism and foundation of the field of quantum information (or quantum computing).

At first, the aim of building quantum computers was pretty much to simulate physical (quantum) systems exactly. Astonishingly, plenty of discoveries and potentials of quantum mechanics in the area that are not directly relevant to the initial goal have been emerging, *e.g.*, quantum and post-quantum cryptography [11, 12, 15, 21–26], quantum internet [13, 27–31], categorical quantum mechanics [32–34], quantum machine learning [35–37], and various quantum algorithms listed in [38]. Quantum computing evolves as a fast-growing interdisciplinary field that intrigues diverse workers, mostly physicists, computer scientists, mathematicians, engineers, and programmers. As a result, nowadays, quantum computing gains traction with industries, government agencies, and banking.

This thesis reflects a small fraction of interdisciplinary traits in quantum computing. The topics that we will touch on here include delegated computation (Section 3.1, Chapter 5), measurement-based computing (Section 2.1), measurement calculus (Section 2.3), security proofs (Sections 3.2 and 5.2), quantum algorithms (Chapter 4, Sections 6.4 to 6.6), numerical optimizations (Sections 6.1 and 6.3), electronic properties (Section 7.1), and a physical implementation (Section 7.2). Given such a diverse background, our primary goal is to produce exciting oracular quantum computations explicitly with the consideration of a physically meaningful setting. In the end, we hope to inspire the experimentalists to consider our results in their demonstrations.

Our work was initially inspired by the experimental demonstration of Barz et al. on *blind quantum computing* (BQC) [39]. BQC is a client-server setting in which a client without a quantum computer delegates a quantum computation to a powerful server that may be dishonest; this setting is known as secure quantum cloud computing. A BQC scheme guarantees the client’s privacy so that even the powerful server cannot gain any valuable information about the client’s input, output, and computation. It is widely believed that cloud computing settings will be prevailing in the future — instead of a personal quantum desktop — for its high maintenance, complexity, and remarkably high-price. Hence, secure cloud quantum computing might become the key to real-life applications in the future.

The blindness in Barz et al.’s experiment is based on the Universal Blind Quantum Computation scheme by Broadbent, Fitzsimons, and Kashefi [25]. It is a cryptographic client-server scheme, where the server has a *One-Way Quantum Computer* (1WQC). 1WQC is a quantum computational model with a paradigm very different from the conventional quantum computation model, *viz.*, the circuit model. 1WQC offers universality with a potentially small set of operations: CPHASE-gates and 1-qubit measurements in the xy -plane basis. Unfortunately, 1WQC computations use graph states as their resource, *i.e.*, a highly entangled state entangled according to a graph, which generally require high number of qubits.

As an overall goal in this thesis, we aim to extend Barz et al.’s experiment by proposing the next challenging demonstration in a cloud computing setting where the oracles are physically meaningful. We incorporate the idea of oracle separation into a cloud computing setting, where there is a party outside the computation with access to the oracle. Outsourcing the oracle is done rather naturally on 1WQC computations, *i.e.*, by outsourcing the block of the graph. Therefore, we can achieve blindness in our scheme (almost) without communication between the client and the oracle. We call our scheme *Blind Oracular Quantum Computation* (BOQC). To this date, there is no known computation setting that separates the

oracle from the computation. We will briefly discuss that it is natural to treat oracles of a quantum algorithm as a separate entity from the algorithm itself.

An oracle is an abstract machine that evaluates an unknown function, which only gives an output — in unit time — when queried. Oracles are widely used in computer science theory. They help us study the relative difficulty among different problems by identifying some barriers to prove results in complexity theory; this concept is known as *relativization*. For instance, relativization is used to relate classes $\mathbf{P}$ and $\mathbf{NP}$, *i.e.*, by proving that $\mathbf{P} \neq \mathbf{NP}$ given access to an oracle, whereas $\mathbf{P} = \mathbf{NP}$ for another oracle [40].

In (oracular) quantum algorithms, relativizations are arranged between classical and quantum settings. For instance, the separation $\mathbf{BQP} \not\subseteq \mathbf{BPP}$ is proven on Simon’s (oracle) problem [41] and implies that a quantum computer may finish the job faster than classical computers. As another example, the statement $\mathbf{BQP} \not\subseteq \mathbf{PH}$ relative to an oracle [42] implies that quantum computers may still have an advantage over classical computers even if the polynomial hierarchy collapses, *i.e.*, when someone proves that $\mathbf{P} = \mathbf{NP}$! Moreover, given the same algorithm, adding internal dice to an oracle allows for stronger separation than the oracle without dice, *e.g.*, Simon’s oracle with dice is unsolvable on classical computers even with infinitely many queries; in contrast, a linear number of quantum queries solve the problem [43].

Nevertheless, a quantum oracle always evaluates an unknown classical function in any model. Depending on the oracle’s behavior, quantum computers might or might not perform better than classical computers. Thus, it is natural to physically separate the oracle from a quantum algorithm for which the BOQC scheme can provide.

BOQC is a scheme that allows a client (Alice) without a quantum computer, nor the capacity to construct the oracle, to privately delegate her oracular quantum algorithm on a powerful quantum server. Thus, she uses Oscar’s service, since he runs a company that provides oracle evaluations, *e.g.*, he provides a database such that Alice’s request is revealed as the output of the computation, only to herself. She also uses Bob’s service since his company provides a quantum computer. However, Alice and Oscar do not risk revealing information about their computation while running it on Bob’s quantum computer; therefore, they run their computation within the BOQC scheme in which Bob is blinded to the client’s computation. We prove Bob’s blindness formally, within a *composable* definition, which means the security of BOQC is also maintained as a part of a more extensive cryptographic system.

Having the confidence that BOQC is a provably secure protocol, we proceed to realize the protocol in a physical setting, in particular, in a nitrogen-vacancy center platform (NV-center). We chose the NV-center system for its progress in individual qubit control [44–48] and quantum networking [49–52]. Our NV-center platform admits the optimized BOQC protocol (BOQCo). BOQCo is a BOQC scheme that employs *lazy 1WQC* instead of 1WQC, such that the physical qubit requirement is minimal. Lazy 1WQC uses the idea of “reusing a qubit” after being measured. The BOQCo works not only on the NV-center platform, but also for systems that can be entangled at a distance whose qubits have quantum-memory like properties, *i.e.*, qubits that possess permanence and can be rapidly re-initialized.

The lazy 1WQC is developed to overcome the extensive requirement needed to run a 1WQC computation. To come up with a minimal resource (graph state), we provide a systematic numerical optimization that exhaustively seeks the resource with a minimal number of qubits. Moreover, our numerical solutions result in graphs that are compatible with BOQC schemes.

The culmination of works in this thesis lies in the last section of [Chapter 7](#), *i.e.*, we provide explicit oracular quantum algorithms that admit BOQC security criteria to be executed on the NV-center platform. The algorithms include 2-qubit Grover using three qubits, 3-qubit exact Grover’s algorithm using four qubits, 2-qubit Simon’s algorithm with *useless*¹ oracle using four qubits, and Deutsch’s algorithm using three physical qubits. These are the algorithms provided by us that hopefully intrigue some experimental physicists to try to demonstrate them.

OUTLINE

As mentioned earlier, this thesis covers a somewhat diverse background; we decided to set the main audience as physicists and computer scientists. Some topics might be more familiar to one than another; however, we try to provide references and examples so that both parties can go through this thesis safely.

This thesis comprises three big topics: blind quantum computation (BQC), quantum algorithms, and physical implementations. BQC-related topics include [Chapter 2](#), [Chapter 3](#), and [Chapter 5](#). Quantum algorithms are covered in [Chapter 4](#) and [Chapter 6](#). Finally, the physical implementation is covered in [Chapter 7](#).

¹ The notion of useless comes from [43, 53], *i.e.*, queries on an oracle is useless if it does not gain any information.

[Chapter 2](#) mainly discusses the one-way quantum computer (1WQC), *i.e.*, the server’s computation model. The chapter provides solid definitions of graph state and 1WQC computations and covers determinism of 1WQC computations — since measurements are the heart of the 1WQC model. We also discuss the measurement calculus formalism. Measurement calculus allows us to express the ongoing processes in a 1WQC. Finally, in the last section of [Chapter 2](#), we introduce our lazy 1WQC scheme. Lazy 1WQC uses the idea of “reusing qubits”, which is also practiced in [\[54\]](#). However, we present lazy 1WQC as a clear-cut scheme that is provably correct, whereas reference [\[54\]](#) covers only the bounds on the number of physical qubits in such a scheme.

[Chapter 3](#) comprises two main discussions: Universal Blind Quantum Computation (UBQC) and abstract cryptography (AC). UBQC is the scheme implemented in Barz et al.’s experiment, as mentioned earlier. It is also the basis of our three-party scheme, *i.e.*, Blind Oracular Quantum Computation (BOQC). AC is a cryptographic framework that we use to prove the security of BOQC protocols.

In [Chapter 5](#), we introduce and elaborate our BOQC protocols: BOQC and BOQCo (BOQC-optimized); BOQCo is the BOQC that incorporates lazy 1WQC scheme. This chapter provides explicit protocols that include adaptive measurements, formal definitions, and security proofs using AC. This chapter presents one of our main contributions.

In the next topic, *i.e.*, quantum algorithms related, we start with [Chapter 4](#) that briefly reviews Turing Machines, computational complexity, and several well-known quantum algorithms: Grover’s algorithm, Høyer’s algorithm, Simon’s algorithm, and the Deutsch-Jozsa algorithm. We proceed to [Chapter 6](#), which presents another of our main contributions. In [Chapter 6](#), we develop a numeric-based algorithm to obtain 1WQC computations (graph states and measurement angles) that admit the security criteria in BOQC (BOQC-compatible graphs). Our numerical algorithms are based on DiVincenzo and Smolin’s algorithm in [\[55\]](#). In this chapter, we produce several BOQC-compatible algorithms: 2- and 3-qubit exact Grover algorithms, 2-qubit Simon algorithm, and a Deutsch algorithm.

The last chapter addresses the overall goal, [Chapter 7](#) covers a physical implementation of a BOQC protocol, particularly the BOQCo protocol. This chapter begins with a review of the nuclear-vacancy center systems (NV-centers). We discuss the electronic structure and qubit definitions in NV-centers. We also touch on the qubit control and distant-entanglement generation. We design an NV-center platform to run the BOQC scheme that works for quantum algorithms with classical input and output; this is also our main contribution. Finally, we adapt the algorithms constructed in [Chapter 6](#) on the specific NV-center setup that we designed.

Lastly, we conclude our works and provide several outlooks in [Chapter 8](#). The appendices contain several explicit algorithms and overviews of the software used in this thesis. A comprehensive list of acronyms, symbols, and operators used in this thesis is provided in the nomenclature.

MEASUREMENT-BASED QUANTUM COMPUTATION

Measurement-based quantum computation introduces a different paradigm from the conventional quantum computation model, *viz.*, the gate model. Processing quantum information is done by a series of simple measurements over a highly entangled state, instead of explicitly operating quantum gates (unitaries). The promise of this simplicity is particularly exciting in quantum information processing.

There are two well-studied measurement-based quantum computation models: *One-Way Quantum Computer* by Raussendorf and Briegel [56, 57] and Teleportation Quantum Computation by Nielsen [58] that are based on the idea of Gottesman and Chuang's work [59]. Additionally, some follow up of Nielsen's work appeared [60, 61]. Nevertheless, both models can eventually be mapped to one another [62]. In this work, we consider only the One-Way Quantum Computer model (1WQC). For those who are interested in both models, we recommend starting with an introduction by Jozsa [63].

In this chapter, we first start with details on the 1WQC scheme; we will learn that a 1WQC computation can be represented with a graph and a set of measurement angles. Secondly, we touch on graphs in an abstract sense and associate them with *graph states*. Thirdly, we briefly discuss the *measurement calculus* formalism, which is employed to express processes during the computation and to formulate proofs. Finally, we introduce our *lazy 1WQC* computation, which minimizes the required number of physical qubits, given that the qubits possess permanence and can be rapidly re-initialized after being measured.

2.1 ONE-WAY QUANTUM COMPUTER

In a nutshell, a computation within the 1WQC model [56] is done by a series of one-qubit measurements over a class of highly entangled quantum states, so-

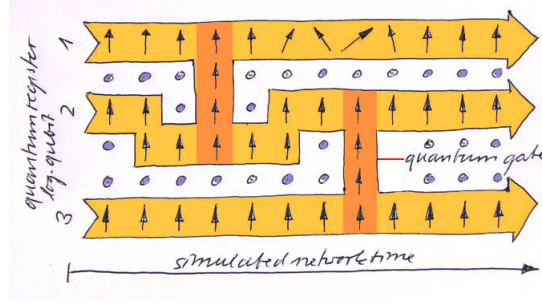


Figure 2.1: An illustration of one-way quantum computer by Raussendorf [64]. The dots and arrows describe qubits sitting on a square lattice. The simulated network time determines the direction of the measurements. The dots signify measurements in the z -basis; the arrows signify measurements in the xy -plane of the Bloch sphere. The z -measurements effectively disregard the qubits, while other measurements form 1- and 2-qubit gates.

called *cluster states*. In this model, a cluster state represents the “quantum computer” (computation resource), while the series of measurement defines the quantum algorithm. Raussendorf created a perfect illustration to describe a 1WQC computation, shown in Figure 2.1. The figure shows an example of how to perform a 1WQC computation.

Figure 2.1 illustrates a computation that starts from a square lattice, where the combination of measurements in z -basis and xy -plane bases shape a cluster state, such that it implements the desired quantum gates. The qubits at the last layer are left unmeasured, for they are the final result of the computation. The computation process is done by “consuming” the resource, *viz.*, measuring a qubit breaks its entanglement; therefore, this model is called the one-way quantum computer.

Interestingly, this paradigm that is remote from conventional quantum computation started from a question “what can one do with ultra-cold atoms in optical lattices where one has only global control over the atom-atom interaction?”[65] — as it came as a surprise that such a resource suffices to perform universal computation.

In 1998, Jaksch et al. [66] proposed a control mechanism between two cold atoms in an optical lattice and a scheme that creates a massive parallel entanglement of many atoms in an optical lattices. Later, Briegel and Raussendorf derived quantum states of such systems in [67] and introduced a class of quantum states, which is cluster state. Cluster states are generated by Ising type next-neighbour interaction with the following interaction Hamiltonian

$$H_{\text{int}} = g(t) \sum_i \frac{1 + \sigma_z^i}{2} \frac{1 - \sigma_z^{i+1}}{2} \propto -g(t) \sum_i \frac{1}{4} \sigma_z^i \sigma_z^{i+1}, \quad (2.1)$$

where σ_z is Pauli- Z operator and strength $g(t)$ is an external control parameter; the parallel entanglement generation is done by switching on H_{int} within a time interval T , such that $\int_0^T dt g(t) = \pi$ [67]. The eigenstates of H_{int} define the cluster state class; in particular, it is defined in Definition 2.1.

Definition 2.1. [56, 67] A cluster state $|\Phi\rangle_{\mathcal{C}}$ with a 1-, 2-, or 3-dimensional¹ cluster² $\mathcal{C}$ is any state that is characterized by

$$\sigma_x^i \bigotimes_{k \in N_{\mathcal{C}}(i)} \sigma_z^k |\Phi\rangle_{\mathcal{C}} = \kappa_i |\Phi\rangle_{\mathcal{C}}, \quad (2.2)$$

where $\kappa_i \in \{1, -1\}$, σ_j signifies a Pauli operator, and $N_{\mathcal{C}}(i)$ is a set of neighbor nodes of i .

Notice that, if $\kappa_i = 1$ for all i , the cluster state is a *stabilizer state* with *stabilizers*³ $\sigma_x^i \bigotimes_{k \in N_{\mathcal{C}}(i)} \sigma_z^k$. For $\kappa = 1$, a cluster state can be constructed by associating every node in $\mathcal{C}$ with the state $|+\rangle := \frac{|0\rangle + |1\rangle}{2}$ and every edge in $\mathcal{C}$ with a CPHASE gate.

Remark that, Definition 2.1 is restricted to the specified cluster $\mathcal{C}$ (2- or 3-dimensional regular lattice). However, one does not necessarily start from a lattice, shaping the lattice into an arbitrary graph by including z -measurements,⁴ but from a graph that implements the desired map. Hence, we can omit z -measurements without compromising universality. Thus, every measurement can be parameterized with one variable, *e.g.*, ϕ . Given

$$|\pm_{\phi}\rangle = \frac{|0\rangle \pm e^{i\phi}|1\rangle}{\sqrt{2}}, \quad (2.3)$$

we call the measurement in the 1WQC scheme as “measure in $|\pm_{\phi}\rangle$ basis”, namely a measurement with projectors

$$\{|+\phi\rangle\langle+\phi|, |-\phi\rangle\langle-\phi|\} \quad (2.4)$$

-
- 1 A two dimensional cluster is needed to perform universal computation, for it can produce two-qubit gates.
 - 2 By cluster here we follow Raussendorf, *i.e.*, a regular lattice.
 - 3 A state encoded by m physical qubits and has m generators (2^m stabilizers) is called a stabilizer state. Stabilizers will be reviewed in Section 2.2.
 - 4 Recall that the z -measurement effectively removes a node.

that will report outcomes **0** and **1** respectively, where $\phi \in [0, 2\pi)$. The projectors are equivalently written as⁵

$$\{|+\phi\rangle\langle+\phi|, |+\phi+\pi\rangle\langle+\phi+\pi|\}. \quad (2.5)$$

A state $|+\phi\rangle$ can be obtained by applying operator $R_z(\phi)$ to the state $|+\rangle$, i.e., $R_z(\phi)|+\rangle$, where we define

$$R_z(\phi) := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}. \quad (2.6)$$

Like the conventional quantum computation model, the main goal of the 1WQC is the universality: capacity to perform any unitary operations on an arbitrary state. Along with this motivation, [Definition 2.2](#) formalizes a 1WQC computation.

Definition 2.2. *Given a graph $\mathcal{G}$, input nodes I , output nodes O , measurement angles $\phi = \{\phi_i\}$, and quantum input ρ , a 1WQC computation is expressed as a set*

$$\{(\mathcal{G}, I, O), \phi, \rho\},$$

where the triplet $(\mathcal{G}, I, O)$ is called an open graph, ρ is assigned to nodes I , nodes O are not measured, and $I \cap O$ can be non-empty.

Note that *open graph* is not a notion from graph theory, but it is a convention introduced in [\[25\]](#). We formally define open graph in [Definition 2.5](#), which is discussed in detail in the next section.

To get the picture of how the 1WQC produces quantum gates, consider [Example 2.1](#), where it shows the simplest case of a 1WQC computation, namely one measurement case. There, we will see that the key to understanding the resulting quantum gates is the z -teleportation.

Example 2.1. *Given $\mathcal{G} = (\{1, 2\}, (1, 2))$, $I = \{1\}$, $O = \{2\}$, input $|\psi\rangle$, and measurement angle $\phi_1 = \alpha$, the final state of a computation characterized by $\mathcal{G}$, I , O , and ϕ is the following.*

$$\begin{array}{c} \text{Measurement of } |\psi\rangle \text{ at angle } \alpha \\ \text{---} \bullet \text{---} \end{array} \stackrel{z\text{-telep}}{=} \begin{array}{c} |\psi\rangle \text{---} \bullet \text{---} \boxed{R_z(-\alpha)} \text{---} \boxed{H} \text{---} \boxed{\text{Measurement } s} \\ |+\rangle \text{---} \bullet \text{---} \text{---} \end{array} \quad X^s H R_z(-\alpha) |\psi\rangle \quad = \quad |\psi\rangle \text{---} \boxed{H R_z(-\alpha)} \text{---} \boxed{X^s} \text{---}$$

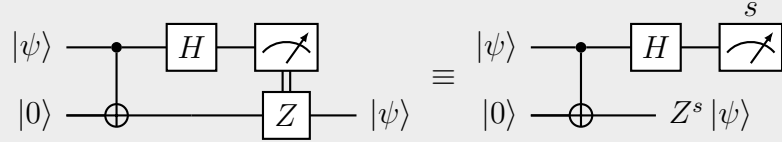
5 $|-\phi\rangle = \frac{|0\rangle + e^{i\pi} e^{i\phi}|1\rangle}{\sqrt{2}} = |+\phi+\pi\rangle.$

The final output is $HR_z(-\alpha)|\psi\rangle$ for measurement outcome 0, and $HR_z(-\alpha+\pi)|\psi\rangle$ for measurement outcome 1. This result is straightforwardly deduced from the z -teleportation.

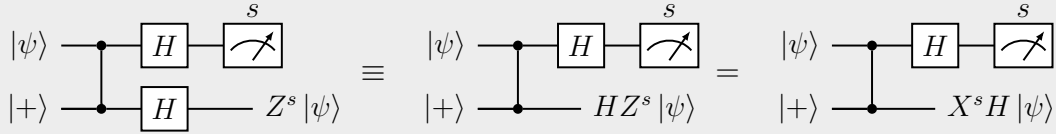


z -teleportation [68]

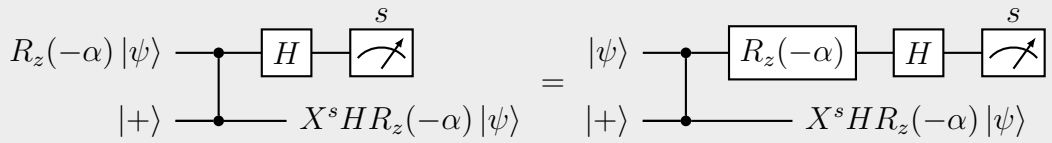
A z -teleportation is teleporting an arbitrary quantum state $|\psi\rangle$ of a qubit to another qubit with state $|0\rangle$; the process comprises entanglement, one qubit measurement, and one z -correction. The circuit is shown below.



By rewriting and commuting some gates, we obtain the following equivalent circuits.



If we apply a z -rotation to the initial state, the rotation is also teleported. Now, let us teleport $R_z(-\alpha)|\psi\rangle$ instead of $|\psi\rangle$.



Which is the result of one measurement in a 1WQC computation from [Example 2.1](#). Thus, a measurement in $|\pm_\alpha\rangle$ basis on the z -teleportation setting results in teleporting $X^s H R_z(-\alpha)|\psi\rangle$, where s is the measurement outcome.

For clarity reason, we will always draw the shape of the nodes according to [Table 2.1](#).

Table 2.1: Descriptions of node shapes used in this thesis.

Node	Description
 	Quantum input, assigned with an arbitrary state, and measured
	Auxiliary node, assigned with state $ +\theta\rangle$, and measured
 	Quantum output, assigned with state $ +\rangle$, and not measured
  	Quantum input and output, assigned with an arbitrary state, and not measured

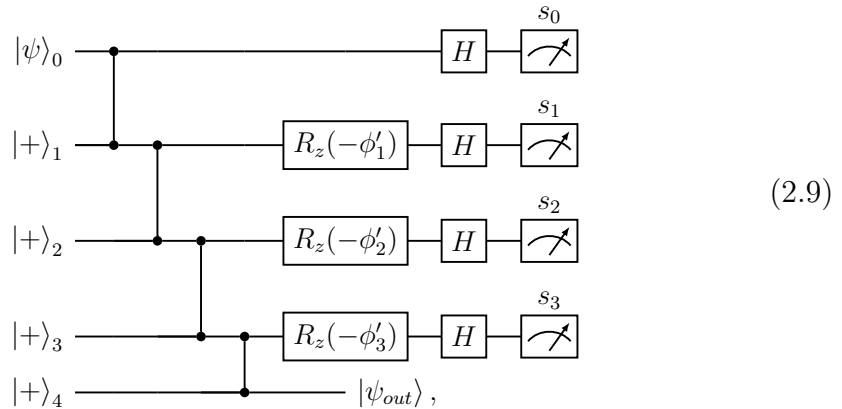
Now, we can show the universality of the 1WQC scheme in implementing unitary maps, *i.e.*, by obtaining an arbitrary 1-qubit rotation and a CNOT gate.⁶ First, any 1-qubit gate can be produced from an open graph

$$\{(\mathcal{G} = \text{star} \text{---} \textcircled{1} \text{---} \textcircled{2} \text{---} \textcircled{3} \text{---} \textcircled{4}, I = \{0\}, O = \{4\}), \phi = (0, -\alpha, -\beta, -\gamma), |\psi\rangle\langle\psi|\}, \quad (2.7)$$

where the input state $|\psi\rangle$ is assigned to node 0. To obtain a deterministic result, we need to perform adaptive measurements *i.e.*, measurements that are depending on the previous measurement outcomes. Consider an adaptive measurement with angles

$$\phi'_0 = 0 \quad \phi'_1 = -(-1)^{s_0}\alpha \quad \phi'_2 = -(-1)^{s_1}\beta \quad \phi'_3 = -(-1)^{s_0+s_2}\gamma, \quad (2.8)$$

where s_i signifies the measurement outcome of node i and measurements are done consecutively from node 0 to node 3. Notice that, as this is the computation of [Example 2.1](#) in the form of a chain, it has circuit



⁶ The resulting CPTP map of a 1WQC computation is discussed in [Section 2.3](#).

where

$$|\psi_{out}\rangle = [X^{s_3} H R_z((-1)^{s_0+s_2}\gamma)][X^{s_2} H R_z((-1)^{s_1}\beta)][X^{s_1} H R_z((-1)^{s_0}\alpha)][X^{s_0} H] |\psi\rangle. \quad (2.10)$$

Let R_x be the rotation operator around the x -axis; we simplify the final output state using the gates commutation relations:

$$\begin{aligned} \clubsuit \quad & XH = HZ \\ \diamondsuit \quad & HR_z(\phi)H = R_x(\phi) \\ \heartsuit \quad & R_x(-\phi)Z = ZR_x(\phi) \text{ and } R_z(-\phi)X = XR_z(\phi), \\ \spadesuit \quad & R_x(\phi)X = XR_x(\phi) \text{ and } R_z(\phi)Z = ZR_z(\phi), \end{aligned} \quad (2.11)$$

and we rewrite the output state as

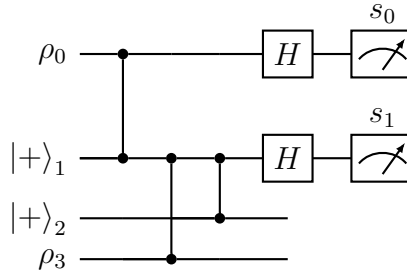
$$\begin{aligned} |\psi_{out}\rangle &\stackrel{\clubsuit}{=} [X^{s_3} H R_z((-1)^{s_0+s_2}\gamma)][HZ^{s_2} R_z((-1)^{s_1}\beta)][X^{s_1} H R_z((-1)^{s_0}\alpha)][HZ^{s_0}] |\psi\rangle \\ &\stackrel{\diamondsuit}{=} X^{s_3} R_x((-1)^{s_0+s_2}\gamma) Z^{s_2} R_z((-1)^{s_1}\beta) X^{s_1} R_x((-1)^{s_0}\alpha) Z^{s_0} |\psi\rangle \\ &\stackrel{\heartsuit}{=} X^{s_3} Z^{s_2} R_x((-1)^{s_0}\gamma) X^{s_1} R_z(\beta) Z^{s_0} R_x(\alpha) |\psi\rangle \\ &\stackrel{\spadesuit, \heartsuit}{=} X^{s_3+s_1} Z^{s_2+s_0} R_x(\gamma) R_z(\beta) R_x(\alpha) |\psi\rangle \end{aligned} \quad (2.12)$$

which can produce any rotation (the Euler rotation theorem).⁷ The final output is deterministic up to Pauli X and Z , which we call it as a “Pauli frame”.

To obtain a CNOT gate, consider computation

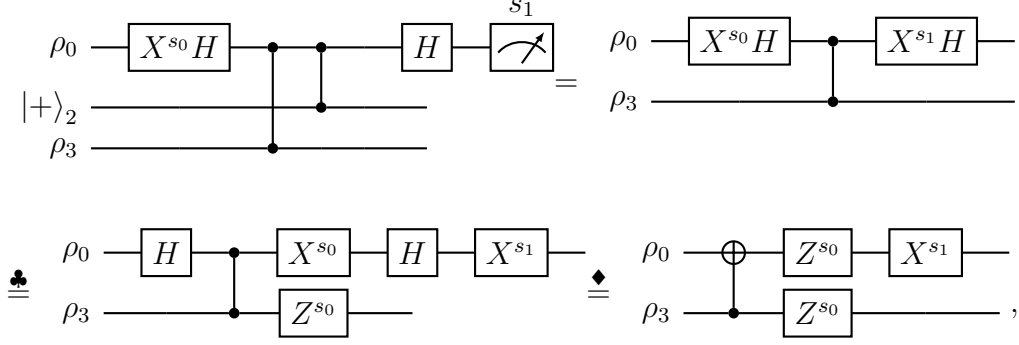
$$\{(\mathcal{G} = \begin{array}{c} \textcircled{3} \\ | \\ \textcircled{0} - \textcircled{1} - \textcircled{2} \end{array}, I = \{0, 3\}, O = \{2, 3\}), \phi = (0, 0), \rho_{03}\}, \quad (2.13)$$

where the corrected angles $\phi' = \phi = 0$. The computation has the following circuit



⁷ The theorem says that any rotation of a rigid body can be described by three angles with respect to a fixed coordinate system[69].

where $\rho_0 \equiv \text{tr}_3(\rho_{03})$ and $\rho_3 \equiv \text{tr}_0(\rho_{03})$. Using z -teleportation and relations: $\clubsuit X_i \text{CPHASE}_{ij} = \text{CPHASE}_{ij} X_i Z_j$, and $\blacklozenge HX = ZH$, we obtain



which is the CNOT gate up to the Pauli frame. Notice that, this open graph has a nonzero intersection between input-output nodes, $I \cap O = \{3\}$; it means node 3 is an untouched input state, which matches the role of a control qubit. Thus, we now have shown that 1WQC is unitary universal.

At this point, we might have the impression that preparing a 1WQC scheme is cumbersome. One may think that it is tricky to see the final map, it is not apparent to obtain the desired graph state, or it is an overly intricate adaptive measurement scheme. However, many workers analyzed those problems from different directions; efficient schemes that address those problems have thus emerged. Here, we employ *measurement calculus* of Danos, Kashefi, and Panangaden in [70] to describe processes within a 1WQC computation; we use the concept of *flow* of Danos and Kashefi in [71] to specify measurement-dependency structure, which also determines the final map of a 1WQC computation.

2.2 FROM GRAPH THEORY TO GRAPH STATES

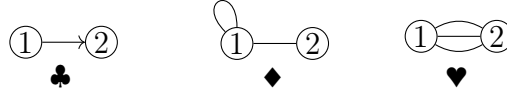
Cluster states — or graph states, the resource states of a 1WQC computation, are derived from a physical system, namely ultracold atoms in an optical lattice. The definition of a cluster state is then simplified to a system with Ising type nearest-neighbor interaction, as written in [Definition 2.1](#); such an approach is known as *bottom-up*. In following, we approach conversely, *viz.*, *top-down*, by starting from an object at a high-level abstraction, *i.e.*, graph theory. The notion of graphs and their properties discussed here are useful as a basic knowledge for future discussions, such as optimization problems and graph iterations.

In the following, we provide several basic definitions of graph theory. While it might sound trivial for some, solid definitions help us to express and to break down complex problems to be solved on a computer. One may encounter slight variations of definitions in other references; it depends on the author's preferred style. Most of graph-related definitions here are adapted from the books by Tutte [72] and by Bondy and Murty [73].

Definition 2.3. A graph $\mathcal{G}$ is an ordered pair $\mathcal{G} = (V, E)$ where $V \equiv V(\mathcal{G})$ is a finite set of elements and $E \equiv E(\mathcal{G})$ is a set of 2-subset of V , i.e., $E \subseteq \{\{x, y\} \mid (x, y) \in V^2, x \neq y\}$. An element of V is called a vertex and an element of E is called an edge. The number of edges connected to vertex v is called degree of v , and the quantity $|V|$ is called the order of $\mathcal{G}$ and $|E|$ is called the size of $\mathcal{G}$.

Notice that Definition 2.3 considers only simple graphs. A simple graph — also known as a strict graph, is an unweighted and undirected graph, without loops and multiple edges. Consider Example 2.2 that shows several non-simple graphs. Note that consider only finite graphs.

Example 2.2. The following graphs are non-simple,



because ♣ has a direction, ♦ has a loop, and ♥ has multiple edges.

A vertex-induced subgraph (or simply subgraph) $\mathcal{F}$ of a graph $\mathcal{G}$ is a graph $\mathcal{F}$ whose vertex set and edge set are subsets of those of $\mathcal{G}$. If subgraph $\mathcal{F}$ has vertex set W , we write

$$\mathcal{F} \equiv \mathcal{G}[W] = (W, E(\mathcal{F})), \text{ where } E(\mathcal{F}) = \{\{i, j\} \in E(\mathcal{G}) \mid i, j \in W\}. \quad (2.14)$$

We indicate the *open neighborhood* or simply neighborhood⁸ of vertex i in $\mathcal{G}$ as

$$N_{\mathcal{G}}(i) := \{k \in V(\mathcal{G}) \mid \{i, k\} \in E(\mathcal{G})\}. \quad (2.15)$$

Given two vertices $v_j, v_k \in V$, vertex v_k is reached from vertex v_j by a *walk*. A walk is a sequence of vertices and edges in a graph, e.g., $v_1 e_1 v_2 e_5 \dots v_n e_m$, where

⁸ We will find later another notion called *closed neighborhood* which includes the vertex itself.

$v_i \in V(\mathcal{G})$ and $e_i \in E(\mathcal{G})$; since $\mathcal{G}$ is a simple graph, it is sufficient to describe a walk as a sequence of vertices $v_1 v_2 \dots v_n$. If every sequence in a walk is distinct, we call it a *path*.

Definition 2.4. A graph $\mathcal{G}$ is a *connected graph*, if any two distinct vertices in $\mathcal{G}$ are connected by a path.

A graph specified by Definition 2.4 is also commonly known as weakly connected.

Definition 2.5. An *open graph* is a triplet $(\mathcal{G}, I, O)$, where $\mathcal{G}$ is a simple connected graph, $I \subset V(\mathcal{G})$, $O \subset V(\mathcal{G})$, and $I \cap O$ can be non-empty.

Recall Definition 2.2; an open graph identifies the graph state in a 1WQC computation. Another important concept that we define is graph isomorphism, which essentially defines equivalence in graphs.

Definition 2.6. *Isomorphism of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ is a bijection between vertex sets of $\mathcal{G}_1$ and $\mathcal{G}_2$*

$$\Theta : V(\mathcal{G}_1) \rightarrow V(\mathcal{G}_2) \quad (2.16)$$

which preserves adjacency and nonadjacency which means

$$\{u, v\} \in E(\mathcal{G}_1) \iff \{\Theta(u)\Theta(v)\} \in \mathcal{G}_2. \quad (2.17)$$

Thus we say graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are *isomorphic*, which is notated as $\mathcal{G}_1 \cong \mathcal{G}_2$.

One may see map Θ as a way of relabelling the nodes. For instance, the following example shows isomorphic graphs, where one is obtained from another by relabeling their nodes.

Example 2.3. *The following statement is true.*



Let $\mathcal{G}_1$ and $\mathcal{G}_2$ define the graph on the left and right side correspondingly. Using Definition 2.6, graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic since there exists a bijection $\Theta : V(\mathcal{G}_1) \rightarrow V(\mathcal{G}_2)$, which maps $1 \mapsto 2$, $2 \mapsto 4$, $3 \mapsto 1$, and $4 \mapsto 3$.

Another representation of a graph is the adjacency matrix. In the following, we will see the direct connection between graph and graph states through the adjacency matrix and the *stabilizers*. First, consider the following definitions.

Definition 2.7. A Pauli group of n -qubit $\mathcal{P}_n$ is defined as all the tensor products of Pauli matrices with phase

$$\mathcal{P}_n = \{i^k I, X, Y, Z\}^{\otimes n}, \text{ where } k \in \{0, 1, 2, 3\}. \quad (2.18)$$

Definition 2.8. Stabilizers $\mathcal{S}$ of an n -qubit state $|\psi\rangle$ is defined as a subset of n -qubit Pauli operators $\mathcal{S} \subset \mathcal{P}_n$, where

$$\mathcal{S} = \{M \in \mathcal{P}_n \mid M|\psi\rangle = |\psi\rangle\}, \quad (2.19)$$

which creates an Abelian group of Pauli operators. The state $|\psi\rangle$ is also called the stabilizer state of $\mathcal{S}$.

A stabilizer group can be uniquely identified by its *generator*. A generator, $\mathcal{S}_g \subset \mathcal{S}$, is a minimum linearly independent set generating a stabilizer group $\mathcal{S}$. We write generator elements within a bracket, e.g., $\mathcal{S}_g = \langle S_1, S_2, \dots \rangle$. One can obtain any element of $\mathcal{S}$ by multiplying some elements of $\mathcal{S}_g$, because a generator forms the basis of a stabilizer group. We refer to Gottesman's thesis [74] for further details on stabilizers. A generator has another representation called *check matrix* — sometimes it is called symplectic or binary representation — defined in the following.

Definition 2.9. Check matrix $M_{\mathcal{S}}$ of an n -qubit generator $\mathcal{S}_g = \langle S_1, \dots, S_m \rangle$ is an $m \times 2n$ matrix in the form of two vertical partitions

$$\left(L \middle| R \right)$$

where each row of the matrix corresponds to an element in the generator and is constructed as follows. Let $S_i \in \mathcal{S}_g$ and $S_i = P_{i1} \dots P_{in}$ where $P_{ij} \in \{X, Y, Z, I\}$,

and let l_{ij} and r_{ij} be the elements of row i of the left and the right partition of the check matrix respectively, then

$$M_{\mathcal{S}_g} = \left(\begin{array}{ccc|ccc} l_{11} & \dots & l_{1n} & r_{11} & \dots & r_{1n} \\ \vdots & & \vdots & \vdots & & \vdots \\ l_{m1} & \dots & l_{mn} & r_{m1} & \dots & r_{mn} \end{array} \right) \text{ where } l_{ij}, r_{ij} = \begin{cases} 0, 0 & \text{if } P_{ij} = I \\ 1, 0 & \text{if } P_{ij} = X \\ 0, 1 & \text{if } P_{ij} = Z \\ 1, 1 & \text{if } P_{ij} = Y \end{cases}. \quad (2.20)$$

The check matrix is also called *tableau representation* in some references [75, 76], where the phase of each element is tracked accordingly; here, we ignore the phases.

Example 2.4. The check matrix of generator $\langle XX, -YY, ZZ \rangle$ is

$$\begin{array}{c} +1 \\ -1 \\ +1 \end{array} \left(\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{array} \right).$$

Recall the definition of a graph state in Definition 2.1. If we set all eigenvalues to one, *viz.*, $\kappa_i = 1 \forall i$, we obtain stabilizers of a graph state. An equivalent definition of a graph state, defined by its stabilizers is shown in Definition 2.10.

Definition 2.10. A graph state is a stabilizer state, whose check matrix can be transformed into the form

$$\left(I \middle| A \right),$$

where on the left partition is an identity matrix I , and the right partition is an adjacency matrix A ; matrix A must be symmetric with zero diagonal.

One may also find other references [77–79] using only the adjacency matrix to represent a graph state. In Definition 2.10, the zero diagonal requirement results in graphs without a loop, the symmetric requirement results in graphs without a direction, and the binary numbers result in graphs without multiple edges. In conclusion, Definition 2.10 gives rise to simple graphs, which is consistent with the definition of open graph in Definition 2.5.

Example 2.5. A generator $\mathcal{S}_g = \langle XZI, ZXZ, IZX \rangle$ is a graph state, because it has a check matrix that is consistent with [Definition 2.10](#), i.e.,

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{array} \right) \equiv \textcircled{1} - \textcircled{2} - \textcircled{3}. \quad (2.21)$$

Constructing a generator from an adjacency matrix or a graph $\mathcal{G}$ is done as follows

$$\mathcal{S}_g = \{X_j \bigotimes_{k \in N_{\mathcal{G}}(j)} Z_k \mid j \in V(\mathcal{G})\}, \quad (2.22)$$

where $N_{\mathcal{G}}(j)$ is the open neighborhood of j — see [Equation \(2.15\)](#).

Apart from the stabilizer, the quantum state also has a direct connection to the check matrix. Consider a check matrix with form $(I \mid A)$, where A is an adjacency matrix that fulfils [Definition 2.10](#). Initially, the nodes are in a product state of

$$|\psi_{init}\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^{\otimes n}} |x\rangle. \quad (2.23)$$

A CPHASE operator that corresponds to nodes $\{j, k\}$ can be written as

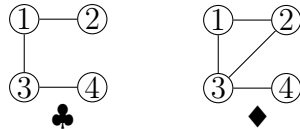
$$\text{CPHASE}_{jk} = \text{CPHASE}_{kj} = (-1)^{jk} |jk\rangle\langle jk| = \exp(-i\pi jk) |jk\rangle\langle jk|, \quad (2.24)$$

which shifts the phase when $jk = 1$, i.e., when $\{j, k\}$ is an edge. Thus, we can obtain the graph state of graph $\mathcal{G}$ with adjacency matrix A as follows

$$|\psi_{\mathcal{G}}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in \{0,1\}^{\otimes n}} \exp\left(-i\pi \sum_{j=1}^n \sum_{k>j}^n A_{jk} x_j x_k\right) |x\rangle, \quad (2.25)$$

where condition $k > j$ prevents the double-counting since A is symmetric.

Example 2.6. Consider graphs $\clubsuit$ and $\diamondsuit$ as shown below. In this example, we construct the generator and the graph state of each graphs $\clubsuit$ and $\diamondsuit$.



Firstly, we use [Equation \(2.22\)](#) to obtain the generators:

$$\mathcal{S}_\clubsuit = \langle XZZI, ZXII, ZIXZ, IIZX \rangle \quad \mathcal{S}_\diamond = \langle XZZI, ZXZI, ZZXZ, IIZX \rangle.$$

Secondly, we use [Equation \(2.25\)](#) to obtain the quantum states:

$$\begin{aligned} |\psi_\clubsuit\rangle &= \frac{1}{4} \sum_{x \in \{0,1\}^4} (-1)^{x_1x_2+x_1x_3+x_3x_4} |x_1x_2x_3x_4\rangle \\ |\psi_\diamond\rangle &= \frac{1}{4} \sum_{x \in \{0,1\}^4} (-1)^{x_1x_2+x_1x_3+x_2x_3+x_3x_4} |x_1x_2x_3x_4\rangle. \end{aligned}$$

Definition 2.11. *Given a graph $\mathcal{G} = (V, E)$ and edges of a complete graph $K = \{\{v, w\} \in V \times V \mid v \neq w\}$, the complement of $\mathcal{G}$ is the graph $\mathcal{G}^c = (V, K \setminus E)$.*

Another graph operation that we introduce here is *local complementation* (LC), defined as follows.

Definition 2.12. [\[80\]](#) *The local complementation of $\mathcal{G}$ at v , indicated as $\text{LC}_v[\mathcal{G}]$, is obtained from $\mathcal{G}$ by replacing the subgraph $\mathcal{G}[N_{\mathcal{G}}(v)] =: \mathcal{F}$ in $\mathcal{G}$ with its complement, $\mathcal{F}^c$.*

For instance, graphs in [Example 2.6](#) are related by the LC operation, namely $\mathcal{G}_\diamond = \text{LC}_1[\mathcal{G}_\clubsuit]$; for more examples, see [Example 2.7](#). In addition, the LC has a direct connection to graph state. Nest *et al.* have shown that two graphs are equivalent under the local Clifford group if and only if there exists a sequence of local complementations which relates their associated graph [\[77\]](#).

Definition 2.13. [\[79\]](#) *Let $\mathcal{G}$ be a graph state. The local complementation operator, with parameter vertex j , which is notated as LC_j is the unitary*

$$\text{LC}_j = \sqrt{X_j} \bigotimes_{k \in N_{\mathcal{G}}(j)} \sqrt{Z_k}, \quad (2.26)$$

where operators $\sqrt{X}$ and $\sqrt{Z}$ are the square root of Pauli matrices that can be chosen to be

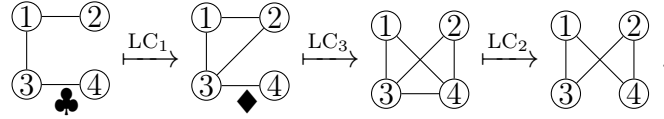
$$\sqrt{Z} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \quad \sqrt{X} = \sqrt{\frac{i}{2}} \begin{bmatrix} -i & 1 \\ 1 & -i \end{bmatrix}. \quad (2.27)$$

Given a graph state $|\psi_{\mathcal{G}}\rangle$ has graph representation $\mathcal{G}$. An LC operator applied to a quantum state $\text{LC}_i |\psi_{\mathcal{G}}\rangle$ follows the transformation of [Definition 2.13](#), whereas the LC operation applied to a graph $\text{LC}_i[\mathcal{G}]$ follows the transformation of [Definition 2.12](#). The LC operator and LC are isomorphic, namely $\text{LC}_i |\psi_{\mathcal{G}}\rangle = |\psi_{\text{LC}_i[\mathcal{G}]}\rangle$.

Clearly, applying an LC operator twice brings a graph into its original form; the same applies to graph states, *e.g.*, $\text{LC}_j^2 |\psi\rangle = |\psi\rangle$. The square root of Pauli operators might look peculiar, however they belong to the Clifford group, because

$$\begin{aligned} \sqrt{Z}Z(\sqrt{Z})^\dagger &= Z & \sqrt{Z}X(\sqrt{Z})^\dagger &= Y & \sqrt{Z}Y(\sqrt{Z})^\dagger &= -X \\ \sqrt{X}Z(\sqrt{X})^\dagger &= Y & \sqrt{X}X(\sqrt{X})^\dagger &= X & \sqrt{X}Y(\sqrt{X})^\dagger &= -Z. \end{aligned} \quad (2.28)$$

Example 2.7. Successive LC operations on vertices 1, 3, and 2 on graph $\mathcal{G}_{\clubsuit}$ ([Example 2.6](#)), transform graph $\mathcal{G}_{\clubsuit}$ as follows.



We see that LC can transform one graph into another. In [Example 2.8](#), some graphs might look more entangled by the presence of more edges; however, the entanglement is not increasing since local operations alone cannot increase entanglement.

Example 2.8. Operating LC_1 on a generator $\mathcal{S}_g = \langle XZZI, ZXII, ZIXZ, IIZX \rangle$ is done as the following. Using [Definition 2.13](#), $\text{LC}_1 \equiv U := \sqrt{X}_1 \otimes \sqrt{Z}_2 \otimes \sqrt{Z}_3$. By applying U to each stabilizer element of $\mathcal{S}_g$, we obtain

$$\mathcal{S}'_g := U\mathcal{S}_gU^\dagger = \langle XZZI, YYII, YIYZ, IIZX \rangle \equiv \langle S_1, S_2, S_3, S_4 \rangle.$$

Since a generator forms a stabilizer basis, we can replace S_2 with $S_1 \cdot S_2$ and S_3 with $S_1 \cdot S_3$. Note that the multiplication order is irrelevant since stabilizers forms an abelian group. Thus, we obtain another equivalent generator

$$\mathcal{S}'_g = \langle XZZI, -ZXZI, -ZZXZ, IIZX \rangle.$$

Stabilizers with minus sign correspond to the $\kappa = -1$ in [Definition 2.1](#). For example, $ZXZI |\psi\rangle = -|\psi\rangle$ and $ZZXZ |\psi\rangle = -|\psi\rangle$, given $|\psi\rangle$ is the stabilizer state of $\mathcal{S}'_g$. Thus, it is consistent with the definition graph state in [Definition 2.1](#). In addition, $\mathcal{S}_g$ and $\mathcal{S}'_g$ correspond to graphs $\mathcal{G}_{\clubsuit}$ and $\mathcal{G}_{\blacklozenge}$ in [Example 2.7](#), respectively.

LC operations have direct applications. For instance, a privately delegated 1WQC computation experiment conducted by Barz et al. [1], employed LC operations to realize a graph state beyond a standard setting only by adjusting the measurement angles. Moreover, the inter-relation between LC operation and graph states still has room to explore, not only in the quantum computing field, but also in other areas such as quantum networking, secret sharing, and (plausibly) error correction. For instance, a recent study in [81] explored some properties of graphs and entanglement in graph states under LC operations, such as colorability and Schmidt measure, to seek relationships between connectivity and the entanglement of its graph states; they found some entertaining connection called quantum Rubik’s cube.

2.3 MEASUREMENT CALCULUS

In Section 2.1, we showed the unitary universality of 1WQC, by realizing an arbitrary rotation gate and a CNOT gate in the 1WQC scheme. The correctness of computation is demonstrated through the z -teleporation picture⁹, which indeed realizes the expected gates. Such reasoning, however, is cumbersome and impractical for large computations. Therefore, here we employ *measurement calculus* that is introduced by Danos, Kashefi, and Panangaden [70]. Measurement calculus is a mathematical model underlying the 1WQC scheme, which presents a concrete syntax and operational semantics for programs, called *measurement pattern* or just *pattern*. We use measurement calculus to reason locally and compositionally about these patterns.

Definition 2.14. [70] *A pattern consists of three finite sets (V, I, O) together with a finite sequence of commands $A_n \dots A_1 := \mathfrak{P}$, read from right to left, applying to qubits in V in that order, i.e., A_1 first and A_n last, such that:*

- (D0) *no command depends on an outcome not yet measured*
- (D1) *no command acts on a qubit already measured*
- (D2) *no command acts on a qubit not yet prepared, unless it is an input qubit*
- (D3) *a qubit i is measured if and only if i is not an output.*

Conditions (D0)–(D3) are essentially capturing the causality. The set V signifies the *measurement space*, set I is the pattern input, and set O is the pattern output. We write $\mathcal{H}_V$, $\mathcal{H}_I$, and $\mathcal{H}_O$ as the quantum state space associated with V , I , and O , respectively. The sequence $A_n \dots A_1$ is called *pattern command sequence*. A command sequence comprises basic commands that are tabulated in Table 2.2.

⁹ See the extra box in Section 2.1 for the z -teleporation.

Table 2.2: Basic commands in patterns.

Command	Name	Description
s_i	Signal	measurement outcome of qubit i
N_j	Preparation	prepare qubit j in state $ +\rangle$
E_{ij}	Entanglement	apply CPHASE between qubits i and j
M_j^θ	Measurement	measure i in $ \pm_\theta\rangle$ basis (destructive measurement)
X_i^s, Z_i^s	Corrections	Pauli corrections on qubit i with a signal s .

Running a pattern is done by executing each command from right to left. Given n measurements in pattern $\mathfrak{P}$, with outcomes $s \in \{0, 1\}^n$, there are 2^n possible s , where each s results in a *branch map* E_s representing a linear transformation $\mathcal{H}_I \rightarrow \mathcal{H}_O$. A branch map admits a decomposition $E_s = C_s \Pi_s U$, where C_s is a unitary map $\mathcal{H}_O \rightarrow \mathcal{H}_O$, collecting all corrections on the outputs, Π_s is a projection $\mathcal{H}_V \rightarrow \mathcal{H}_O$, representing the particular measurement on the corresponding branch, and U is the isometry $\mathcal{H}_I \rightarrow \mathcal{H}_V$, collecting the preparation and entangling operations. Therefore,

$$\sum_s E_s^\dagger E_s = \sum_s U^\dagger \Pi_s U = I, \quad (2.29)$$

and

$$\sum_s E_s \rho E_s^\dagger := \mathcal{E}(\rho) \quad (2.30)$$

is a CPTP map, explicitly given in the operator sum representation (Kraus form). Thus, pattern $\mathfrak{P}$ realizes CPTP map $\mathcal{E}$.

The calculus of patterns (measurement calculus) comprises the same rule as vector algebra in a Hilbert space; however, it employs the pattern syntax. For instance, we can concatenate two patterns, which is analogous to a tensor product between two Hilbert spaces. We present several rules that we will use in this thesis. The first equations are the Pauli propagations, which are analogous to the Pauli commutation relations.

$$\begin{aligned}
E_{ij} X_i^s &= X_i^s Z_j^s E_{ij} \\
E_{ij} X_j^s &= X_j^s Z_i^s E_{ij} \\
E_{ij} Z_i^s &= Z_j^s E_{ij} \\
E_{ij} Z_j^s &= Z_i^s E_{ij}.
\end{aligned} \quad (2.31)$$

The second equations are the adaptive measurements that correct measurement angles with the presence of Pauli corrections.

$$\begin{aligned} M_i^\phi X_i^s &= M_i^{(-1)^s \phi} \\ M_i^\phi Z_i^s &= M_i^{\phi + s\pi}. \end{aligned} \quad (2.32)$$

The Pauli corrections allows a pattern to implement a deterministic unitary map; consider [Example 2.9](#) for an illustration.

Example 2.9. *In the following, we obtain a pattern that deterministically realizes CNOT gate.*

Recall [Equation \(2.13\)](#), the following 1WQC computation realizes CNOT gate.

$$\{(\mathcal{G} = \begin{array}{c} \textcircled{3}^* \\ | \\ \textcircled{1} \\ | \\ \textcircled{0}^* \end{array} - \textcircled{1} - \textcircled{2}, I = \{0, 3\}, O = \{2, 3\}, \phi = (0, 0), \rho_{03}\}.$$

We perform the same procedure as in the process 1WQC scheme: (1) input preparation, (2) auxiliary nodes initialization, (3) qubit entanglements, (4) qubit measurements, and (5) Pauli corrections.

Step (1) is preparing state ρ_{03} on input qubits I . Step (2) is initializing qubits 1 and 2 in state $|+\rangle$, equivalently $N_2 N_1$. Step (3) is entangling qubits with $E_{13} E_{12} E_{01}$. Step (4) is measuring qubit 0 with angle 0 (M_0^0), followed by measuring qubit 1 with angle 0 (M_1^0). Step (5) is correcting qubit 2 with $Z^{s_0} X^{s_1}$ and qubit 3 with Z^{s_0} , where s_i is the signal of qubit i . Arranging all these commands together, we obtain pattern

$$Z_3^{s_0} X_2^{s_1} Z_2^{s_0} M_1^0 M_0^0 E_{13} E_{12} E_{01} N_2 N_1,$$

which deterministically realizes CNOT gate.

2.4 DETERMINISM IN THE 1WQC: FLOW AND GFLOW

Contrary to classical physical systems, quantum systems possess true randomness, which can be harnessed via measurements. As measurement is the gist of a 1WQC computation, indeterminacy becomes inevitable, yet determinism is fundamental for the computation. Hence, adaptive measurements are performed to compute deterministically; for instance, in [Section 2.1](#), [Equation \(2.8\)](#), the measurement angles are adapted accordingly, depending on the signals (previous measurement outcomes). So, what are the criteria to achieve determinism in the 1WQC scheme?

Several methods that guarantee determinism (by construction) have been proposed [78, 82, 83]. Interestingly, Danos and Kashefi approached the problem from a different perspective in [71]; they introduced a direct condition on open graphs¹⁰ that guarantees deterministic behavior on computations defined over them. Their criteria bear only on the geometric structure of the graph, which then single out a class of patterns with *flow*.

Note that, here we use the measurement calculus syntax that was discussed in Section 2.3. Given an open graph $(\mathcal{G}, I, O)$ with $\mathcal{G} = (V, E)$, a *flow* f is a map from the measured qubits to the prepared qubits $f : O^c \rightarrow I^c$, where A^c denotes the complement of set A . More specifically for cluster states, $f(j)$ indicates the X correction and $N_{\mathcal{G}}(f(j))$ ¹¹ indicate Z corrections for the measured node j .

Definition 2.15. [71] *An open graph $(\mathcal{G}, I, O)$ has flow if there exists a map $f : O^c \rightarrow I^c$ together with a partial order $\succ$ over nodes V such that for all $j \in O^c$:*

$$\begin{aligned} (\mathbf{F0}) \quad & (j, f(j)) \in E, \\ (\mathbf{F1}) \quad & f(j) \succ j, \\ (\mathbf{F2}) \quad & \forall k \in N_{\mathcal{G}}(f(j)) \setminus \{j\}, k \succ j. \end{aligned} \tag{2.33}$$

In practice, it has been discovered that the flow of an arbitrary open graph can be computed efficiently — if the flow exists — using Mhalla and Perdrix’s method [84]. In general, flow is not unique if it exists, except that for $|O| = |I|$, the flow is unique [85]. By construction from the given criteria, a flow induces a partial ordering that covers V . We will see that such an ordering is the key to determinism.

Theorem 2.1. [71] *Suppose the open graph state $(\mathcal{G}, I, O)$ has flow $(f, \succ)$, then the pattern:*

$$\mathfrak{P}_{f, \mathcal{G}, \succ, \theta} := \mathop{\mathbb{H}}\limits_{i \in O^c}^{\succ} \left(X_{f(i)}^{s_i} \left(\prod_{k \in N_{\mathcal{G}}(f(i)) \setminus \{i\}} Z_k^{s_i} \right) M_i^{\theta_i} \right) E_{\mathcal{G}} N_{I^c}^0 \tag{2.34}$$

is deterministic for all angles θ and signals s , and realizes isometry $\bigotimes_{i \in O^c} \langle +_{\theta_i} |_i E_{\mathcal{G}} N_{I^c}^0$.

¹⁰ An open graph is a graph with inputs and outputs, see Definition 2.5 in Section 2.2 for the formalism.

¹¹ Recall the definition of an open neighborhood $N_{\mathcal{G}}(i)$ is given in Equation (2.15).

Notice that [Theorem 2.1](#) introduces new notations. First, entanglement over a graph $\mathcal{G} = (E, V)$,

$$E_{\mathcal{G}} := \prod_{(i,j) \in E(\mathcal{G})} E_{ij}. \quad (2.35)$$

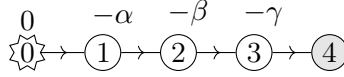
Note that, symbol E without subscript denotes a set of edges and the one with two subscripts, *e.g.*, E_{ij} indicates the entanglement command. Secondly, The preparation command over a set,

$$N_{I^c}^{\alpha} := \prod_{i \in I^c} N_i^{\alpha_i}, \quad (2.36)$$

where in the pattern above, $\alpha_i = 0$ for all i . Lastly, a product notation $\mathbb{H}$ signifies an ordered product with respect to the partial order $\succ$.

Obtaining a pattern is straightforward once the flow is known. Consider [Example 2.10](#) for an illustration.

Example 2.10. *A 1-qubit arbitrary rotation realized with $R_x(\gamma)R_z(\beta)R_x(\gamma)$ is achieved with four measurements on an open graph $(\mathcal{G}, I, O)$, where $I = \{0\}$ and $O = \{4\}$, as shown below.*



The desired measurement angles are specified above the nodes; the flow is identified with arrows, where $f(0) = 1, f(1) = 2, f(2) = 3$, and $f(3) = 4$. By using [Theorem 2.1](#), Pauli commutation relations, and adaptive measurements from [Equation \(2.32\)](#), we obtain pattern

$$\begin{aligned} \mathfrak{P} &:= X_4^{s_3} M_3^{-\gamma} Z_4^{s_2} X_3^{s_2} M_2^{-\beta} Z_3^{s_1} X_2^{s_1} M_1^{-\alpha} Z_2^{s_0} X_1^{s_0} M_0^0 E_{\mathcal{G}} N_{1,2,3,4} \\ &= X_4^{s_3} Z_4^{s_2} M_3^{-\gamma} X_3^{s_2} Z_3^{s_1} M_2^{-\beta} Z_2^{s_0} X_2^{s_1} M_1^{-\alpha} X_1^{s_0} M_0^0 E_{\mathcal{G}} N_{1,2,3,4} \\ &= X_4^{s_3} Z_4^{s_2} M_3^{(-1)^{s_2}\gamma + \pi s_1} M_2^{(-1)^{s_1}\beta + \pi s_0} M_1^{(-1)^{s_0}\alpha} M_0^0 E_{\mathcal{G}} N_{1,2,3,4}, \end{aligned}$$

which realizes a one-qubit unitary operation up to Pauli frame $X_4^{s_3} Z_4^{s_2}$. Notice that we obtain different adaptive measurements and Pauli frame from the one in [Equation \(2.8\)](#); here, measurements of node 2 and node 3 also taking into account the missing Pauli frame $X_4^{s_1} Z_4^{s_0}$.

[Theorem 2.1](#) states that the existence of a flow guarantees a deterministic computation; however, the reverse is untrue [\[86\]](#). The sufficient and necessary condition

that guarantees a deterministic computation of a graph is the existence of a *gflow*, which is introduced by Browne et al. in [86]. Gflow stands for “generalized flow”; it extends the correction scheme of flow by allowing a subset of vertices to correct one measurement. A gflow g is a map $g : O^c \rightarrow \mathcal{P}(I^c)$, where $\mathcal{P}(I^c)$ denotes the power set of I^c . Note that, we only consider graphs that have flow in this thesis. However, for its interesting features, for its potential in further development of measurement-based quantum computation, and for giving us deeper understanding in deterministic computation, we shortly discuss gflow here.

There are two ingredients to determinism in a 1WQC computation [86]:

$$\begin{aligned} \clubsuit \quad & \langle +_\theta |_i = M_i^\theta Z^{s_i} \\ \diamondsuit \quad & S_i = X_i \left(\prod_{j \in N_G(i)} Z_j \right). \end{aligned} \tag{2.37}$$

Firstly, we convince ourselves that the first statement ($\clubsuit$) is true. Given an arbitrary state ρ that lies in a Hilbert space of qubit i , *viz.*, $\mathcal{H}_i$ — can also be larger than $\mathcal{H}_i$, we measure ρ in basis $|\pm_\theta\rangle$, *i.e.*, with measurement projectors $\{|+\theta\rangle\langle +_\theta|_i, |-\theta\rangle\langle -_\theta|_i\}$. The probability (p) for each outcome ($s_i = 0$ and $s_i = 1$) is

$$\begin{aligned} p(0) &= \text{tr}_i(|+\theta\rangle\langle +_\theta|_i \rho) = \langle +_\theta | \rho | +_\theta \rangle_i \\ p(1) &= \text{tr}_i(|-\theta\rangle\langle -_\theta|_i \rho) = \langle -_\theta | \rho | -_\theta \rangle_i = \langle +_\theta | Z_i \rho Z_i | +_\theta \rangle_i \end{aligned}$$

The final state for each outcome is

$$\begin{aligned} \rho'(0) &= \text{tr}_i \left(\frac{|+\theta\rangle\langle +_\theta|_i \rho |+\theta\rangle\langle +_\theta|_i}{\sqrt{\langle +_\theta | \rho | +_\theta \rangle_i}} \right) = \frac{\langle +_\theta | \rho | +_\theta \rangle_i}{\sqrt{\langle +_\theta | \rho | +_\theta \rangle_i}} \\ \rho'(1) &= \text{tr}_i \left(\frac{|-\theta\rangle\langle -_\theta|_i \rho |-\theta\rangle\langle -_\theta|_i}{\sqrt{\langle -_\theta | \rho | -_\theta \rangle_i}} \right) = \frac{\langle -_\theta | \rho | -_\theta \rangle_i}{\sqrt{\langle -_\theta | \rho | -_\theta \rangle_i}} = \frac{\langle +_\theta | Z_i \rho Z_i | +_\theta \rangle_i}{\sqrt{\langle +_\theta | Z_i \rho Z_i | +_\theta \rangle_i}}. \end{aligned}$$

Thus, if we have measurement projectors as $\{|+\theta\rangle\langle +_\theta|_i, Z |+\theta\rangle\langle +_\theta|_i Z\}$ (equivalently, $M_i^\alpha Z^{s_i}$) instead, we will always obtain an outcome with certainty. The catch is, we need a future’s knowledge (what the outcome will be), which breaks causality. The second ingredient ($\diamondsuit$) comes to the rescue, namely the graph state stabilizers. Recall from the definition of stabilizers (Definition 2.8); as they left the graph state unchanged, we can arbitrarily apply any of the stabilizers to the graph state; it restores the causality in the following manner.

Now, we demonstrate how applying particular stabilizers leads to a deterministic computation. Consider a 1WQC with graph $\mathcal{G}$ defined below, where the number also indicates the measurement order.

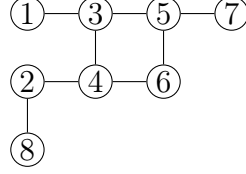


Figure 2.2: Graph $\mathcal{G}$.

Given that we are at the stage of measuring node 3. To obtain a deterministic outcome, we apply operator $Z_3^{s_3}$ to node 3 before measurement. If we picture the graph with a timeline, one may obtain Figure 2.3 below.

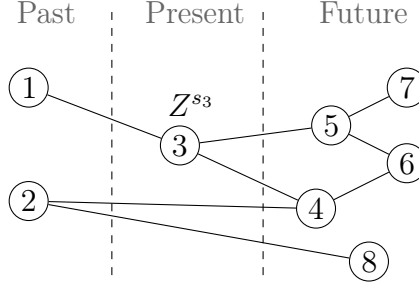


Figure 2.3: Measure node 3 of graph $\mathcal{G}$ with the timeline shown.

Now, the key is applying the proper stabilizer S_i to eliminate acausality $Z_3^{s_3}$. There are two choices, either applying stabilizer $S_4^{s_3}$ (Figure 2.4a) or $S_5^{s_3}$ (Figure 2.4b) in the future, after we know the outcome s_3 .

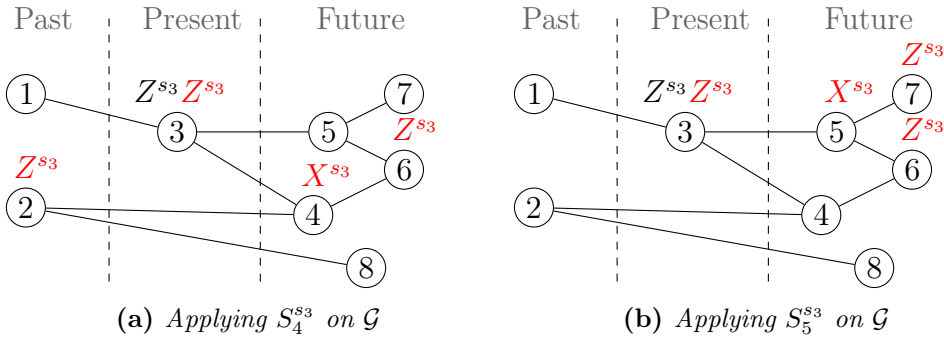


Figure 2.4: Possible operation of stabilizers in order to eliminate acausality $Z_3^{s_3}$. The choice are the neighborhood nodes of 3, because the definition of a stabilizer S_i in Equation (2.37) makes it possible to eliminate a Pauli Z .

Both cases in Figure 2.4 eliminate acausality Z^{s_3} at node 3; however, the case in Figure 2.4a introduces another acausality, *i.e.*, we need to apply Z^{s_3} to node 2 that lies in the past. Thus, the correction of measuring node 3 is done with applying S_5 , which means $f(3) = 5$. This causality of choosing S_4 or S_5 is captured by flow conditions (F1) and (F2), in Equation (2.33). A flow is essentially choosing the future stabilizer to correct past measurements.

On the other hand, as the image of a gflow is a set, S_4 can still be chosen to correct measurement 3, but it must come together with S_8 in order to eliminate acausality caused by S_4 . If we apply both stabilizers, it results in Figure 2.5.

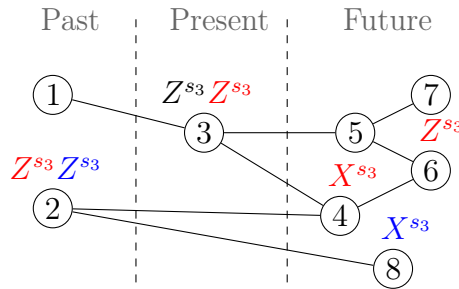


Figure 2.5: The correction of measuring node 3 with gflow.

Thus, the gflow of 3 can either be $g(3) = \{4, 8\}$ or $g(3) = \{5\}$. The choices given by a gflow give advantages to systems that have more restrictions in entanglement. Like the criteria of flow, the gflow also has three criteria, defining a partial ordering that covers the vertices.

Consider an open graph $(\mathcal{G}, I, O)$. Given a subset of vertices $K \subset V(\mathcal{G})$, an *odd neighbourhood* $\text{Odd}(K)$ is defined as

$$\text{Odd}(K) := \{v \in V(\mathcal{G}) \mid |N_{\mathcal{G}}(v) \cap K| = 1 \pmod{2}\}. \quad (2.38)$$

For example, consider the graph in Figure 2.5, and we set $K = \{4, 8\}$. The odd neighbourhood of K is $\text{Odd}(K) = \{3, 6\}$, since $|N_{\mathcal{G}}(2) \cap K| = 2$, $|N_{\mathcal{G}}(3) \cap K| = 1$, and $|N_{\mathcal{G}}(6) \cap K| = 1$. The definition of gflow given in Definition 2.16.

Definition 2.16. An open graph state $(\mathcal{G}, I, O)$ has a gflow if there exists a map $g : O^c \rightarrow \mathcal{P}(I^c)$ and a partial order $\succ$ over $V(\mathcal{G})$, such that for all $i \in O^c$:

$$\begin{aligned} (G0) \quad & j \in g(i) \Rightarrow i \prec j \\ (G1) \quad & j \prec i \Rightarrow j \in \text{Odd}(g(i)) \\ (G2) \quad & i \notin g(i) \Rightarrow i \in \text{Odd}(g(i)), \end{aligned} \quad (2.39)$$

where $\mathcal{P}(I^c)$ indicates the powerset of I^c .

Theorem 2.2. [86] Suppose the open graph state $(\mathcal{G}, I, O)$ has a gflow $(g, \succ)$, then the pattern

$$\mathfrak{P}_{g,\mathcal{G}} := \prod_{i \in O^c}^{\succ} \left(X_{g(i) \setminus \{i\}}^{s_i} Z_{\text{Odd}(g(i))}^{s_i} M_i^{\theta_i} \right) E_{\mathcal{G}} N_{I^c},$$

where the product follows the dependency order $\succ$, is deterministic for all θ_i and for any signals s_i , is stepwise deterministic, and realizes the isometry $(\prod_{i \in O^c} \langle +_{\theta_i} |) E_{\mathcal{G}} N_{I^c}$.

The resulting computation using gflow is the same with the one with flow, however gflow provides a stronger statement, namely the pattern is also *stepwise deterministic*. A pattern is stepwise deterministic if it is deterministic after each single measurement is performed and the corrections are applied. Interestingly, gflow is not only restricted to measurements in the xy -plane bases, but can also be adjusted to other planes, *viz.*, xz - and yz -planes

Here, we use the pattern decomposition method from [71] to translate a pattern into a quantum circuit and vice-versa,¹² which necessitates the graphs to have a flow. Thus, in this thesis, we consider only graphs that have a flow.



Gflow with measurements on the xz - and yz -planes.

The strategy to construct a deterministic computation in a 1WQC also applies to measurements in other planes on graph states. Firstly, we denote

$$\begin{aligned} |+\theta^{\text{xz}}\rangle &:= \cos \frac{\theta}{2} |0\rangle + \sin \frac{\theta}{2} |1\rangle \quad \text{a state on the } xz \text{ plane with angle } \theta, \\ |+\theta^{\text{yz}}\rangle &:= \cos \frac{\theta}{2} |0\rangle + i \sin \frac{\theta}{2} |1\rangle \quad \text{a state on the } yz \text{ plane with angle } \theta. \end{aligned}$$

Secondly, the measurement with a parameter $\theta \in [0, 2\pi)$ on xz -plane ($M^{\text{xz},\theta}$) and on yz -plane ($M^{\text{yz},\theta}$) have projectors (and measurement outcomes)

$$\begin{aligned} &\{(|+\theta^{\text{xz}}\rangle\langle+\theta^{\text{xz}}|, 0), (|+\theta+\pi^{\text{xz}}\rangle\langle+\theta+\pi^{\text{xz}}|, 1)\} \quad \text{and} \\ &\{(|+\theta^{\text{yz}}\rangle\langle+\theta^{\text{yz}}|, 0), (|+\theta+\pi^{\text{yz}}\rangle\langle+\theta+\pi^{\text{yz}}|, 1)\} \end{aligned}$$

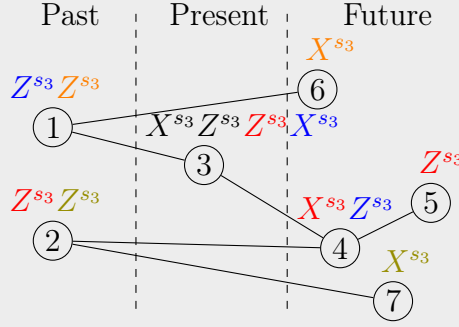
¹² The translation between 1WQC computation and quantum circuit is demonstrated in [Section 6.3](#), *e.g.*, [Example 6.3](#).

respectively, together with its corresponding measurement outcomes — 0 or 1. Thus, a deterministic measurement means having

$$\begin{aligned} \langle +_{\theta}^{xz} |_i &= M_i^{xz, \theta} Y^{s_i} = M^{xz, \theta} Z^{s_i} X^{s_i} \quad \text{and} \\ \langle +_{\theta}^{yz} |_i &= M_i^{yz, \theta} X^{s_i} \end{aligned}$$

accordingly. Like 1WQC, one only need to place the proper stabilizers to eliminate the acausality.

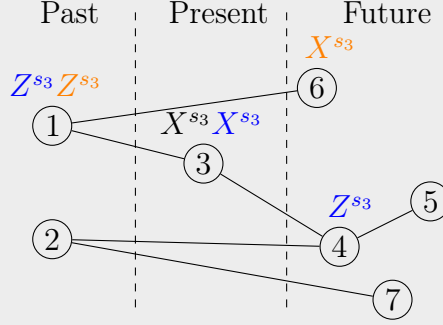
Consider the following figure that shows a measurement on qubit 3 with an xz -plane basis, *e.g.*, with pattern $M_3^{xz, \theta}$, together with an effort to achieve determinism.



The determinism is achieved by eliminating acausality $X_3^{s_3} Z_3^{s_3}$. Acausality $X_3^{s_3}$ is eliminated by stabilizers $S_4^{s_3}$ and $S_7^{s_3}$. Acausality $Z_3^{s_3}$ is eliminated by acausality $S_3^{s_3}$, which implies applying $Z_1^{s_3}$ and $Z_4^{s_3}$; since $Z_1^{s_3}$ is acausal, it is eliminated by S_6 . Here, we define $g(3) = \{3, 4, 6, 7\}$, where $\text{Odd}(g(3)) = \{3, 5\}$; therefore, criteria **(G2)** in Definition 2.16 is replaced with

$$\textbf{(G2)} \quad i \in g(i) \Rightarrow i \in \text{Odd}(g(i)).$$

Now, consider the same graph, but the measurement of node 3 is done on the yz -plane, *e.g.*, with pattern $M_3^{yz, \theta}$. The following stabilizers implementation is done to achieve determinism.



For this case, we only need to eliminate acausality $X_3^{s_3}$ using another acausality $S_3^{s_3}$. The introduced acausality $Z_1^{s_3}$ from $S_3^{s_3}$ is eliminated with $S_6^{s_3}$. Thus, $g(3) = \{3, 6\}$, where $\text{Odd}(g(i)) = \{4, 5\}$. Thus, the criteria **(G2)** is replaced with

$$\textbf{(G2)} \quad i \in g(i) \Rightarrow i \notin \text{Odd}(g(i)).$$

It is straightforward to construct patterns analogous to [Theorem 2.2](#) for xz - and yz -planes measurements; it is also clear to see that those patterns have the same properties, *i.e.*, they are deterministic for all θ_i and any signals s_i , and stepwise deterministic. The stepwise deterministic property allows us to compose a pattern composed of different measurements straightforwardly. Let λ be a variable carrying information of the plane of each measurement, *i.e.*, $\lambda_i \in \{XY, XZ, YZ\}$; the generalization of [Theorem 2.2](#) is the following.

Theorem 2.3. *[86] Suppose the open graph state $(\mathcal{G}, I, O, \lambda)$ has a gflow $(g, \succ, \lambda)$, then the pattern*

$$\mathfrak{P}_{g, \mathcal{G}} := \mathop{\mathop{\mathop{\prod}}\limits^{\succ}}_{i \in O^c} \left(X_{g(i) \setminus \{i\}}^{s_i} Z_{\text{Odd}(g(i)) \setminus \{i\}}^{s_i} M_i^{\lambda_i, \theta_i} \right) E_{\mathcal{G}} N_{I^c},$$

where the product follows the dependency order $\succ$, is deterministic for all θ_i and any signals s_i , stepwise deterministic, and realizes the isometry $(\prod_{i \in O^c} \langle +_{\theta_i}^{\lambda_i} |) E_{\mathcal{G}} N_{I^c}$.

2.5 LAZY 1WQC

The 1WQC scheme was initially introduced to perform quantum computations on ultracold atoms in an optical lattice. The resource state of 1WQC, which is a

highly entangled state, can be generated massively at once on an ultracold atom in an optical lattice system [66]; scalability is not a concern of this physical system. However, to individually control the qubit, to perform adaptive measurements, and to create the desired graph state with high fidelity remain a challenge. On the other hand, solid-state qubits are more promising in the individual qubit control, yet, scalability has been hard to date. Hence, we seek possibilities to adapt the 1WQC scheme on solid-state qubits, such that the number of physical qubits is minimal.

To overcome the scalability problem, we employ the idea of “reusing” qubits, which works on *qubits with quantum memory*. We define qubits with quantum memory as qubits that possess permanence and can be rapidly re-initialized. Such qubits are found in solid-state systems. Note that solid-state qubits cannot be available in unlimited quantities. Also, using solid-state qubits means eliminating 1WQC’s feature as a memoryless quantum computer; however, its simple requirements to achieve universal computation remain promising.

Previous work in generating graph state using solid-state qubits has been done in [54]. They found that, for a graph with n inputs, n outputs, and m nodes, given the graph has a flow, only $\min(n + 1, m)$ qubits are required; however, given the graph has only gflow, up to $m - 2$ qubits are required. Here, we extend their scheme; while they need a class of graphs with the same number of inputs and outputs, we relax the requirement to any graphs with a flow. Our main interest is implementing quantum algorithms on a 1WQC scheme, thus, we are only interested in graphs with flow. We call our scheme *lazy 1WQC*.¹³

Essentially, lazy 1WQC is a 1WQC computation scheme that allows one to prepare parts of the graph state as needed, which minimizes the required physical qubits. In this section, we provide a formal statement of the lazy 1WQC, an explicit scheme to run a quantum algorithm, a proof of the correctness, and a bound on the physical qubits needed for this scheme. For practicality, we heavily employ measurement calculus syntax, which is reviewed in [Section 2.3](#).

As mentioned beforehand, we are only interested in graphs with flow. We extend the 1WQC computation definition from [Definition 2.2](#) into [Definition 2.17](#), where a flow must be given. [Definition 2.17](#) also works for the lazy scheme. Note that the total order $>$ is arbitrary, but it must respect the partial order $\succ$, *i.e.*, partial ordering induced by the open graph’s flow.

¹³ The “lazy” name is inspired from a computation programming paradigm called *lazy evaluation*. Lazy evaluation means that the evaluation of an expression is delayed until the value is needed [87]

Definition 2.17. A 1WQC or lazy 1WQC computation is expressed as a set

$$\{(\mathcal{G}, I, O), f, \phi, \rho^{in}\} \quad \text{or} \quad \{(\mathcal{G}^>, I, O), f, \phi, \rho^{in}\},$$

where $(\mathcal{G}, I, O)$ is an open graph from [Definition 2.5](#) that has a flow $(f, \succ)$, ϕ is a set of measurement angles, and ρ^{in} is a quantum (input) state. State ρ^{in} is assigned to nodes I , nodes O are not measured, and total order $>$ is consistent with $\succ$.

Before going into the lazy 1WQC, let us consider running a quantum algorithm within the 1WQC scheme in [Algorithm 2.1](#). The scheme takes an input from [Definition 2.17](#) and outputs $\mathcal{E}(\rho)$, where $\mathcal{E}$ is a CPTP map with the isometry defined in [Theorem 2.1](#).

In line 7 of [Algorithm 2.1](#), we actively perform (implicit) Pauli corrections during measurements by absorbing them into the measurement angle ϕ' — see [Equation \(2.32\)](#) for the rules. Applying actual Pauli operators is only necessary for the final quantum output — indicated in the last line, where the tensor product $\otimes$ signifies concurrent corrections independently performed on each output qubit. If the outputs are classical (nodes O are measured at the end), the last line of [Algorithm 2.2](#) vanishes, because the loop in line 6 goes over V instead of O^c .

Algorithm 2.1 1WQC computation

Input: $\{(\mathcal{G}, I, O), f, \phi, \rho^{in}\}$

▷ see [Definition 2.17](#)

Output: $\mathcal{E}(\rho^{in})$

▷ see [Theorem 2.1](#)

Conventions:

(I) Partial order $\succ$ is induced by flow f .

(II) $z(i) := \bigoplus_{k \prec i, i \in N_{\mathcal{G}}(f(k))} s_k$, $invf(i) \equiv f^{-1}(i)$.

(III) $s_{invf(i)} = 0$ for all $i \in I$.

- 1: Assign ρ^{in} to the input nodes I .
 - 2: **for** $i \in I^c$ **do**
 - 3: assign state $|+\rangle$ to node i
 - 4: **end for**
 - 5: Apply entangling operations $E_{\mathcal{G}}$
 - 6: **for** $i \in O^c$ with order follows partial ordering $\succ$ **do**
 - 7: Measure i in $\phi'_i := (-1)^{s_{invf(i)}} \phi_i + z(i)\pi$, and obtain measurement outcome s_i
 - 8: **end for**
 - 9: Correct the outputs by applying $\bigotimes_{i \in O} X_i^{s_{invf(i)}} Z_i^{z(i)}$.
-

Now, by assembling the measurement commands from [Algorithm 2.1](#), we obtain the following pattern

$$\mathfrak{P}_{1wqc} := \bigotimes_{j \in O} X_j^{s_{\text{invf}(j)}} Z_j^{z(j)} \succ \prod_{i \in O^c} (M_i^{\phi_i} X_i^{s_{\text{invf}(i)}} Z_i^{z(i)}) E_{\mathcal{G}} N_{I^c}^0, \quad (2.40)$$

where the parameters $z(i)$ and $\text{invf}(i)$ are defined as

$$z(i) := \bigoplus_{k \prec i, i \in N_{\mathcal{G}}(f(k))} s_k \quad (2.41)$$

$$\text{invf}(i) \equiv f^{-1}(i), \quad (2.42)$$

where $\bigoplus$ is modulo 2 summation, and operator $\succ \prod$ signifies an ordered product with an ordering that is consistent with $\succ$.¹⁴

We claim that [Algorithm 2.1](#) is identical to the 1WQC scheme defined in [Theorem 2.1](#), but its expression has a different perspective from the 1WQC; thus, pattern $\mathfrak{P}_{1wqc}$ must be identical to the pattern from [Theorem 2.1](#), *i.e.*,

$$\mathfrak{P}_{1wqc} = \mathfrak{P}_{f, \mathcal{G}, \succ, \phi} = \succ \prod_{i \in O^c} (X_{f(i)}^{s_i} \prod_{k \in N_{\mathcal{G}}(f(i)) \setminus \{i\}} Z_k^{s_i} M_i^{\phi_i}) E_{\mathcal{G}} N_{I^c}^0. \quad (2.43)$$

We prove this equality in [Lemma 2.1](#).

The primary difference between both patterns is the presence of additional parameters $z(i)$ and $\text{invf}(i)$, that is initially introduced in [88]. Parameter $z(i)$ contains the information of Z -corrections applied to node i due to previous measurements; parameter $\text{invf}(i)$ contains the information of X -corrections applied to i due to measuring $\text{invf}(i)$. Consider [Example 2.11](#) that illustrates those parameters explicitly.

Lemma 2.1. *Suppose the open graph state $(\mathcal{G}, I, O)$ has flow $(f, \succ)$, then the following patterns $\mathfrak{P}_1, \mathfrak{P}_2$ are identical $\forall \phi_i$:*

$$\mathfrak{P}_1 := \succ \prod_{i \in O^c} (X_{f(i)}^{s_i} \prod_{k \in N_{\mathcal{G}}(f(i)) \setminus \{i\}} Z_k^{s_i} M_i^{\phi_i}) E_{\mathcal{G}} N_{I^c}^0 \quad (2.44)$$

$$\mathfrak{P}_2 := \bigotimes_{j \in O} X_j^{s_{\text{invf}(j)}} Z_j^{z(j)} \succ \prod_{i \in O^c} (M_i^{\phi_i} X_i^{s_{\text{invf}(i)}} Z_i^{z(i)}) E_{\mathcal{G}} N_{I^c}^0, \quad (2.45)$$

where $z(i) := \bigoplus_{k \prec i, i \in N_{\mathcal{G}}(f(k))} s_k$, $\text{invf}(i) \equiv f^{-1}(i)$, and $\text{invf}(i) = 0$ for all $i \in I$.

¹⁴ Recall that $\succ$ is a partial ordering; thus $\succ \prod$ has a total ordering that follows $\succ$.

Proof. Observe that both patterns are ordered in the following manner: (1) graph state preparation (qubits initialization and entangling operations), (2) adaptive measurements that follow the partial order $\succ$, and (3) applied Pauli corrections interspersed among the measurements. It is evident from inspections that the measurement operators $M_i^{\phi_i}$ occur in the same order in patterns $\mathfrak{P}_1$ and $\mathfrak{P}_2$. Thus, it only remains to prove that both patterns implement identical Pauli corrections.

Consider the X -corrections. After measuring i with output s_i , the X -correction propagates to qubit $f(i)$, namely $X_{f(i)}^{s_i}$ will appear before measuring $f(i)$, as seen in $\mathfrak{P}_1$. Equivalently,¹⁵ qubit $f(i)$ receives X -correction from measurement outcome of qubit i , where i is the preimage of $f(i)$. Thus, before measuring $f(i)$, the correction from measuring i must be present, namely $X_{f(i)}^{s_i}$. Let $f(i) =: j$, thus $X_{f(i)}^{s_i} = X_j^{s_{\text{inv}f(j)}}$, which is the corresponding operator in $\mathfrak{P}_2$.

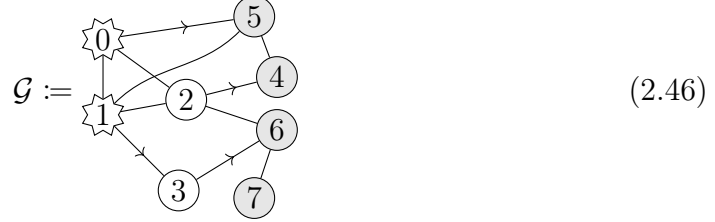
The same treatment applies to the Z -corrections; however, the corrections propagate to a subset of nodes instead of a single node. From pattern $\mathfrak{P}_1$, measuring i produces Z -corrections on nodes $\{N_{\mathcal{G}}(f(i)) \setminus \{i\}\} =: K$, which are the neighbors of $f(i)$. From the perspective of $k \in K$, qubit k receives corrections from all measurement outcomes of i , where the set of neighbors of $f(i)$ contains k . From the flow condition, we know that $i \prec k$, meaning that i has already been measured. Thus, the Z -correction of qubit k must include all measurement outcomes from qubits $\{i \prec k \mid k \in N_{\mathcal{G}}(f(i))\}$. That is what we see in the superscript of Z in pattern $\mathfrak{P}_2$.

Since every node is corrected by one Pauli operation, the correction of the output can be represented as an operator acting on the output qubits. Note that the output qubits lie in the last layer of the graph: for all $k \notin O$, for all $i \in O$, we have $k \prec i$; thus, the corrections can be shown at the very end of the pattern. $\square$

Remark that we distinguish the products of operators $\otimes$ and $\prod$, to indicate the following cases. Operator $\otimes$ always allows for concurrency, which is noticed from disjoint indices attached to every sequence of operators; whereas $\prod$, $\curlywedge$, or $\curlyvee$ allows for overlapping indices of the operators in every sequence.

¹⁵ Note that f is injective, which is shown in [Proposition 2.1](#).

Example 2.11. Here, we compute the values of $z(i)$ and $\text{invf}(i)$, given graph $\mathcal{G}$ shown below, where the flow is indicated by arrows. The flow induces partial ordering $\{0\} \prec \{1\} \prec \{3\} \prec \{2\} \prec \{4, 5, 6, 7\}$.



From the figure, we know $f(0) = 5, f(2) = 4, f(1) = 3$, and $f(3) = 6$, thus we have preimages $\text{invf}(5) = 0, \text{invf}(4) = 2, \text{invf}(3) = 1$, and $\text{invf}(6) = 3$.

Let K_i signify a set of qubits where Z corrections appear after measuring i , e.g., $K_i := N_{\mathcal{G}}(f(i)) \setminus \{i\}$; then we have $K_0 = \{1, 4\}, K_1 = \{6\}, K_3 = \{2, 7\}$, and $K_2 = \{5\}$. Thus, $z(0) = \emptyset, z(1) = \{0\}, z(3) = \emptyset, z(2) = \{3\}, z(4) = \{0\}, z(5) = \{2\}, z(6) = \{1\}$, and $z(7) = \{3\}$.

In this work, we will mainly use our point of view of “correcting the qubit before measuring it” in expressing patterns. Although it looks more complex, it allows a direct way in proving correctness for our lazy 1WQC scheme, and it provides simplicity in the code writing for simulations.

The lazy 1WQC scheme is formalized in [Algorithm 2.2](#). The algorithm takes inputs with the format given in [Definition 2.17](#), i.e., $\{(\mathcal{G}^>, I, O), f, \phi, \rho^{\text{in}}\}$. It also gives the same output $(\mathcal{E}(\rho^{\text{in}}))$ as the 1WQC scheme, which is proven in [Theorem 2.4](#). Note that we include the total order $>$ in the input for a physical reason: any two valid total orderings will result in the same computation and the same requirements on physical qubit, but they might require different coherence time on each qubit. In the following, we will see how we come to this conclusion.

Several new notations appear in [Algorithm 2.2](#) such as $E_{iN_{\mathcal{G}}(i)}^>, E_O$, and $A(i)$; they convey the following notions. Given a graph $\mathcal{G}$ with total ordering $>$, for any subgraph $\mathcal{G}[K]$, where $K \subset V$,

$$E_K := \prod_{(i,j) \in E(K)} E_{ij}, \quad E_{iK} := \prod_{k \in K} E_{ik} \text{ where } i \notin K, \quad E_{iK}^> := \prod_{k \in K, k > i} E_{ik}, \quad (2.47)$$

where E_{ij} is the entangling command from measurement calculus (CPHASE gate); also, E_{ij} are all mutually commuting. Parameter $A(i)$ signifies the allocation of free physical qubits at a time step i , we define it as

$$A(i) := N_{\mathcal{G}}[i] \setminus (I \cup_{j < i} N_{\mathcal{G}}[j]), \quad (2.48)$$

which is a *closed neighborhood*, excluding the ones that have been assigned before; these qubits are then initialized in state $|+\rangle$. A closed neighborhood of vertex i in graph $\mathcal{G}$ is defined as

$$N_{\mathcal{G}}[i] := \{k \in V(\mathcal{G}) \mid \{i, k\} \in E(\mathcal{G})\} \cup \{i\}, \quad (2.49)$$

which is the open neighborhood of i that includes i . Notice that, $A(i)$ always excludes I , because input nodes I are assigned with a quantum input state ρ^{in} beforehand.

If we compare [Algorithm 2.1](#) and [Algorithm 2.2](#), the difference can be recognized by comparing the loops; the 1WQC scheme has separated preparation and measurement loops, whereas the lazy 1WQC scheme mingles those loops into a two-nested

Algorithm 2.2 Lazy 1WQC computation

Input: $\{(\mathcal{G}^>, I, O), f, \phi, \rho^{in}\}$

▷ see [Definition 2.17](#)

Output: $\mathcal{E}(\rho^{in})$

▷ see [Theorem 2.4](#)

Conventions:

(I) Partial order $\succ$ is induced by flow f .

(II) $z(i) := \bigoplus_{k < i, i \in N_{\mathcal{G}}(f(k))} s_k$, $invf(i) \equiv f^{-1}(i)$.

(III) $s_{invf(i)} = 0$ for all $i \in I$.

- 1: Assign ρ^{in} to the input nodes I .
 - 2: **for** $i \in V$ with ordering $>$ **do**
 - 3: **for** $k \in A(i)$ **do** ▷ see [Equation \(2.48\)](#)
 - 4: assign state $|+\rangle$ to node k
 - 5: **end for**
 - 6: Apply entangling operations $E_{iN_{\mathcal{G}}(i)}^>$
 - 7: **if** $i \in O^c$ **then**
 - 8: $\phi'_i := (-1)^{s_{invf(i)}} \phi_i + z(i)\pi$
 - 9: Measure i in basis $|\pm_{\phi'}\rangle$ and obtain measurement outcome s_i .
 - 10: **else**
 - 11: Correct output i applying $X_i^{s_{invf(i)}} Z_i^{z(i)}$.
 - 12: **end if**
 - 13: **end for**
-

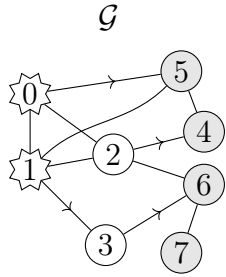
loop. This distinction connects to getting away from the 1WQC feature as a memoryless quantum computer.

The lazy 1WQC scheme allows one to prepare fewer physical qubits than neighbors of the qubits that are about to be measured. The preparation here indicates assigning $|+\rangle$ to “fresh qubits”. The required number of fresh qubits at step i in the loop is captured in $A(i)$. Additionally, we also have a term “active qubits” at time step i , that indicates the qubits that are present during measuring qubit i . Thus, there are fewer fresh qubits than active qubits at a time. Consider [Example 2.12](#) for an illustration.

Example 2.12. *The fresh qubits $A(i)$ for all $i \in V(\mathcal{G})$, where $\mathcal{G}$ is defined in [Example 2.11](#), are shown below.*

Remark graph $\mathcal{G}$ has a partial ordering $\{0\} \prec \{1\} \prec \{3\} \prec \{2\} \prec \{4, 5, 6, 7\}$, thus, for the non-output qubits, we may define a total ordering as $0 < 1 < 3 < 2 < 4 < 5 < 6 < 7$. From the picture, $I = \{0, 1\}$ and $O = \{4, 5, 6, 7\}$. It is easy to tabulate the answers as follows.

$i(\text{sorted})$	$N_{\mathcal{G}}[i]$	$A(i) := N_{\mathcal{G}}[i] \setminus (I \cup_{j < i} N_{\mathcal{G}}[j])$
0	$\{0, 1, 2, 5\}$	$\{2, 5\}$
1	$\{0, 1, 2, 3\}$	$\{3\}$
3	$\{1, 3, 6\}$	$\{6\}$
2	$\{0, 1, 2, 4, 6\}$	$\{4\}$
4	$\{2, 4, 5\}$	$\emptyset$
5	$\{0, 4, 5\}$	$\emptyset$
6	$\{2, 6, 7\}$	$\{7\}$
7	$\{6\}$	$\emptyset$



From here, we prove that the lazy 1WQC behaves as expected; namely, it results in the same computation as if it runs in the 1WQC scheme. Our primary strategy is investigating the resulting measurement patterns.

Firstly, we write the resulting pattern from [Algorithm 2.2](#) by consecutively placing the measurement commands.

$$\mathfrak{P}_{\text{lazy}} = \bigotimes_{j \in O} (X_j^{\text{inv}f(j)} Z_j^{z(j)}) E_O \mathop{\mathbb{H}}\limits_{i \in O^c} M_i^{\phi_i} X_i^{\text{inv}f(i)} Z_i^{z(i)} E_{iN_{\mathcal{G}}(i)}^> N_{A(i)}^0, \quad (2.50)$$

where $z(i) := \bigoplus_{k < i, i \in N_G(f(k))} s_k$, $\text{inv}f(i) = 0 \ \forall i \in I$, and we apply the measurement correction relation from Equation (2.32) to obtain $M_i^{(-1)^{s_{\text{inv}f(i)}\phi_i + z(i)\pi}} = M_i^{\phi_i} X_i^{\text{inv}f(i)} Z_i^{z(i)}$.

Secondly, consider the following lemmas, which will support Theorem 2.4.

Lemma 2.2. *Suppose the open graph state $(\mathcal{G}^>, I, O)$ has flow $(f, \succ)$, then $A(i)$ contains at least $f(i)$ for all $i \in O^c$ and $A(i) \cap A(j) = \emptyset$ for all $i \neq j$.*

Proof. We need to prove that $f(i) \in A(i)$ and $A(i) \cap A(k) = \emptyset, k \neq i$, for all $i, k \in O^c$.

Proof of the first part: Using the existence of flow $f : O^c \rightarrow I^c$, thus $f(i) =: j$ exists (where $j \in I^c$), and Equation (2.33) applies. It follows that $j \in N_G(i)$, and also $j \in N_G[i]$, where $j \succ i$. But $j \notin \cup_{k < i} N_G[k]$, because the flow condition $k \in N_G(j) \setminus \{i\}, k \succ i$ (it is also true that $k > i$);¹⁶ then clearly $j \notin I$. Thus, $A(i)$ contains at least $f(i)$.

Proof of the second part: Denote $a_i := N_G[i]$, $b_i := I \cup_{j < i} N_G[j]$, thus $A(i) \equiv a_i \setminus b_i$. Given that every node has an ordering, consider the case $i < k$:

$$A(i) \cap A(k) \equiv (a_i \setminus b_i) \cap (a_k \setminus b_k) = [(a_i \setminus b_i) \setminus b_k] \cap a_k \stackrel{b_i \subseteq b_k}{=} (a_i \setminus b_k) \cap a_k \stackrel{a_i \subseteq b_k}{=} \emptyset.$$

It is clear that $b_i \subseteq b_k$ and $a_i \subseteq b_k$, since $i < k$. □

Lemma 2.2 confirms that we will only initialize every qubit $i \in I^c$ once and at least one fresh qubit will be initialized at a time step i , *viz.*, $f(i)$. For an illustration see the table in Example 2.12, which clearly shows that every set $A(i)$ is disjoint to all $A(j)$ for $i \neq j$ and contains at least $f(i)$. Also, we can directly observe in the table that the union of all $A(i)$ results in the non-input sets, *viz.*, $\bigcup_{i \in O^c} A(i) = I^c$; the formal statement is given in Lemma 2.3.

Lemma 2.3. *Suppose the open graph state $(\mathcal{G}^>, I, O)$ has flow $(f, \succ)$ and a proper total order $>$, then $\bigcup_{i \in V} A(i) = I^c$.*

Proof. Let $\mathcal{G} = (V, E)$. Recall that $\bigcup_{i \in V} A(i) = I^c \iff \bigcup_{i \in V} A(i) \subseteq I^c$ and $\bigcup_{i \in V} A(i) \supseteq I^c$; we assume the contrary, that is, let $\bigcup_{i \in V} A(i) = S$, thus $S \neq I^c$, *i.e.*, either $S \subsetneq I^c$ or $S \supsetneq I^c$ is true.

¹⁶ The flow must not have a neighbour in the past, *i.e.*, a neighbour that has already measured.

Consider the first case, where $S \subsetneq I^c$. Here, there exists $k \in I^c$, such that $k \notin A(i)$ for all $i \in I^c$. We split this first case proof into two parts: for $i \in O^c$ and for $i \in O$.

For $i \in O^c$, by using [Lemma 2.2](#), $A(i)$ contains at least $f(i)$. Note that $f : O^c \rightarrow I^c$; thus, $\text{inv}f(k) =: j$ exists, where $j \in O^c$. Therefore, $k \in A(j)$. Moreover, from [Lemma 2.2](#) we know that $S \neq \emptyset$.

For $i \in O$, we have $A(i) \subset O$ because if $A(i)$ contains an element in O^c , it is covered in the case $i \in O^c$ above. As output nodes also have a total ordering, $\cup_{i \in O} A(i)$ covers all output nodes that are disjoint to O^c nodes.

We arrive at a contradiction, so the assumption is incorrect, and we conclude that $I_c \setminus S = \emptyset$ or $I_c \subseteq S$.

Consider the second case, where $S \supsetneq I^c$. Here, there exists a $k \in I^c$, such that $A(k)$ contains m where $m \notin I^c$. By definition, $A(k)$ contains at most its closed neighborhood excluding the inputs, *viz.*, $N_{\mathcal{G}}(k) \setminus I$. Thus, $A(k)$ must be in the graph, since $\mathcal{G}[N_{\mathcal{G}}(k)]$ is a subgraph of $\mathcal{G}$; also, any element of I cannot be in $A(k)$. We again arrive at a contradiction, and must conclude that $S \setminus I_c = \emptyset$ or $S \subseteq I_c$.

Since $S \subseteq I^c$ and $I^c \subseteq S$, it follows that $S = I^c$. $\square$

[Lemma 2.3](#) shows that the total number of initialized fresh qubits during the computation run are given by all the non-input qubits I^c ; this is expected since the input qubits are assigned with input state ρ^{in} beforehand.

By distributing qubits preparation in such a manner, the entanglement commands need to be prepared accordingly. At a time-step i , as the preparation commands apply to qubits $A(i)$, the entanglement commands apply to $E_{iN_{\mathcal{G}}(i)}^>$ that is defined in [Equation \(2.47\)](#). To obtain the correct graph state, the total entanglement commands $E_{iN_{\mathcal{G}}(i)}^>$ for all i must result in the full set of edges $E(\mathcal{G})$; we prove this in [Lemma 2.4](#).

Lemma 2.4. *Suppose an open graph state $(\mathcal{G}^>, I, O)$ has flow $(f, \succ)$ and a proper total order $\succ$, then $\prod_{i \in V} E_{iN_{\mathcal{G}}(i)}^> = E_{\mathcal{G}}$, where $E_{iN_{\mathcal{G}}(i)}^> := \prod_{k \in N_{\mathcal{G}}(i), k > i} E_{ik}$.*

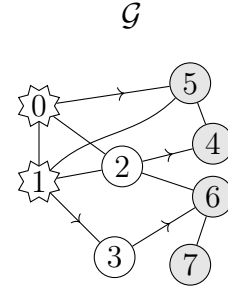
Proof. Note that we are employing the notation of [Equation \(2.47\)](#). Denote (i, j) as an edge in $E(\mathcal{G})$ with end nodes i and j . If we represent $\prod_{i \in V} E_{iN_{\mathcal{G}}(i)}^>$ as a collection of edges, we obtain $\bigcup_{i \in V} \{(i, k) \mid k \in N_{\mathcal{G}}(i), k > i\} =: S$. On the other hand, the Handshaking lemma [\[89\]](#) implies that, $\biguplus_{i \in V} \{(i, k) \mid k \in N_{\mathcal{G}}(i)\}$ will result in a multiset that contains double copies of edges — $\forall a \in E(\mathcal{G})$ the frequency

of a is 2, where $\uplus$ signifies the union of multisets [90]. Thus restricting to edges (i, k) where $i < k$ eliminates the double counting; therefore $S = E(\mathcal{G})$. $\square$

The Handshaking lemma states that $\sum_{v \in V(\mathcal{G})} \text{degree}(v) = 2|E(\mathcal{G})|$.¹⁷ Consider [Example 2.13](#) that illustrates [Lemma 2.4](#).

Example 2.13. Here, we show that [Lemma 2.4](#) is true for the graph $\mathcal{G}$ defined in [Example 2.11](#). Recall from [Example 2.12](#) that one may define the total ordering as $0 < 1 < 3 < 2 < 4 < 5 < 6 < 7$. Let us tabulate the entangling operations performed for parameters $i \in V(\mathcal{G})$.

$i(\text{sorted})$	$N_{\mathcal{G}}(i)$	$E_{iN_{\mathcal{G}}(i)}^> := \prod_{k \in N_{\mathcal{G}}(i), k > i} E_{ik}$
0	$\{1, 2, 5\}$	$\{(0, 1), (0, 2), (0, 5)\}$
1	$\{0, 2, 3, 5\}$	$\{(1, 3), (1, 2), (1, 5)\}$
3	$\{1, 6\}$	$\{(3, 6)\}$
2	$\{0, 1, 4, 6\}$	$\{(2, 4), (2, 6)\}$
4	$\{2, 5\}$	$\{(4, 5)\}$
5	$\{0, 4\}$	$\emptyset$
6	$\{3, 2, 7\}$	$\emptyset$
7	$\{6\}$	$\{(7, 6)\}$



From the figure on the right, we know that $\cup_{i \in V(\mathcal{G})} E_{iN_{\mathcal{G}}(i)}^> = \{(0, 1), (0, 2), (0, 5), (1, 3), (1, 2), (1, 5), (3, 6), (2, 4), (2, 6), (4, 5), (6, 7)\}$. Obviously, the union of all sets in the third column above results in $E(\mathcal{G})$.

We now have all the lemmas needed to state that the lazy 1WQC scheme ([Algorithm 2.2](#)) is correct, i.e., it produces the same pattern as the 1WQC scheme ([Algorithm 2.1](#)); see [Theorem 2.4](#).

Theorem 2.4. *The lazy 1WQC scheme and the 1WQC scheme implement the same map, they produce the same output for the same input.*

Proof. Let $\{(\mathcal{G}^>, I, O), f, \phi, \rho^{in}\} := \mathcal{J}$ be the input of the lazy scheme ([Algorithm 2.2](#)), where $>$ is consistent with the flow $(f, \succ)$, i.e., the flow of the open graph $(\mathcal{G}, I, O)$. Since $>$ is consistent with $\succ$, $\mathcal{J}$ is a valid input of the 1WQC scheme.

¹⁷ See [Definition 2.3](#) for the terms.

One can prove the map equality by comparing the patterns, *e.g.*, we will show that $\mathfrak{P}_{1wqc}$ in Equation (2.43) can be reduced to pattern $\mathfrak{P}_{lazy}$ in Equation (2.50). With $z(i)$ defined in Equation (2.41) and $A(i)$ defined in Equation (2.48), we have

$$\mathfrak{P}_{1wqc} = \mathop{\mathbb{H}}^{\succ}(X_{f(i)}^{s_i} \prod_{k \in N_G(f(i)) \setminus \{i\}} Z_i^{s_i} M_i^{\phi_i}) E_G N_{I^c}^0 \quad (2.51)$$

$$\stackrel{\text{Lem. 2.1}}{=} \prod_{j \in O} X_j^{s_{invf(j)}} Z_j^{z(j)} \mathop{\mathbb{H}}^{\succ}(M_i^{\phi_i} X_i^{s_{invf(i)}} Z_i^{z(i)}) E_G N_{I^c}^0 \quad (2.52)$$

$$\stackrel{\text{Lems. 2.3, 2.4}}{=} \prod_{j \in O} X_j^{s_{invf(j)}} Z_j^{z(j)} \mathop{\mathbb{H}}^{\succ}(M_i^{\phi_i} X_i^{s_{invf(i)}} Z_i^{z(i)}) E_O \mathop{\mathbb{H}}^{\succ}_{k \in O^c} E_{kN_G(k)}^{\succ} \mathop{\mathbb{H}}^{\succ}_{l \in O^c} N_{A(l)}^0 \quad (2.53)$$

$$= \bigotimes_{j \in O} X_j^{s_{invf(j)}} Z_j^{z(j)} \mathop{\mathbb{H}}^{\succ}(M_i^{\phi_i} X_i^{s_{invf(i)}} Z_i^{z(i)}) E_O \mathop{\mathbb{H}}^{\succ}_{k \in O^c} E_{kN_G(k)}^{\succ} \mathop{\mathbb{H}}^{\succ}_{l \in O^c} N_{A(l)}^0. \quad (2.54)$$

Note that in the third equality, the partial ordering $\succ$ is replaced with the total ordering $>$; this is valid since $>$ is consistent with $\succ$.

Now we need to commute the entangling and preparation operators such that they are distributed according to the lazy scheme. First, consider any two nodes i and k , where $i, k \in O^c$ and $i < k$. The preparation and entangling operators are commuting, *i.e.*,

$$E_{kN_G(k)}^{\succ} E_{iN_G(i)}^{\succ} N_{A(k)}^0 N_{A(i)}^0 = E_{kN_G(k)}^{\succ} N_{A(k)}^0 E_{iN_G(i)}^{\succ} N_{A(i)}^0. \quad (2.55)$$

However, we need to check if the condition of causality holds: there is no entanglement operation involving qubits that are already measured or not yet created. Denote the set of edges $e(i) := \{(i, k) \mid k \in N_G(i), k > i\}$, *i.e.*, edges that correspond to entangling operations $E_{iN_G(i)}^{\succ}$. By definition, set $e(i)$ does not contain any node that has already been measured, namely any $k < i$. The nodes that correspond to edges $e(i)$ are

$$\{k \in N_G[i] \mid k > i\} =: n(i). \quad (2.56)$$

By definition, $A(i)$ contains all nodes in $N_G[i]$ minus the ones that have already been created $I \cup_{k < i} N_G[k]$, thus $\forall x \in e(i), x \in \{(i, j) \mid j \in \cup_{j \leq i} A(j)\}$, which means every qubit connected by an edge in $e(i)$ is already initialized. Thus, there is no entanglement involving a qubit that has not yet been created. Therefore, Equation (2.55) is causal.

Considering the measurement operator and the Pauli corrections, we need to commute the entangling operation through them, namely

$$M_i^\phi X_i Z_i E_{kN_G(k)}^> N_{A(k)}^0 E_{iN_G(i)}^> N_{A(i)}^0 = E_{kN_G(k)}^> N_{A(k)}^0 M_i^\phi X_i Z_i E_{iN_G(i)}^> N_{A(i)}^0, \quad (2.57)$$

which is true, if and only if $i \notin n(k)$. By definition of $n(k)$ — see [Equation \(2.56\)](#), $i < k$, thus, $i \notin n(k)$. Thus, we can distribute the entangling and preparation operators in [Equation \(2.51\)](#) with respect to the ordering $>$ and obtain

$$\bigotimes_{j \in O} (X_j^{s_{\text{invf}}(j)} Z_j^{z(j)}) E_O^> \prod_{i \in O^c} M_i^\phi X_i^{s_{\text{invf}}(i)} Z_i^{z(i)} E_{iN_G(i)}^> N_{A(i)}^0 = \mathfrak{P}_{\text{lazy}}. \quad (2.58)$$

□

So far, we only considered computations with quantum inputs and outputs, since it is a requirement for our formalisms. Quantum outputs are required to obtain a pattern, which is formulated as pattern criteria **(D2)** and **(D3)** in [Section 2.3](#), *i.e.*, no command acts on a qubit not yet prepared and output qubits are not measured. However, one can undoubtedly measure the outputs qubits, obtaining classical outputs. Similarly, a quantum input is assumed in [Theorem 2.1](#), as it is assigned in the input nodes beforehand; yet one can still have classical inputs by encoding it to a quantum state, *e.g.*, bit 0 is encoded to $|0\rangle$ and bit 1 is encoded to $|1\rangle$. Within such a definition, it is straightforward to prove that the causality and determinism properties are intact, given classical inputs and outputs. However, does it give a particular advantage to the lazy 1WQC scheme, such as reducing the number of physical qubits? In our observation, it does not, in that sense, but in the physical realization, postponing the creation of input states will reduce the required coherence time.

What changes in the lazy 1WQC scheme from [Algorithm 2.2](#) when we have classical inputs and outputs? First, [Algorithm 2.2](#) assigns input qubit ρ^{in} initially — see [Algorithm 2.2](#). For classical inputs, ρ^{in} is in the form of product states, thus, we can prepare it together as a preparation commands (N_i^θ) implicitly,¹⁸ where bit 0 is prepared with command N_i^0 and bit 1 is prepared with command N_i^π . As it can be described with the preparation commands, we capture the input states preparation in parameter $A(i)$ defined in [Equation \(2.48\)](#). Thus, if classical inputs are expected, eliminating assignment of ρ^{in} in line [Algorithm 2.1](#) and setting $I = \emptyset$ in the definition of $A(i)$ will do the trick.

¹⁸ We call it an implicit preparation because we consider it as input state ρ^{in} , which are not written explicitly as a series of preparation commands.

As the number of physical qubits is the pivot of the lazy 1WQC, it is natural to question the bounds of the physical qubits given an arbitrary 1WQC computation; we provide our expectation in [Conjecture 2.1](#).

Conjecture 2.1. *The number of physical qubits required to run lazy 1WQC in [Algorithm 2.2](#), regardless the input and output type — whether classical or quantum — is bounded by $|O| + 1$.*

The intuition behind [Conjecture 2.1](#) stems from a property of a graph with flow; namely, the number of nodes per layer can not shrink. It is due to non-colliding correction nodes: two distinct nodes i, j can not have the same X -correction node, $f(i) \neq f(j)$; otherwise, it violates the flow criteria — see [Proposition 2.1](#).

Proposition 2.1. *Suppose i and j are two distinct vertices in an open graph $(\mathcal{G}, I, O)$ with flow $(f, \succ)$, then $f(i) \neq f(j)$.*

Proof. Let us assume the contrary, there exists three distinct nodes $i, j, k \in \mathcal{G}$ such that $f(i) = f(j) = k$. Given that flow f induces a partial ordering $\succ$; by the definition of flow, every criterion in [Definition 2.15](#) must be satisfied, *i.e.*,

$$\text{(F0)} \quad (k, i) \in E(\mathcal{G}) \text{ and } (k, j) \in E(\mathcal{G})$$

$$\text{(F1)} \quad k \succ i \text{ and } k \succ j$$

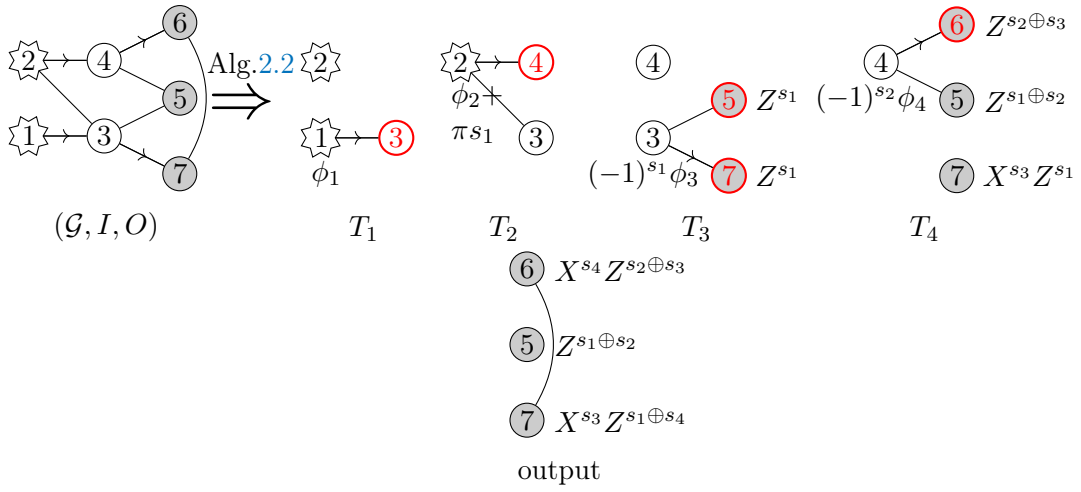
$$\text{(F2)} \quad \forall v \in N_{\mathcal{G}}(k) \setminus \{i\}, v \succ i \text{ and } \forall w \in N_{\mathcal{G}}(k) \setminus \{j\}, w \succ k.$$

The first criterion of **(F2)** entails $j \in N_{\mathcal{G}}(k) \setminus \{i\}$ since $j \in N_{\mathcal{G}}(k)$ by criterion **(F0)**, which implies $j \succ i$; hence, **(F1)** imposes an ordering $k \succ j \succ i$. However, the second criterion of **(F2)** also implies that $i \in N_{\mathcal{G}}(k) \setminus \{j\}$, because $i \in N_{\mathcal{G}}(k)$ — **(F0)**; thus, $i \succ j$, which leads to a contradiction. $\square$

The bound in [Conjecture 2.1](#) is rather tight. From our simulations, we find that most cases have tight bounds $|O| + 1$, otherwise $|O|$. A loose bound occurs on graphs with disjoint output nodes to non-output nodes. To increase the confidence in [Conjecture 2.1](#), we simulate the lazy 1WQC on all possible graphs with flow up to order 8 — our simulation is elaborated in [Section A.1.1](#). From our simulations, we find that, if there is an output node that is disjoint from all vertices $i \in O^c$, the number of physical qubits needed is $|O|$. Consider examples that illustrate the tight bound case in [Example 2.14](#) and the loose bound case in [Example 2.15](#). More examples of the loose bound cases are shown in [Section A.1.2](#).

Example 2.14. Here, we describe the process of a lazy 1WQC computation explicitly. Given the computation is $\{(\mathcal{G}^>, I, O), f, \phi, \rho^{in}\}$, where $\mathcal{G} = (V, E)$ for $V = \{1, 2, 3, 4, 5, 6, 7\}$ and $E = \{(1, 3), (2, 3), (2, 4), (4, 6), (4, 5), (3, 5), (3, 7)\}$; state ρ^{in} is assigned to input nodes $I = \{1, 2\}$, and output nodes $O = \{5, 6, 7\}$. We will see that $\mathcal{G}$ has a tight bound from [Conjecture 2.1](#).

The following figure shows the formed subgraph for every time T_i together with measurement angles and the Pauli corrections, where i indicates the measured vertex. Red signifies the initialized fresh qubits. The input state ρ^{in} is assigned to vertices $\{1, 2\}$ beforehand.

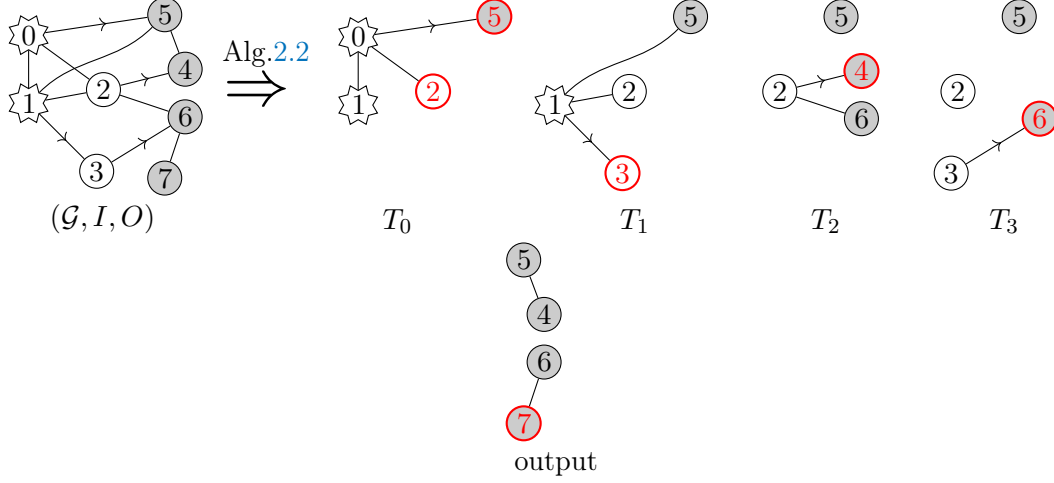


We can see that at most four physical qubits are needed, which is $|O| + 1$, and it occurs at T_3 and T_4 . The defined flow in the figure induces a partial ordering $\{1\} \prec \{2\} \prec \{3, 4\} \prec \{5, 6, 7\}$; one can define a total ordering $1 < 2 < 3 < 4 < 5 < 6 < 7$. If the inputs are classical, at time T_1 , node 2 is removed and is assigned at T_2 instead. If the outputs are classical, output nodes can be measured in an arbitrary order since all $i \in O$ are incomparable; however, since initially we define the ordering $<$, output nodes are measured with order 5, 6, then 7.

Example 2.15. Here, we describe the process of another lazy 1WQC computation explicitly. Given a computation $\{(\mathcal{G}^>, I, O), f, \phi, \rho^{in}\}$, where $(\mathcal{G}, I, O)$ is the graph from [Example 2.13](#). We will see that $\mathcal{G}$ has a loose version of the bound from [Conjecture 2.1](#).

From [Example 2.13](#), we may set a total ordering as $0 < 1 < 3 < 2 < 4 < 5 < 6 < 7$. Thus, let ρ^{in} be assigned to nodes I beforehand, and by using the same notation

as in [Example 2.14](#), we obtain the following subgraph for every measurement i at time step T_i ,



with measurement angles $(-1)^{\text{invf}(i)}\phi_i + z(i)\pi$ and output corrections $\bigotimes_{i \in O} X_i^{\text{invf}(i)} Z_i^{z(i)}$, where $\text{invf}(i)$ and $z(i)$ have been calculated in [Example 2.11](#). From the figures, at most four physical qubits are needed at a time. Even though classical inputs and outputs are expected, the number physical qubits needed remains four, which is $|O|$.

DELEGATED QUANTUM COMPUTATION AND PRIVACY

The omnipresence of the internet allows us to access any connected machines from arbitrary locations. Some machines are powerful, sophisticated, and equipped with high computation power, while others have small computation power. The internet allows us (users) to access powerful machines to perform a computation on our behalf; the powerful machine is called *server* and the user is called *client*. The client-server paradigm has been implemented in many aspects of our daily life for decades; this paradigm is well-known as *cloud computing*.

It is most likely that quantum computers will also play the same role as cloud computers, for its high maintenance, complexity, and remarkably high-price. Several quantum servers are already available today to carry out our quantum computations. For example, the companies and organizations that give public access to their platforms are IBM with 16 superconducting transmon qubits [91], Rigetti with 19 superconducting qubits [92], and University of Bristol with two photonic qubits [93].

Requesting a server to perform our computations gives rise to privacy issues. There are many maturely secure classical cryptographic protocols that have been implemented in our devices, carrying out tasks with the desired level of privacy. The same issues arise in cloud quantum computers; however, due to the nature of quantum systems, one can not merely implement a classical encryption scheme to achieve the same level of privacy. Instead, we study the nature of quantum systems and build a cryptographic scheme based on this nature.

The study of private delegated quantum computations was initiated by Childs in 2001 [94]. To date, various private delegated quantum computation protocols have been proposed and well studied. They can be broken down into two categories: *Blind Quantum Computation* (BQC) [25, 88, 95–104] and *Quantum Homomorphic Encryption* (QHE) [105–110]. BQC and QHE aim towards the same goal, *i.e.*, pri-

vately delegate quantum computations; however, BQC allows multiple rounds of interactions between a client and a server, whereas QHE allows only one round of interaction between them. While the requirement of many interactions limits the efficiency of BQC, it has an essential advantage over QHE: it does not rely on the assumption of the adversary’s computational power. On the other hand, a fully¹ QHE cannot achieve perfect privacy while consuming a non-exponential resource [111].

Our work falls into category BQC. In particular, it is an extension of a BQC protocol, so-called *Universal Blind Quantum Computation* (UBQC). Here, we discuss the UBQC, followed by a brief discussion on *Abstract Cryptography* (AC), a security framework that models cryptographic protocols.

3.1 UNIVERSAL BLIND QUANTUM COMPUTATION

The Universal Blind Quantum Computation (UBQC) is a delegated quantum computation protocol introduced by Broadbent, Fitzsimons, and Kashefi [25] that allows a client (Alice) to delegate her quantum computation to a powerful quantum server (Bob), where Alice’s computation is in the form of One-Way Quantum Computer (1WQC) model. The protocol promises blindness as the security flavor, which means that Bob cannot learn about Alice’s computation nor her measurement outcomes.

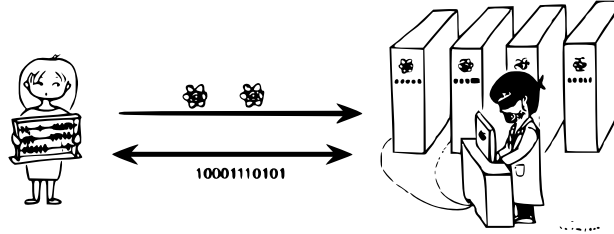


Figure 3.1: A UBQC illustration taken from Fitzsimons’ talk [112]; The client (Alice), shown on the left, delegates her computation to the server (Bob), shown on the right. Alice’s quantum power is very small, which is portrayed by an abacus; on the contrary, Bob has a powerful quantum computer, which is portrayed by a stack of servers. Alice and Bob interact using quantum channels and classical channels, which are shown by the arrow with a “quantum picture” and a binary number, respectively. Bob’s blindfold portrays blindness in the UBQC.

¹ Fully quantum homomorphic encryption means that it can delegate universal computation.

Protocol 3.1 [25, 88] Universal Blind Quantum Computation

Alice's input: $\{(\mathcal{G}, I, O), f, \phi, \rho_{\mathcal{A}}^{in}\}$ ▷ see [Definition 2.17](#)

Conventions:

(I) $\Omega = \{\frac{n\pi}{4}\}_{0 \leq n \leq 7}$ and $\phi_i \in \Omega$.

(II) $invf(i) \equiv f^{-1}(i)$, $z(i) := \bigoplus_{k \prec i, i \in N_{\mathcal{G}}(f(k))} s_k$, and $t(i) := \bigoplus_{k \in I, i \in N_{\mathcal{G}}(k)} t_k$.

(III) $s_{invf(i)} = 0$ for all $i \in I$ and $t_i = 0$ for all $i \in I^c$.

1 State preparation

- 1: **for** $i \in I$ **do**
- 2: Alice chooses $\alpha_i \in \Omega$ and $t_i \in \{0, 1\}$ at random.
- 3: Alice applies $Z_i(\alpha_i)X_i^{t_i}$ to $\text{tr}_{I \setminus i}[\rho_{\mathcal{A}}^{in}]$.
- 4: Alice updates the angles :

$$\phi_i = (-1)^{t_i} \phi_i \tag{3.1}$$

$$\phi_k = \phi_k + t_k \pi, \forall k \in N_{\mathcal{G}}(i) \tag{3.2}$$

- 5: **end for**
- 6: Alice prepares $|+\alpha_i\rangle_i$, for $i \in V(\mathcal{G}) \setminus (I \cup O)$, where $\alpha_i \in \Omega$ is randomly chosen.
- 7: Alice transmits all qubits $i \in O^c$ to Bob.
- 8: Bob prepares $|+\rangle_i$, for all $i \in O$.
- 9: Bob entangles all qubits $i \in V(\mathcal{G})$ by entangling command $E_{\mathcal{G}}$ ([Equation \(2.47\)](#)).

2 Classical interaction and measurement

- 10: **for** $i \in V(\mathcal{G}) \setminus O$ which follows partial ordering $\succ$ **do**
- 11: Alice computes $\phi'_i = (-1)^{s_{invf(i)}} \phi_i + z(i)\pi$.
- 12: Alice computes $\delta_i := \phi'_i + \pi r_i + \alpha_i$, where $r_i \in \{0, 1\}$ is chosen at random.
- 13: Alice sends Bob δ_i .
- 14: Bob measures i in $|\pm_{\delta_i}\rangle$ basis, and tells Alice his measurement outcome $\tilde{s}_i$.
- 15: Alice sets $s_i = \tilde{s}_i \oplus r_i$.
- 16: **end for**

3 Output transmission and correction.

- 17: Bob sends Alice output qubits ρ_B^{out} (all qubits $i \in O$).
 - 18: Alice corrects the final output $\bigotimes_{i \in O} X_i^{s_{invf(i)} + t_i} Z_i^{z(i) + t(i)} (\rho_B^{out}) \equiv P(\rho_B^{out})$.
-

[Figure 3.1](#) illustrates a situation where Alice delegates her private quantum computation on Bob's powerful quantum computer via UBQC protocol defined in [Protocol 3.1](#). To be more precise, [Protocol 3.1](#) shows the general case when the input and

output are quantum, whereas Figure 3.1 illustrate the case for classical output;² quantum output requires two-way quantum channel. Remark that we define

$$Z(\alpha) := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}. \quad (3.3)$$

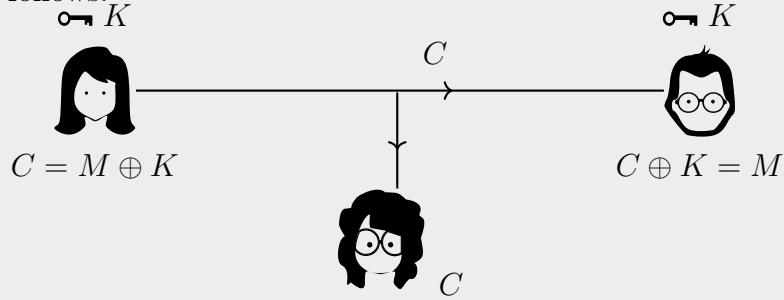
Protocol 3.1 is rewritten with a different notation from [25]; we combine our notation in Section 2.5 with the notation from [88].

The UBQC protocol comprises three parts that are denoted by ①, ②, and ③. Part ① is the graph state preparation. At this stage, Alice encrypts all of her qubits (O^c), then transmits them to Bob; her input qubits are fully encrypted using the *quantum one-time pad* — see the extra box in this section. When Bob receives all necessary qubits from Alice, he entangles all qubits accordingly, forming the desired graph state. Part ② comprises classical interactions. Alice informs Bob of measurement angles, which at the same time shuffles Bob’s measurement outcomes. Finally, in part ③, Bob sends Alice the final output on which she performs Pauli corrections.



One-time pad and quantum one-time pad

The one-time pad (OTP) is a cipher (encryption scheme) that is simple, yet unbreakable. Suppose Alice wants to securely send Bob a secret $M = m_n \dots m_2 m_1$, $m_i \in \{0, 1\}$, which is a binary string with length n . The OTP works as follows.



First, Alice (left) and Bob (right) initially hold a symmetric random key $K = k_n \dots k_2 k_1$, where $k_i \in \{0, 1\}$. Secondly, Alice encrypts her message $M \oplus K =: C$, where $\oplus$ is addition modulo 2, and the encrypted message C is called a *ciphertext*. Thirdly, Alice sends ciphertext C via an *authentic channel*, which is a channel that is resistant to tampering but not resistant

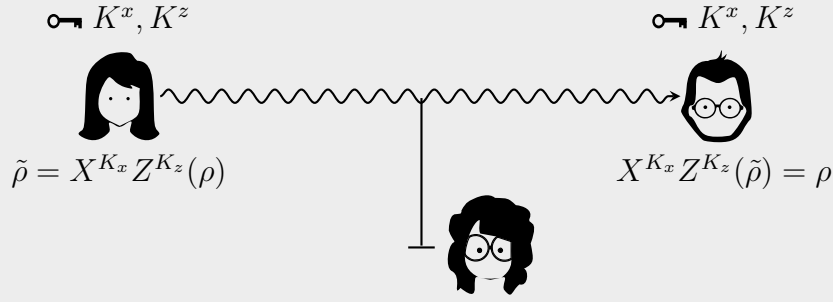
² The UBQC protocol with classical input and output is adjusted accordingly, analogous to Protocol 5.3.

to overhearing. At the same time, the malicious Eve (the middle party) is listening to the channel — portrayed by her receiving a copy of C . Finally, Bob obtains his message by deciphering his ciphertext

$$C \oplus K = M \oplus K \oplus K = M.$$

The OTP is private (we skip the proof), meaning that Eve cannot learn anything about the message apart from its length n , but she can have the copy of ciphertext.

The quantum one-time pad (QOTP) works similarly to the OTP, but now Alice's message is a quantum state ρ_A . Alice and Bob need two keys: $K^x = x_n \dots x_2 x_1$ for the bit flip and $K^z = z_n \dots z_2 z_1$ for the phase flip.



Like the OTP, Alice and Bob initially hold some symmetric keys: K^x and K^y . Then, Alice produces a quantum ciphertext $\tilde{\rho}$, and sends it to Bob via a (insecure) quantum channel. Finally, Bob decipheres his quantum ciphertext, and receives his quantum message

$$(X^{K_x} Z^{K_z})^\dagger(\tilde{\rho}) = Z^{K_z} X^{K_x}(X^{K_x} Z^{K_z}(\rho)) = \rho,$$

where $X^{k_x} := \bigotimes_{i=1}^n X_i^{x_i}$ and $Z^{k_z} := \bigotimes_{i=1}^n Z_i^{z_i}$. Due to the nature of quantum systems, Eve cannot retrieve any information (from the quantum ciphertext) with certainty without disturbing the quantum state. We give no further proof, but QOTP is not only private but also secure.

The UBQC scheme employs QOTP to encrypt Alice's input state ρ^{in} . Note that, in the UBQC, ρ^{in} is encrypted with z -rotations $Z(\alpha)$ instead of Pauli Z , where $\alpha \in [0, 2\pi)$ and $Z(\alpha)$ is defined in [Equation \(3.3\)](#). Thus, one needs to cancel the whole rotations as illustrated below.

Consider an encrypted quantum state

$$\tilde{\rho} = \bigotimes_{i=1}^n X_i^{x_i} Z(\alpha_i)(\rho),$$

where $\alpha_i \in [0, 2\pi)$ and $x_i \in \{0, 1\}$ are chosen at random. Bob retrieves his quantum message with operations

$$\bigotimes_{i=1}^n (X_i^{x_i} Z(\alpha_i))^\dagger (\tilde{\rho}) = \bigotimes_{i=1}^n Z_i(-\alpha_i) X_i^{x_i} X_i^{x_i} Z_i(\alpha_i)(\rho) = \rho.$$

The OTP is private and the QOTP is secure, but here is the catch. Firstly, the key must be as long as the message and must be established securely before the protocol starts. Second, a key cannot be used twice, otherwise it produces a correlation. For instance, in OTP, if we encrypt another message M_2 with the same key K , we obtain another ciphertext $M_2 \oplus K := C_2$; correlation $C \oplus C_2 = M \oplus K \oplus M_2 \oplus K = M \oplus M_2$ eliminates the randomness of K entirely.

The security of UBQC is *unconditionally blind* (this notion is elaborated in [Section 3.2](#)), which is formally proven in [\[113\]](#). Here, we do not aim for the formal security proof, but only give the intuition behind the blindness. In the following, we explicitly illustrate how the UBQC hides Alice's information by working on a simple example. This illustration is inspired by Fitzsimons' talk [\[112\]](#).

Consider Alice's computation in [Figure 3.2](#), where the corresponding measurement angles are written under the node and input state $|\psi\rangle$ is assigned to node 0. Assume that those angles cover the corrections; we do not need to worry about determinism here.

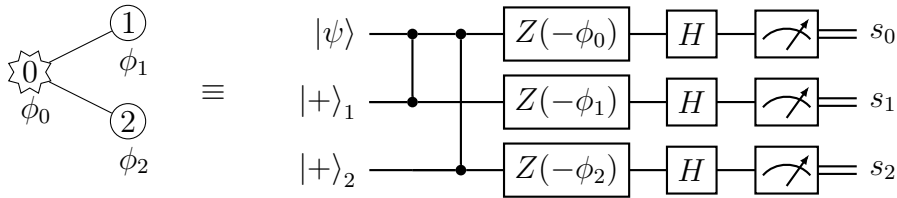
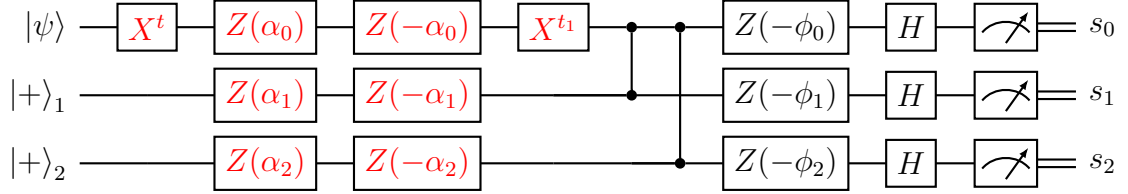
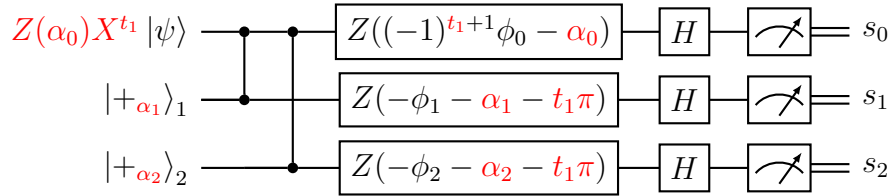


Figure 3.2: Alice's computation is shown on the left as a graph, where the adaptive measurement angles are shown below the nodes. The quantum circuit corresponds to the 1WQC computation on the left. The measurement follows ordering $0 < 1 < 2$.

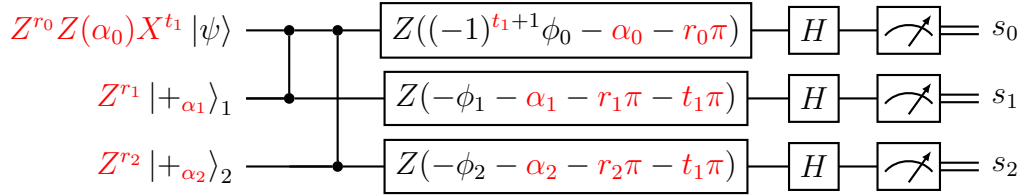
In part ① of Protocol 3.1, Alice introduces randomness α_i, t_i to her qubits. To keep the computation intact, she needs to apply the reverse operations. The following circuit is identical to Figure 3.2, where the red color indicates randomness.



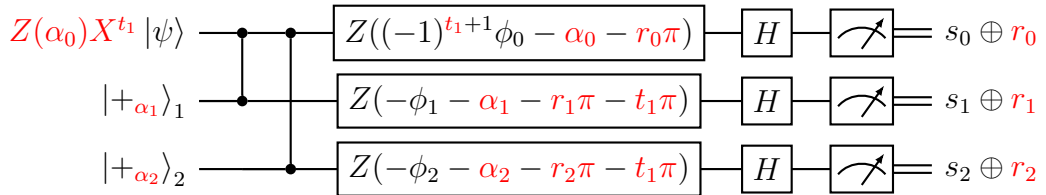
By using gate propagation $\text{CPHASE}_{ij}X_i = X_iZ_j\text{CPHASE}_{ij}$ and the commutation relation between z -rotations and CPHASE , we rewrite the circuit as follows.



To hide the measurement outcomes from Bob, Alice adds additional randomness r_i , which corresponds to applying Z^{r_i} .



Instead of applying actual random Z operators, Alice simply tracks random variable r_i ; this results in flipping measurement outcomes.



Alice wraps the measurement details into δ_i (step 12 of Protocol 3.1), and keeps the following information to herself: α_i, ϕ_i, t_i , and r_i . Thus, Bob knows only $\delta_i, \tilde{s}_i$, and $|\psi\rangle$ as shown in Figure 3.3 below.

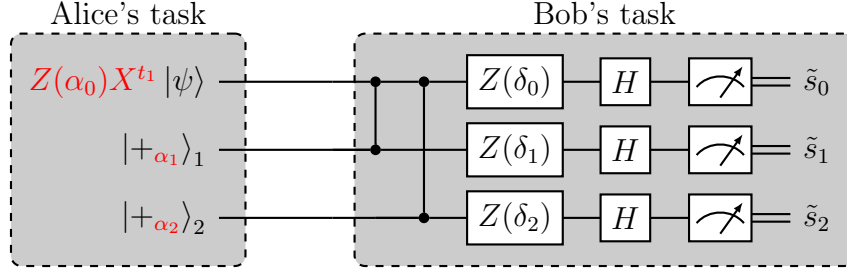


Figure 3.3: Task separation between Alice and Bob, in the sense of quantum operations. Alice’s task is to produce encrypted states, which are transmitted to Bob afterward. Bob’s task is to perform the whole 1WQC protocol on her encrypted qubits.

The task separation shown in Figure 3.3 represents also the distribution of quantum power; it shows quantum operations that Alice and Bob must be able to perform.

The information hiding strategy can be observed from the task separation picture: Bob owns only classical information δ_i and $\tilde{s}_i$, which are uncorrelated to Alice’s computation. It is obvious that δ_i is independent of ϕ_i , and $\tilde{s}_i$ is independent of s_i , therefore, the UBQC is blind. However, such statements are specifically attributed to a statistical distribution; they do not provide a strong security definition. Nevertheless, the UBQC is still secure given a stronger definition: it has been proven that the UBQC is blind within a *composable* security definition [113]. A cryptographic protocol is composable secure if it is also secure as part of a bigger system. Later, in Section 3.2, we discuss a framework that provides a composable security definition so-called *Abstract Cryptography*.

Recall that, a unitary operation can be produced by various forms of graphs. To reduce the information received by Bob (the definition of graph), the UBQC uses a fixed graph state as the universal resource state, the so-called *brickwork state* (see Definition 3.1). Thus, Bob knows only the size of the graph. Also, the UBQC admits a finite set of measurement angles $\{0, \pi/4, 2\pi/4, \dots, 7\pi/4\} := \Omega$, which is sufficient to obtain approximate universality [25].³

Definition 3.1. [88] A brickwork state $\mathcal{G}_{n \times m}$, where $m \equiv 5 \pmod{8}$, is an entangled state of $n \times m$ qubits constructed as follows:

³ The notion of approximate universality comes from the Solovay-Kitaev theorem [114], which states that, if a set of single-qubit quantum gates generates a dense subset of $SU(2)$, then that set is guaranteed to fill $SU(2)$ quickly, thus it can give an efficient approximation for any unitary. Such a set of operations is $\{H, \text{CNOT}, \frac{\pi}{8}\text{-phase}\}$ [114].

1. Prepare all qubits in state $|+\rangle$ and assign to each qubit an index (i, j) , i being a row ($i \in [n]$) and j being a column ($j \in [m]$).
2. For each row, apply the operator $CPHASE$ on qubits (i, j) and $(i, j+1)$ where $1 \leq j \leq m-1$.
3. For each column $j \equiv 3 \pmod{8}$ and each odd row i , apply the operator $CPHASE$ on qubits (i, j) and $(i+1, j)$ and also on qubits $(i, j+2)$ and $(i+1, j+2)$.
4. For each column $j \equiv 7 \pmod{8}$ and each even row i , apply the operator $CPHASE$ on qubits (i, j) and $(i+1, j)$ and also on qubits $(i, j+2)$ and $(i+1, j+2)$.

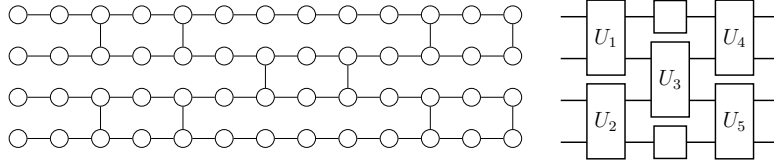


Figure 3.4: The left figure is the resulting brickwork state from Definition 3.1 with $m = 13$ and $n = 8$. The right figure is the circuit representation of the brickwork state on the left, where U_i corresponds to a 2-qubit gate and the empty boxes represent 1-qubit gates. The measurement order is from left to right and from top to bottom.

In a brickwork state, the input nodes sit in the first column, whereas the output nodes sit in the last column. It is obvious that the flow goes from left to right. Consider an example of a brickwork state in Figure 3.4 that is produced from Definition 3.1. Also, brickwork state is universal, *i.e.*, it can produce arbitrary 1-qubit gates and a CNOT gate as shown in Figure 3.5.

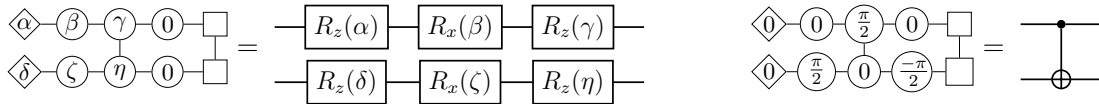


Figure 3.5: Adapted from [25]. Arbitrary 1-qubit rotation and a CNOT gate implemented on a “brick” of the brickwork state, where the measurement angles are specified on the node. The presence of unmeasured nodes — ones without angle — indicate quantum output. The left graph implements 1-qubit arbitrary rotations on two qubits; the right graph implements a CNOT gate.

3.2 ABSTRACT CRYPTOGRAPHY

A cryptographic protocol is said to be secure when it satisfies a particular parameterized equation that captures the desired security features; however, it is tricky to define such equations. In classical cryptography, it sometimes happens that a well-known protocol that was proven to be secure shows an unexpected behavior in a different setting; this happens due to the security definition, where the statement of “secure” itself does not have a standard definition. Since many cryptographic protocols are running on various devices that can be interconnected, to come up with a standard security definition is hard.

The problem of finding a good security definition is also encountered in quantum cryptography. For instance, [115–117] suggest that to encrypt a quantum state⁴ ρ_A with a random key K , where K has distribution $p_K(k)$, the condition called ε -secure can be achieved by finding a randomizing CPTP maps $\{E_k\}_{k \in K}$ such that

$$\left\| \sum_k p_K(k) E_k(\rho_A) - \xi \right\|_1 \leq \varepsilon, \quad (3.4)$$

where $\xi = \frac{1}{d}I$ for dimension d . After a few years, it was proven that such CPTP maps do not hide the information from a player (Eve) that entangles her state to ρ_A [118]. Instead, we need to consider the statement over the purification, namely

$$\left\| \sum_k p_K(k) (E_k \otimes I_E)(\rho_{AE}) - \xi \otimes \text{tr}_A(\rho_{AE}) \right\|_1 \leq \varepsilon. \quad (3.5)$$

Eventually, the problem of finding a standard security definition has been addressed independently by Pfitzmann and Waidner [119] and by Canetti [120]. They proposed a universal *composable* framework within a paradigm referred to as “real-world ideal-world”; this paradigm assesses the security by quantifying how well a real protocol approximates an ideal protocol to perform some cryptographic task. Later, several composable frameworks that also work for quantum systems have been introduced by Ben-Or and Mayers [121], by Unruh [122], and by Maurer and Renner [123].

The frameworks mentioned above are designed to model the security of an arbitrary cryptographic protocol. We can think of these frameworks as a language; a security definition is said to be composable if it is expressed within a composable language. A cryptographic protocol that is secure within a composable security definition is

⁴ For an explicit example, see QOTP in the extra box in [Section 3.1](#).

also (composable) secure when composing a more extensive cryptographic system. Thus, to prove the security of a complex system, one can decompose it into smaller and weaker systems, then confirm the security of those small systems instead. Note that composable security does not necessarily provide a stronger definition; instead, the goal is to make a weak or strong definition becomes apparent.

In our study, we use the *Abstract Cryptography* (AC) framework from Maurer and Renner [123] to model the security of our protocols. In the following, we briefly discuss the AC framework.

The AC has an ideal-world real-world paradigm, where the distance between the two worlds quantifies the security. Essentially, we want to construct an ideal resource, living in the ideal world, from real resources living in the real world via a cryptographic protocol. An ideal resource captures the desired functionalities, and a real resource captures functionalities that are available and implementable in practice. When a real resource is completely indistinguishable from the ideal resource, we say the protocol is unconditionally or perfectly secure. The AC approach is top-down, *i.e.*, starting from the highest possible level of abstraction, then proceeds downwards. It introduces in each level the minimum necessary specializations.

In AC, a *system* is an abstract object with interfaces. There are two kind of systems: *resource* systems and *converter* systems. A *resource* is a system with a set of interfaces that are associated with parties, *e.g.*, a set of interfaces $\mathcal{I} = \{A, B, C\}$, where A, B, C , indicate the interface of Alice, Bob, and Charlie, respectively. An interface associated to a party models how the corresponding party interacts with the resource. A *converter* is a system with two interfaces: inside and outside interfaces. The inside interface is connected to resources, and the outside interface receives inputs and gives outputs. A converter is attached to a resource, and a party can use the resource via her converter.

Given a resource $\mathcal{R}$ with interface set $\mathcal{I}$, a (*cryptographic*) *protocol* is a set of converters of honest parties $\{\pi_i\}$, where $i \in \mathcal{I}$. The inside interface of converter π_i is connected to resource $\mathcal{R}$ via interface i , forming a new resource $\pi_i\mathcal{R}$ with the same set of interfaces $\mathcal{I}$. Note that, the writing order is arbitrary, *i.e.*, $\pi_i\mathcal{R} = \mathcal{R}\pi_i$.

Constructing an ideal resource $\mathcal{S}$ from a real resource $\mathcal{R}$ as good as ε using a *protocol* π , is denoted as

$$\mathcal{R} \xrightarrow{\pi, \varepsilon} \mathcal{S},$$

which is obtained by proving

$$d(\pi\mathcal{R}, \mathcal{S}) \leq \varepsilon, \tag{3.6}$$

where d is a pseudo-metric defined on the resource space. Given resources $\mathcal{R}, \mathcal{S}, \mathcal{T}$ and converter π , pseudo-metric d must have the following properties:⁵

$$\begin{aligned} d(\mathcal{R}, \mathcal{R}) &= 0 & (\text{identity}) \\ d(\mathcal{R}, \mathcal{S}) &= d(\mathcal{S}, \mathcal{R}) & (\text{symmetry}) \\ d(\mathcal{R}, \mathcal{S}) &\leq d(\mathcal{R}, \mathcal{T}) + d(\mathcal{T}, \mathcal{S}) & (\text{triangle inequality}), \end{aligned} \tag{3.7}$$

Additionally, the following non-increasing composition rules must be fulfilled:

$$\begin{aligned} d(\pi\mathcal{R}, \pi\mathcal{S}) &\leq d(\mathcal{R}, \mathcal{S}) & (\text{with converters}) \\ d(\mathcal{R} \parallel \mathcal{T}, \mathcal{S} \parallel \mathcal{T}) &\leq d(\mathcal{R}, \mathcal{S}) & (\text{with resources}), \end{aligned} \tag{3.8}$$

where $\parallel$ indicates concurrency, *e.g.*, $\mathcal{R} \parallel \mathcal{T}$ means $\mathcal{R}$ and $\mathcal{T}$ exists in parallel independently. Thus, when $d(\pi\mathcal{R}, \mathcal{S}) \leq \varepsilon$ is achieved, protocol π is called ε -secure to perform functionalities $\mathcal{S}$, using functionalities in $\mathcal{R}$, with probability of failure ε .

In practice, we use the help of a *distinguisher* to distinguish between two resources. Simply put, two resources $\mathcal{R}$ and $\mathcal{S}$ are indistinguishable if the rest of the world cannot tell whether it is interacting with $\mathcal{R}$ or $\mathcal{S}$; thus, the notion of distinguisher is introduced to capture the rest of the world. Given a black box that contains either resource $\mathcal{R}$ or resource $\mathcal{S}$, a distinguisher can be considered as referee that can freely interact with the black box. This interaction consists of giving inputs, reading outputs, and taking the role of an adversary. Then, it is asked to guess what system the black box is, whether it is $\mathcal{R}$ ($B = 1$) or $\mathcal{S}$ ($B = 0$). Here, the distance metric is called *distinguishing advantage*. Let random variable D indicate the distinguisher's guess; the distinguishing advantage is defined as

$$|\Pr[D = 0 \mid B = 0] - \Pr[D = 0 \mid B = 1]|, \tag{3.9}$$

which is the difference between guessing correctly and erroneously. Since a distinguisher is not particularly specified, one may have a class of distinguishers, *e.g.*, computationally bounded or unbounded distinguishers. The distinguishing advantage of a class of distinguishers is the maximum value of [Equation \(3.9\)](#). For example, if the resources are quantum channels, the computationally unbounded distinguisher class has distinguishing advantage that corresponds to the diamond norm between channels [\[124\]](#).

Up to this point, we have not mentioned what are the kinds of (mathematical) objects that instantiate converters and resources. Any object that satisfies the AC composition rules can represent an AC object; we will touch on these compositions

⁵ It is not a metric because condition $d(\mathcal{R}, \mathcal{S}) = 0 \Rightarrow \mathcal{R} = \mathcal{S} = 0$ does not hold.

shortly. Usually, we choose an object that can describe our system. For an interactive system such BQC, where back and forth communications and qubit transmissions occur, converters and resources can be instantiated with a *quantum comb* [125]. The quantum comb is a generalization of quantum channels, which maps quantum circuits to other quantum circuits, rather than quantum states to quantum states. Thus, if the system is instantiated with a quantum comb, the compositions are defined according to the operations on quantum combs.

A composable security definition allows us to partially reason about the security of a cryptographic protocol within a complex system. The following constructions show the composition rules whose proofs can be found in [123].

Converters can be composed with other converters. Given two converters π_i and κ_i , and two resources $\mathcal{S}$ and $\mathcal{R}$, there are two compositions :

$$\text{serial composition } (\pi\kappa)_i\mathcal{R} \equiv \pi_i(\kappa_i\mathcal{R}) \quad (3.10)$$

$$\text{parallel composition } (\pi \parallel \kappa)_i(\mathcal{R} \parallel \mathcal{S}) \equiv (\pi_i\mathcal{R}) \parallel (\kappa_i\mathcal{S}). \quad (3.11)$$

The composition rules above imply the following resource compositions:

$$\text{serial composition } \text{if } \mathcal{R} \xrightarrow{\pi, \varepsilon} \mathcal{S} \text{ and } \mathcal{S} \xrightarrow{\pi', \varepsilon'} \mathcal{T} \Rightarrow \mathcal{R} \xrightarrow{\pi'\pi, \varepsilon + \varepsilon'} \mathcal{T} \quad (3.12)$$

$$\text{parallel composition } \text{if } \mathcal{R} \xrightarrow{\pi, \varepsilon} \mathcal{S} \text{ and } \mathcal{R}' \xrightarrow{\pi', \varepsilon'} \mathcal{S}' \Rightarrow \mathcal{R} \parallel \mathcal{R}' \xrightarrow{\pi \parallel \pi', \varepsilon + \varepsilon'} \mathcal{R}' \parallel \mathcal{S}'. \quad (3.13)$$

In the serial composition of resources, protocol $\pi'\pi$ constructs resource $\mathcal{T}$ from resource $\mathcal{R}$. In the parallel composition, protocols π and π' are executed concurrently and separately. Consider the extra box “Composable QKD” to illustrate these compositions.



Composable QKD

Here, we are constructing a secure key channel QKD known as BB84 [11] within the AC construction. Our goal is to obtain a long key channel from weaker channels.

The ideal resource is a long key channel $(\mathcal{K} \parallel \mathcal{K} \parallel \mathcal{K})$; we build the long key channel from a (insecure) quantum channel $\mathcal{Q}$ and an authentic channel $\mathcal{A}$ via protocol $\pi_{\mathcal{K}}$, with error $\varepsilon_{\mathcal{K}}$. Pictorially,

$$(\heartsuit) \quad \begin{array}{c} \text{marker} \\ \text{document} \end{array} \xrightarrow{\pi_{\mathcal{K}}, \varepsilon_{\mathcal{K}}} \begin{array}{c} \text{key} \\ \text{key} \\ \text{key} \end{array} \equiv (\mathcal{Q} \parallel \mathcal{A}) \xrightarrow{\pi_{\mathcal{K}}, \varepsilon_{\mathcal{K}}} (\mathcal{K} \parallel \mathcal{K} \parallel \mathcal{K}).$$

To build the authentic channel $\mathcal{A}$, we need an insecure classical channel $\mathcal{C}$ and a short key $\mathcal{K}$, via protocol $\pi_{\mathcal{A}}$. Notice that we are trying to build a resource with a stronger security definition from the weaker ones.

$$(\spadesuit) \quad \begin{array}{c} \text{document} \\ \text{key} \end{array} \xrightarrow{\pi_{\mathcal{A}}, \varepsilon_{\mathcal{A}}} \begin{array}{c} \text{marker} \\ \text{document} \end{array} \equiv (\mathcal{C} \parallel \mathcal{K}) \xrightarrow{\pi_{\mathcal{A}}, \varepsilon_{\mathcal{A}}} (\mathcal{A}).$$

Using the composability, we can obtain a long key using resources $\{\mathcal{Q}, \mathcal{C}, \mathcal{K}\}$; pictorially

$$\begin{aligned} & \begin{array}{c} \text{document} \\ \text{key} \end{array} \xrightarrow{\pi_{\mathcal{A}}, \varepsilon_{\mathcal{A}}} \begin{array}{c} \text{marker} \\ \text{document} \end{array} \xrightarrow{\pi_{\mathcal{K}}, \varepsilon_{\mathcal{K}}} \begin{array}{c} \text{key} \\ \text{key} \\ \text{key} \end{array} \\ &= (\mathcal{Q} \parallel \mathcal{C} \parallel \mathcal{K}) \xrightarrow{id \parallel \pi_{\mathcal{A}}, \varepsilon_{\mathcal{A}}} (\mathcal{Q} \parallel \mathcal{A}) \xrightarrow{\pi_{\mathcal{K}}, \varepsilon_{\mathcal{K}}} (\mathcal{K} \parallel \mathcal{K} \parallel \mathcal{K}) \\ &= (\mathcal{Q} \parallel \mathcal{C} \parallel \mathcal{K}) \xrightarrow{(\pi_{\mathcal{K}})(id \parallel \pi_{\mathcal{A}}), \varepsilon_{\mathcal{A}} + \varepsilon_{\mathcal{K}}} (\mathcal{K} \parallel \mathcal{K} \parallel \mathcal{K}), \end{aligned}$$

where id indicates an identity protocol that does nothing. We denote the long key as three resources $\mathcal{K}$, to indicate that a small part of the long key, *e.g.*, one $\mathcal{K}$, can be used for a future QKD protocol.

Thus, the BB84 protocol $\pi_{\mathcal{B}} := (\pi_{\mathcal{K}})(id \parallel \pi_{\mathcal{A}})$ obtains a long key $(\mathcal{K} \parallel \mathcal{K} \parallel \mathcal{K})$ from resources $\{\mathcal{Q}, \mathcal{C}, \mathcal{K}\}$ with $(\varepsilon_{\mathcal{A}} + \varepsilon_{\mathcal{K}})$ -secure; this security statement can be concluded by proving its composites, *i.e.*, proving that $(\heartsuit)$ and $(\spadesuit)$ are true.

In a cryptographic context, we consider “honest” and “dishonest” parties. Honest parties follow their protocols, while dishonest parties behave arbitrarily; such unspecified behavior is often called malicious activities. Malicious activities are often modeled as a single party that is called an adversary. In the following,

we define the notion of security that is adapted to our three-party protocol, the so-called Blind Oracular Quantum Computation (BOQC); the BOQC protocol is extensively discussed in [Chapter 5](#).

Given an ideal resource $\mathcal{S}$ and a real resource $\mathcal{R}$ with a set of interfaces $\mathcal{I} = \{\mathcal{A}, \mathcal{B}, \mathcal{O}\}$, where each interface is associated with a party: Alice ($\mathcal{A}$), Bob ($\mathcal{B}$), and Oscar ($\mathcal{O}$). We are given a protocol π (for example, the BOQC protocol in [Protocol 5.1](#)), where Bob can be malicious. Thus, there are two security statements of protocol π that we want to fulfill to achieve composable security: *correctness* (or *completeness*) and *security* (or *soundness*). For our setting (for the BOQC), correctness is defined in [Definition 3.2](#) and security is defined in [Definition 3.3](#) below.

Definition 3.2. [\[123\]](#) Let $\pi = \{\pi_{\mathcal{A}}, \pi_{\mathcal{O}}, \pi_{\mathcal{B}}\}$ be a protocol without adversary and $\mathcal{R}, \mathcal{S}$ be resources. Protocol π is ε -correct if it constructs an ideal resource within ε , viz., $\mathcal{R} \xrightarrow{\pi, \varepsilon} \mathcal{S}$, such that

$$d(\pi_{\mathcal{A}}\pi_{\mathcal{O}}\mathcal{R}\pi_{\mathcal{B}}, \mathcal{S}) \leq \varepsilon.$$

Definition 3.3. [\[123\]](#) Let $\pi = \{\pi_{\mathcal{A}}, \pi_{\mathcal{O}}\}$ be a protocol with an adversary $\mathcal{B}$ and $\mathcal{R}, \mathcal{S}'$ be resources. Protocol π is ε -secure if it constructs an ideal resource within ε , viz., $\mathcal{R} \xrightarrow{\pi, \varepsilon} \mathcal{S}'$, and there exists a converter $\sigma_{\mathcal{B}}$ (called simulator), such that

$$d(\pi_{\mathcal{A}}\pi_{\mathcal{O}}\mathcal{R}, \mathcal{S}'\sigma_{\mathcal{B}}) \leq \varepsilon.$$

The correctness statement applies when all parties are honest, whereas the security statement applies in the presence of an adversary, namely when Bob is cheating — which is the case considered here. Simply put, protocol π supposedly realizes resource $\mathcal{S}$, but it realizes another resource $\mathcal{S}'$ if Bob is dishonest. Note that, in general, the real resource $\mathcal{R}$ and error ε for both definitions are allowed to be different. The notion “secure” is defined accordingly, depending on the desired security flavor, such as blindness, secrecy, authenticity, and non-malleability. In addition, if $\varepsilon = 0$ in both definitions, we say that protocol π is unconditionally or perfectly secure. In this thesis, we aim to achieve perfect blindness as the security of our protocols.

Notice that, the notion simulator $\sigma_{\mathcal{B}}$ appears in the security statement, [Definition 3.3](#). A simulator is an arbitrary converter appended to resource $\mathcal{S}'$ on its $\mathcal{B}$ -interface to make $\mathcal{S}'$ and $\pi_{\mathcal{A}}\pi_{\mathcal{O}}\mathcal{R}$ comparable. Resource $\mathcal{S}'\sigma_{\mathcal{B}}$ is called a *relaxation* of $\mathcal{S}'$ [\[126\]](#). The main idea of a simulator is to assure that none of these relaxations

can be more useful to Bob than the ideal resource. All relaxations are accepted in the ideal world; thus, a real-world system is secure if it is indistinguishable from at least one relaxation of the ideal system.⁶ The definition of simulator σ_B is independent of Bob’s cheating strategies; however, it is common to assume all converters to be computationally efficient.⁷ In this thesis, we assume every converter is computationally efficient, *i.e.*, it runs in a polynomial-time, uses a polynomial-size resource, and does not have an infinite sequence. This assumption implicitly limits Bob’s cheating strategy, *e.g.*, he cannot ask his client to infinitely resend him the lost qubits or using an exponential-size resource to process information from his clients.

It is important to remark that this framework does not capture Bob’s cheating strategy. Also, the notion of error ε does not capture the details on what kind of failure or how severe it is. As a language, the AC aims to make a security definition more apparent (weak or strong). Instead, we should ask whether the defined ideal resource captures all the security features that we need. If it does not, we need to define it differently.

We will see an explicit use of AC in [Chapter 5](#). We use AC to prove the security of our protocols, namely the BOQC and the BOQCo (BOQC-optimized) protocols.

⁶ We can think of a simulator as a device attached to the ideal resource that keeps sending information to Bob. An ideal resource accepts a set of “extensions,” namely simulators attached to the ideal resource, which lives in the ideal world. Thus, if a real resource with the presence of an adversary is indistinguishable to an element in the set, then the protocol is secure. We refer to [126] for further details on relaxation.

⁷ In a practical sense, a cryptographic protocol involve only efficient computations. When a protocol needs *e.g.*, an exponential time or exponential space, it indicates a bad protocol. If we consider asymptotic notion in a protocol, such as sending qubit infinitely, involving an exponential-time computation, or using an exponential-space, these exponential notions must be considered in σ_B .

ORACULAR QUANTUM ALGORITHMS

In this chapter, we briefly discuss several oracular quantum algorithms that are related to demonstrated algorithms in [Section 7.2](#), which are executed on an NV-center platform, within the BOQC scheme that is introduced in [Chapter 5](#). We start with a brief discussion on complexity classes and a description of the oracle model. It is important to point out the role of quantum oracles, which underlies our motivation to come up with the BOQC scheme — a delegated oracular quantum computation with separated oracles. Afterward, we review algorithms related to [Chapter 6](#): Deutsch’s algorithm [\[8\]](#), Deutsch and Jozsa’s algorithm [\[9\]](#), Grover’s algorithm [\[19\]](#), Høyer’s algorithm [\[127\]](#), and Simon’s algorithm [\[41\]](#).

We note that throughout this chapter we use $\Sigma := \{0, 1\}$.

4.1 COMPLEXITY CLASSES AND ORACLE MODELS

There are a vast number of algorithms dedicated to solving problems with various difficulties; however, some algorithms require more resources than others to solve a given computational problem. It is the task of complexity theorists to classify the problems based on their inherent difficulty (hardness) provided the most efficient solver is known. A complexity class is a set of problems solvable on a specified computational model that obey certain constraints. Those constraints may comprise one or many of: time and space constraints, determinism, probabilistic computing machine, quantum computing machine, and particular interaction with different computing machines. An extensive catalog of complexity classes can be found in [\[128\]](#). Here, we briefly discuss several complexity classes that are mentioned in this thesis. One can find different conventions on notations or definitions with other references; here we use some combinations of Arora and Barak’s book [\[129\]](#) and Watrous’s notes [\[130\]](#).

In modeling a computation, one can think of an algorithm (also referred to as a computation, a function, a functionality, or a process) as a machine that receives an input and gives the output as pictured in Figure 4.1, where $\mathfrak{M}$ receives a bit string input x and outputs another bit string $\mathfrak{M}(x)$. The input and

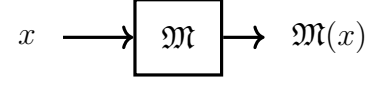


Figure 4.1: A computing machine $\mathfrak{M}$.

output encode an object that we wish to represent, such as number, graph, vector, quantum state, and random sample. The machine $\mathfrak{M}$ refers to a *Turing Machine* (TM) that is described in Figure 4.2, where a formal definition can be found in [129]. Depending on the underlying problem, the machine can be a variation of a TM such as deterministic TM, probabilistic TM, non-deterministic TM,¹ oracle TM, quantum TM, or quantum oracle TM. The definition of TMs can be found in [129]; a reformulated definition asserting physical limitation is written by Deutsch in [8]. The probabilistic TM and quantum TM definitions can be found in [131], and the definition of quantum oracle TM is provided in [132].

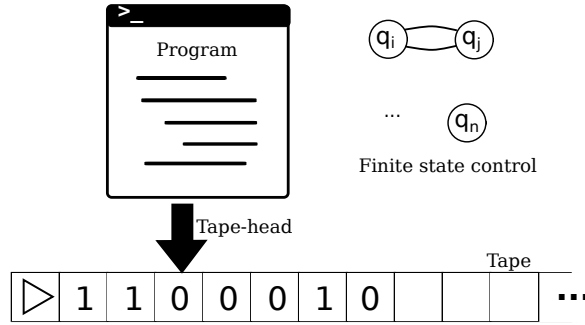


Figure 4.2: A Turing Machine (TM) [129, 133]. TM comprises a program, finite inner states, an infinite tape, and a tape-head that can move one step (to left or right) and stay. A TM is described by a tuple (Γ, Q, δ) , where $\Gamma = \{\triangleright, 0, 1, b\}$ is an alphabet (symbols that the tape can contain, where b indicates “blank” and $\triangleright$ marks the tape’s left edge), $Q = \{q_{start}, q_{halt}, q_i, \dots, q_n\}$ is a finite set of internal states stored in a “register”, and δ is a map (program) that controls the state on each step (head-tape’s move, cell’s value, and when to halt). The machine starts with state q_{start} , the computation proceeds according to δ , and the machine halts — if it ever halts — when it reaches q_{halt} state.

A *decision problem* is a type of problem that is often studied in complexity theory. Referring to Figure 4.1, solving a decision problem means $\mathfrak{M}(x)$ gives a “yes” or “no” answer, *i.e.*, one bit of output. In most cases, observing one bit of output is

¹ Note that non-deterministic TM is different from probabilistic TM. The behavior of a probabilistic TM is depending on the random number generator; thus, it is deterministic with respect to a probability distribution. A non-deterministic TM can behave unexpectedly for the same input in different runs, which happens for different reasons such as the concurrency setting, decisions made during the run, and unknown probability distribution.

enough to learn the behavior of the problem. Given Σ^* signifies an infinite set that contains all possible bit strings, a decision problem can be formalized in the form of a *promise problem* [126, 130, 134]² as defined in Definition 4.1.

Definition 4.1. [126] A *promise problem* Π is a pair of non-intersecting sets, denoted (Π_{yes}, Π_{no}) ; that is, $\Pi_{yes}, \Pi_{no} \subseteq \Sigma^*$ and $\Pi_{yes} \cap \Pi_{no} = \emptyset$. The set $\Pi_{yes} \cup \Pi_{no}$ is called the *promise*.

Let us illustrate Definition 4.1 for our machine $\mathfrak{M}$. Consider the set of allowed strings (a promise) X , thus $X = X_{yes} \cup X_{no}$ and $X_{yes} \cap X_{no} = \emptyset$ where X_{yes} comprises the inputs with “yes” answer (yes-instance) and X_{no} comprises the inputs with “no” answers (no-instance); anything outside the promise gives neither yes-instance nor no-instance. A computation model now can be pictured as Figure 4.3.

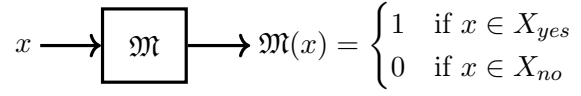


Figure 4.3: A machine $\mathfrak{M}$ that solves a promise problem.

Thus, the main task of a promise problem is distinguishing yes-instances from no-instances. Consider Example 4.1 in which we phrase the graph isomorphism³ problem as a promise problem.

Example 4.1. *Graph isomorphism problem as a promise problem.*

input	$x \in \Sigma^n$ that encodes two simple graphs $\mathcal{G}_1$ and $\mathcal{G}_2$
yes	$\mathcal{G}_1 \cong \mathcal{G}_2$ (x is isomorphic)
no	$\mathcal{G}_1 \not\cong \mathcal{G}_2$ (x is non-isomorphic).

A complexity class is a collection of promise problems obeying certain constraints. A set of problems taking a similar range of time and space to solve by a particular TM are grouped into the same complexity class. The time (the number of steps of a TM) and the space complexity are expressed with the big-O notation ($\mathcal{O}$).⁴ In the following, we provide definition of several well-known classes, which are adapted from [130].

-
- 2 The promise problem is a generalization of decision problems; it provides a different presentation of languages — a more commonly used formulation. One may think of languages as promise problems with an additional constraint, $\Pi_{yes} \cup \Pi_{no} = \Sigma^*$.
- 3 The graph isomorphism is defined in Definition 2.6
- 4 The big-O notation describes the upper bound of an algorithm asymptotically, i.e., $f(n) \in \mathcal{O}(g(n))$ if there exists $c > 0$ and an integer n_0 such that $f(n) \leq cg(n)$, $\forall n \geq n_0$. Here, f and g are integer functions since we are dealing with the time and space quantities.

- P** *Polynomial-time.* A promise problem Π is in **P** if and only if there exists a polynomial-time⁵ deterministic TM that accepts every $x \in \Pi_{yes}$ and rejects every $x \in \Pi_{no}$.
- NP** *Non-deterministic Polynomial-time.* A promise problem Π is in **NP** if and only if there exists a polynomial-bounded function p and a polynomial-time deterministic TM $\mathfrak{M}$, such that $\forall x \in \Pi_{yes}$, $\mathfrak{M}$ accepts (x, y) for some certificate $y \in \Sigma^{p(|x|)}$, and $\forall x \in \Pi_{no}$, $\mathfrak{M}$ rejects $(x, y) \forall y \in \Sigma^{p(|x|)}$.⁶
- PP** *Probabilistic Polynomial-time.* A promise problem Π is in **PP** if and only if there exists a polynomial-time probabilistic TM $\mathfrak{M}$, such that $\mathfrak{M}$ accepts every $x \in \Pi_{yes}$ with probability $> \frac{1}{2}$, and accepts every $x \in \Pi_{no}$ with probability $< \frac{1}{2}$.
- BPP** *Bounded-error Probabilistic Polynomial-time.* A promise problem Π is in **BPP** if and only if there exists a polynomial-time probabilistic TM that accepts every $x \in \Pi_{yes}$ with probability $\geq \frac{2}{3}$ and accepts $x \in \Pi_{no}$ with probability $< \frac{1}{3}$.⁷
- MA** *Merlin-Arthur.* A promise problem Π is in **MA** if and only if there exists a polynomial-bounded function p and a probabilistic polynomial-time TM $\mathfrak{M}$, such that $\forall x \in \Pi_{yes}$, $\Pr[\mathfrak{M} \text{ accepts } (x, y)] \geq \frac{2}{3}$ for some $y \in \Sigma^{p(|x|)}$, and $\forall x \in \Pi_{no}$, $\Pr[\mathfrak{M} \text{ accepts } (x, y)] \leq \frac{1}{3}$, $\forall y \in \Sigma^{p(|x|)}$.
- PSPACE** *Polynomial Space.* A promise problem Π is in **PSPACE** if and only if there exists a polynomial-space deterministic TM that accepts $\forall x \in \Pi_{yes}$ and rejects $\forall x \in \Pi_{no}$.

For some classes, it can be intuitively proven to be larger than another. For example, $\mathbf{BPP} \subseteq \mathbf{PP}$, because having a **PP** problem with significantly smaller errors produces **BPP** class; note that we are using $\subseteq$ here, since it is hard to provide a strict proof. Using some conjectures and proofs, to date, we know that $\mathbf{PSPACE} \supseteq \mathbf{PP} \supseteq \mathbf{BPP} \supseteq \mathbf{P}$ and $\mathbf{MA} \supseteq \mathbf{NP} \supseteq \mathbf{P}$ [130].

Example 4.2. *In the following, we show that the graph isomorphism problem in Example 4.1 is in **NP**.*

-
- ⁵ Polynomial-time means the machine halts after $P(n)$ steps, where P is a polynomial and n some input parameter.
- ⁶ Note that the notion of non-determinism in **NP** problem stems from finding the certificate, which is done on a non-deterministic TM. Simply put, **NP** problems efficiently verify the answer but inefficient in finding the answer.
- ⁷ The number $\frac{2}{3}$ is an arbitrary number that is larger than $\frac{1}{2}$, so that high statistical confidence can build up quickly.

Given that the certificate is an isomorphism, i.e., in the form of permutation of nodes $p : n_i \mapsto m_k$, for all $n_i \in V(\mathcal{G}_1)$ and $m_k \in V(\mathcal{G}_2)$; the verifier function is performing p and checking the equality between $\mathcal{G}_2$ and $\mathcal{G}_p$ by comparing their adjacency matrices, where $\mathcal{G}_p$ is the graph whose nodes are permuted according to p , i.e., $\mathcal{G}_p$ has nodes $\{p(i) \mid i \in V(\mathcal{G}_1)\}$.

In quantum computations, **BQP** (Bounded-error Quantum Polynomial-time) is the class that contains promise problems that are efficiently solvable using a quantum computer — Quantum TM (QTM). **BQP** has been the class of greatest interest in quantum computing, as it is natural to ask what kind of problems that quantum computers can efficiently solve.

Before defining **BQP**, let us define a QTM. Here, we use Watrous's definition in [Definition 4.2](#).⁸ Notice that [Definition 4.2](#) considers a class of quantum circuits that can be generated efficiently by a classical TM. Recall that we use $\Sigma := \{0, 1\}$.

Definition 4.2. [[130](#)] Given $S \subseteq \Sigma^*$, a collection of quantum circuits $\{Q_x : x \in S\}$ is said to be polynomial-time generated if there exists a polynomial-time deterministic TM that, for every input $x \in S$ outputs an encoding⁹ of Q_x .

In this definition, a bit string x hard-codes a collection of quantum circuits Q_x . A polynomial-time generated circuit implies that it takes polynomial-time for a classical TM to decide which circuits to operate. Additionally, [Definition 4.2](#) is motivated from the Solovay-Kitaev theorem [[114](#)]: given a universal set of quantum gates, e.g., $\{\text{Toffoli}, H\}$ [[135](#), [136](#)], $\text{poly}(\log(1/\varepsilon))$ number of gates is enough to approximate a unitary with an arbitrary precision ε , given a fixed number of qubits. Now, we are ready to define the **BQP** class.

BQP Let Π be a promise problem and $a, b : \mathbb{N} \rightarrow [0, 1]$ be polynomials. Then, $\Pi \in \mathbf{BQP}(a, b)$ if and only if there exist polynomial-time generated quantum circuits $Q = \{Q_n : n \in \mathbb{N}\}$, where Q_n is an operator with n -input qubits and one output qubit, such that $\forall x \in \Pi_{yes}$, $\Pr[Q \text{ accepts } x] \geq a(|x|)$, and $\forall x \in \Pi_{no}$, $\Pr[Q \text{ accepts } x] \leq b(|x|)$.

Following the same reasoning as for the **BPP** class, one usually defines $\mathbf{BQP} \equiv \mathbf{BQP}(\frac{2}{3}, \frac{1}{3})$; therefore, **BQP** is often referred to as the quantum analog of **BPP**.

⁸ Watrous's definition of QTM admits Yao's definition in [[14](#)]. However, the first requires polynomial-time for both classical and quantum steps, whereas the latter considers only polynomial-time (or size) of quantum circuit.

⁹ For every quantum circuit Q_x , there exists a unique bit string that encodes Q_x ; one may think of this encoding as a drawing of the circuit.

In the landscape of complexity theory, it is commonly believed that $\mathbf{PP} \supseteq \mathbf{BQP} \supseteq \mathbf{BPP}$. Given that quantum computers are universal and are hard to be simulated by classical TMs, while a QTM can efficiently simulate classical logic gates, it follows that $\mathbf{BPP} \subseteq \mathbf{BQP}$. The same reasoning implies $\mathbf{BPP} \subseteq \mathbf{PP}$, and we say that $\mathbf{PP} \supseteq \mathbf{BQP}$ because the unbounded error in $\mathbf{PP}$ contains the bounded error in $\mathbf{BQP}$. However, it remains unsolved where $\mathbf{BQP}$ fits exactly in the complexity landscape.

In some cases, QTM can solve harder problems efficiently, *e.g.*, integer factorization¹⁰ — a well-known $\mathbf{NP}$ problem — can be efficiently solved by Shor’s algorithm [15]. Moreover, given access to an *oracle*, QTM can efficiently solve a broader class of problems compared to classical TMs; for instance, a recent result from Raz and Tal in [42] shows that there exists an oracle such that $\mathbf{BQP}$ is separated from the polynomial-hierarchy ($\mathbf{PH}$): $\exists$ oracle $\mathbf{A}$ such that $\mathbf{BQP}^{\mathbf{A}} \not\subseteq \mathbf{PH}$.¹¹ Separation of $\mathbf{BQP}$ to $\mathbf{PH}$ has the following implications. Even when the “one million dollar question” turns out $\mathbf{P}=\mathbf{NP}$, which results in the collapse of $\mathbf{PH}$, quantum computers still have an advantage.

An (standard) oracle Turing Machine (or oracle) is a query type of TM that provides a function evaluation for another TM. It has the form of a subroutine that implements an unknown function

$$f : \Sigma^* \rightarrow \Sigma, \quad (4.1)$$

where $\Sigma := \{0, 1\}$ and Σ^* signifies all possible bit strings. To understand the behavior of f , we have to make queries, tabulate inputs and outputs, and then process the inputs-outputs table. An oracle evaluates every query at unit cost; thus, oracle complexity is characterized by the number of queries to the oracle.

A quantum oracle TM is the quantum analog of the oracle TM, which is implemented as a unitary map. Let $|x\rangle$ be a quantum state in a Hilbert space with size 2^n . A quantum oracle $\mathcal{O}_f$ that implements $f : \Sigma^n \rightarrow \Sigma$ is defined as a unitary that acts on $n + 1$ qubits, *i.e.*,

$$\mathcal{O}_f |x, a\rangle = |x, a \oplus f(x)\rangle \quad (4.2)$$

¹⁰ Integer factorization is at the core of RSA scheme [137], *i.e.*, the commonly implemented cryptographic scheme in secure devices today.

¹¹ $\mathbf{PH}$ generalizes $\mathbf{P}$, $\mathbf{NP}$, and $\mathbf{co-NP}$, where $\mathbf{co-NP}$ is analog to $\mathbf{NP}$ but with the certificate that can efficiently verify no-instances. The notation $\mathbf{BQP}^{\mathbf{A}}$ means a $\mathbf{BQP}$ machine with access to oracle $\mathbf{A}$. For example, let $\mathbf{NP}$ and $\mathbf{BQP}$ be provided with access to a machine $\mathbf{A}$; if it turns out that in this scenario $\mathbf{NP}$ performs slower than $\mathbf{BQP}$, we say $\mathbf{NP}^{\mathbf{A}} \subseteq \mathbf{BQP}^{\mathbf{A}}$ or $\mathbf{NP} \subseteq \mathbf{BQP}$ relative to an oracle $\mathbf{A}$.

where $\oplus$ denotes modulo 2 summation, $a \in \Sigma$ is an ancilla qubit, and f is the predicate (or a promise).¹²

The quantum oracle in Equation (4.2) is known as the *standard oracle model*. Another form of oracle is known as the *phase oracle* which acts as

$$\mathcal{O}_f |x, a\rangle = (-1)^{f(x) \cdot a} |x, a\rangle. \quad (4.3)$$

However, it has been proven that, the phase oracle model can be reduced to the standard oracle model [133, 138].

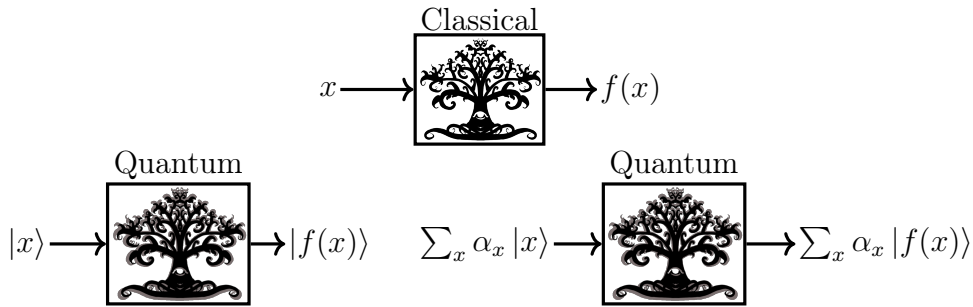


Figure 4.4: Oracles that implement $f : \Sigma^n \rightarrow \Sigma$ in the classical setting (top) and in the quantum setting (below). The superposition of quantum states allows a massive evaluation at once: this is known as *quantum parallelism*, where $\alpha_x \in \mathbb{C}$ and $\sum_x |\alpha_x|^2 = 1$.

To put into words, an oracle works as a “black box”¹³ that computes the answer in unit of time. Consider Figure 4.4 that illustrates classical and quantum oracles. The figure shows that the quantum oracle allows an extensive evaluation at once, albeit the observation is limited to one qubit, *i.e.*, the ancilla qubit. However, some algorithms, such as the Deutsch-Jozsa algorithm, directly take advantage of such parallelism. In the next section, we discuss how the quantum parallelism conditionally provides an exponential speed up to the Deutsch-Jozsa algorithm.

Since some (quantum) oracles do not admit the standard model,¹⁴ Meyer and Pommersheim [139] generalize the standard quantum oracle model as the following. Given that a quantum oracle $\mathcal{O}_f$ acts on $\mathcal{H}_X \otimes \mathcal{H}_R$ evaluating a (classical) function $f : X \rightarrow Y$, where Y contains all possible responses unrestricted to one bit — unlike the standard model, $\mathcal{H}_X$ is the Hilbert space of the input register, and $\mathcal{H}_R$ is the Hilbert space of the response register (ancilla register). Given that R is a fixed finite set, we assign to each $y \in Y$ a permutation $\sigma_y \in S_R$, where S_R signifies a

¹² Note that the promise here has a different notion with the promise problem in Definition 4.1.

¹³ There are also the notions of white and grey boxes. Here, we only consider black boxes, which simply means we do not have any knowledge about function f .

¹⁴ For example, Simon’s oracles that are discussed in Section 4.5 and Section 6.5

symmetric group over set R , and $\sigma : Y \rightarrow S_R$. The quantum oracle acts as follows

$$\mathcal{O}_f |x, a\rangle = |x, \sigma_{f(x)}(a)\rangle, \quad (4.4)$$

where σ is arbitrarily chosen. In this generalized oracle model, the evaluation $f(x)$ is randomized according to permutation σ , while f is a deterministic classical function.

Another oracle model generalization — equivalent to Meyer and Pommersheim's oracle model — so-called *oracle with internal randomness* is introduced by Harrow and Rosenbaum [43], and is extended from Meyer and Pommersheim's oracle model where f is allowed to be a classical probabilistic algorithm. An oracle with internal randomness (inaccessible dice) π is a set of maps $\{\pi_{x,r} \in S_M\}$, where $x \in X$ and r is a random variable distributed according to R . The oracle acts as

$$\mathcal{O}_\pi |x, y\rangle = |x, \pi_{x,r}(y)\rangle. \quad (4.5)$$

We refer to [43] for a more comprehensive definition. Harrow and Rosenbaum introduce some oracle constructions within their model that provide strong separations; for example, *infinity-to-one* means that classical TMs cannot solve the problem even with infinite queries, while a quantum TM is able to solve it with one query. We will mention several constructions of these oracles in the next sections.

4.2 DEUTSCH AND JOZSA'S ALGORITHM

The Deutsch-Jozsa algorithm [9] is the first oracular algorithm suggested to have a substantial advantage over classical computers. The algorithm was discovered in 1992, a few years after the first quantum algorithm is formalized in 1985.¹⁵ The Deutsch-Jozsa problem is formulated in [Problem 4.1](#).

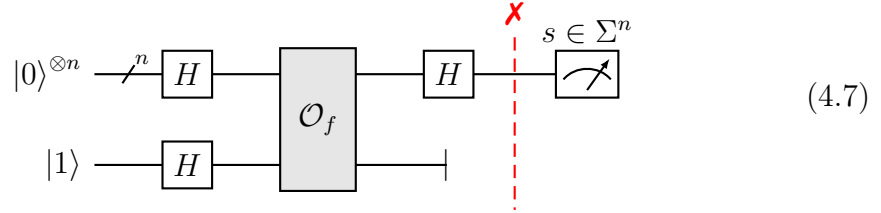
Problem 4.1. *Deutsch and Jozsa's problem [9] using Cleve et al.'s formulation [140]. Given access to an oracle encoding a boolean function $f : \Sigma^n \rightarrow \Sigma$ that is guaranteed to be either constant ($f(x) = 0$ or $f(x) = 1$ for all x) or balanced (2^{n-1} outputs are 1). The problem is determining whether f is constant or balanced.*

¹⁵ The first (oracular) quantum algorithm was introduced by Deutsch in [8], together with a model of quantum Turing Machine. He illustrated a situation that is impossible to solve by classical computers.

Problem 4.1 can trivially be formulated as a promise problem from Definition 4.1 as shown in Equation (4.6). By the promise of f , the no-instance (not constant) in the equation straightforwardly means f is balanced.

$$\begin{array}{ll}
 \textbf{input} & \{x \in \Sigma^n\} \text{ and } f : \Sigma^* \rightarrow \Sigma \\
 \textbf{yes} & f(x) = 1 \text{ or } f(x) = 0 \text{ for all } x \in \Sigma^n \text{ (} f \text{ is constant)} \\
 \textbf{no} & f \text{ is not constant.}
 \end{array} \tag{4.6}$$

The Deutsch-Jozsa algorithm [9] is given in Equation (4.7) below, where only one query to the oracle is needed for an arbitrary n to solve Problem 4.1. The oracle $\mathcal{O}_f$ has a form of the standard model.



Using the fact that

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (-1)^{x \cdot i} |i\rangle, \text{ where } x \in \Sigma^n, \tag{4.8}$$

we can straightforwardly obtain the state at point $\times$ as

$$\frac{1}{2^n} \sum_{j=0}^{2^n-1} \sum_{x=0}^{2^n-1} (-1)^{f(x)+x \cdot j} |j\rangle. \tag{4.9}$$

Thus, for a constant f , it results in the exact measurement outcome $s = 0$. For a balanced f , it results in $s \neq 0$.

Since Equation (4.7) calls the oracle once for all n , the number of queries remain constant with the growing input size; thus the complexity is constant time $\mathcal{O}(1)$. Therefore, Deutsch-Jozsa problem belongs to complexity class **EQP** (exact quantum polynomial-time).¹⁶ On the other hand, $2^{n-1} + 1$ queries are needed to solve Problem 4.1 on classical deterministic TMs. Thus, for deterministic TMs, the quantum oracle provides an exponential speedup. Hence, Deutsch-Jozsa's problem separates **P** and **EQP**.

¹⁶ EQP is the quantum analog of P.

However, if we are not restricted to deterministic TMs, a classical probabilistic TM solves the Deutsch-Jozsa problem in Equation (4.6) efficiently (in polynomial-time), hence this problem belongs to **BPP**. Since Deutsch-Jozsa algorithm solves in polynomial-time, thus, given a quantum oracle, the problem belongs to **BQP**. As we can think of a probabilistic TM as a TM with dice, it is straightforward to see that a few dice throws will reveal the correct answer with high probability; in particular, $\mathcal{O}(\log(1/\varepsilon))$ queries for an error within ε . Therefore, in the computational complexity point of view, there is no advantage in using a quantum computer, as it does not separate **BPP** and **BQP**.

Given that the oracle admits a non-standard oracle model, *e.g.*, the oracle with internal randomness [43], Deutsch-Jozsa (also Deutsch) algorithm can provide a strong separation between classical and quantum queries. For instance, given that the oracle randomly evaluates f constant or f balanced on each query, infinite classical queries are *useless* (no information is gained), whereas one quantum query is not useless, *viz.*, it solves the problem exactly.

4.3 GROVER SEARCH ALGORITHMS

The *unordered search* problem is particularly exciting as it covers a wide range of problems, such as combinatoric search, optimization, and cryptographic problems. For instance, in cryptography, a key search problem can be mapped into an unordered search problem, *e.g.*, by trying all possible keys. This problem is known to be **NP**-complete [141, 142].¹⁷ Simply put, **NP**-complete is a class of problems, for which if one them is efficiently solvable, then all problems in **NP** are efficiently solvable. Thus, knowing the best algorithm to solve an unstructured search problem is crucial.

Grover's algorithm [19] solves an unordered search problem defined in Problem 4.2 with a quadratic speedup — the best-possible speedup to solve this problem [132].

Problem 4.2. *Grover's search [19]. Given access to an oracle that encodes $f : \Sigma^n \rightarrow \Sigma$, does there exist $x \in \Sigma^n$ such that $f(x) = 1$? If such x exists, give x .*

¹⁷ The **NP**-complete problems are harder than all **NP** problems and at least as hard as some **NP**-hard problems. To prove that a problem $\mathfrak{P}$ is **NP**-complete, one needs to reduce all **NP** problems into $\mathfrak{P}$ within polynomial-time. Since all **NP** problems are reducible to an **NP**-complete problem, we can simply reduce an **NP**-complete problem into our problem $\mathfrak{P}$. From the Cook-Levin theorem, we certainly know that the **SAT** problem is in **NP**-complete. It is a common practice to prove **NP**-completeness by reducing **SAT** problem into the corresponding problem.

In the form of a promise, [Problem 4.2](#) can be formulated as

$$\begin{array}{ll}
 \textbf{input} & \text{oracle } f : \Sigma^n \rightarrow \Sigma \\
 \textbf{yes} & \exists x \in \Sigma^n \text{ such that } f(x) = 1 \\
 \textbf{no} & \text{for all } x \in \Sigma^n, f(x) = 0.
 \end{array} \tag{4.10}$$

In the following, we give some intuition why Grover’s problem is in **NP**-complete. First, by referring to [Equation \(4.10\)](#), the problem provides a certificate, namely the oracle f , which efficiently verifies the answer in unit time. Secondly, it is also widely believed that **NP**-complete problems are exponential [\[141\]](#) — on classical TMs. Grover’s problem requires exponential time on deterministic TMs and probabilistic TMs for the following reasons. To obtain no-instance deterministically, one must query 2^n times if unlucky — the worst case; thus, the complexity is $\mathcal{O}(2^n)$. The same exponential complexity is obtained for classical probabilistic algorithms. Given the error probability ε parameterizes a classical probabilistic algorithm, the best strategy is by choosing M out of 2^n inputs at random. Thus, the probability of successfully having a yes-instance x within the choices must be $1 - M/2^n \leq \varepsilon$, which means $M \geq (1 - \varepsilon)2^n$, and so the complexity is $\mathcal{O}(2^n)$ for a probabilistic classical computer. On the other hand, a QTM offers speedup $\mathcal{O}(\sqrt{2^n})$, which is quadratically faster than any classical TMs.

In the following, we will discuss some quantum algorithms that solve [Problem 4.2](#). These algorithms are Grover’s algorithm [\[19\]](#), Høyer’s algorithm [\[127\]](#), and an alternative to Høyer’s strategy to achieve exact results, which was shown in [\[143\]](#).

4.3.1 Grover’s algorithm

Grover’s algorithm [\[19\]](#) is the fastest algorithm that solves an unstructured search problem ([Problem 4.2](#)) providing a quadratic speedup, *i.e.*, it needs only $\mathcal{O}(\sqrt{2^n})$ oracle queries for n qubits database — we will clarify this by the end of this subsection. It is an optimal quantum algorithm [\[144\]](#), which means any algorithm with access the same oracle need to query at least as many as Grover algorithm. This optimality shows the limitation of quantum computers towards solving such problems, which suggests that **NP** is not contained in **BQP**.¹⁸

Although Grover’s algorithm turns out to be useful for many applications, the algorithm was initially motivated by searching for items in a database. Here, we

¹⁸ Note that Grover’s algorithm is not in **BQP**, since the oracle queries are still exponential, albeit quadratically faster than any classical algorithm. Thus, the algorithm is in **NP** ^{$\mathcal{O}$} with some oracle $\mathcal{O}$.

consider Grover's initial motivation, *i.e.*, data searching. For an illustration, given a catalog (database) that lists animals together with their characteristics. We want to find if there exists an animal with a set of characteristics x , where x encodes the combinations of criteria that we seek, for example

$$x := \{\text{living underwater, cell-regenerating capability, biologically immortal}\}.$$

Every set of criteria is equipped with the animal's identification *e.g.*, an integer. Grover's algorithm will reveal the criteria at the end of the computation, which reveals the identification as well. Thus, Grover's search outputs

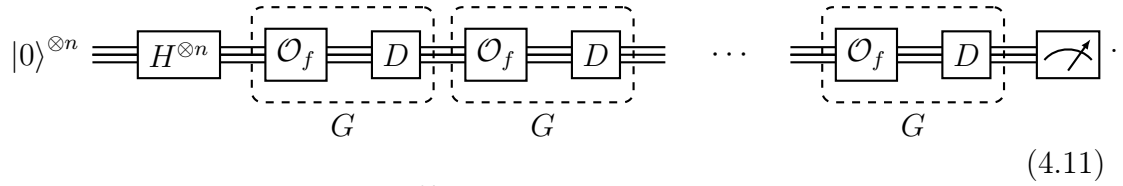
criteria x together with the identification of jellyfish *Turritopsis nutricula*

with high probability for a minimal number of oracle calls (optimal).¹⁹

For practical reasons, we reformulate the unstructured search problem (Problem 4.2) as follows.

Problem 4.3. *Alternative formulation of Grover's search problem. Given a database $\mathbf{d} \subseteq \Sigma^n$ with size N that is representable with $n := \lceil \log N \rceil$ qubits. We want to search a set of solutions $X \subset \Sigma^n$ within $\mathbf{d}$, given access to oracle $f : \Sigma^n \rightarrow \Sigma$, where $f(x) = 1, \forall x \in X$, and $f(x) = 0, \forall x \notin X$.*

Given that $N = 2^n$ for some $n \in \mathbb{N}$, we can write $\mathbf{d} = \{0, 1, \dots, N - 1\}$. The following circuit is Grover's algorithm solving Problem 4.3



Operator $\mathcal{O}_f$ is a phase oracle,²⁰

$$\mathcal{O}_f = I - 2 \sum_{x \in X} |x\rangle\langle x|, \quad (4.12)$$

D is the diffusion operator,

$$D = I - 2 |\Psi_{init}\rangle\langle\Psi_{init}|, \quad |\Psi_{init}\rangle = (H |0\rangle)^{\otimes n}, \quad (4.13)$$

¹⁹ Note that the question of how to effectively load the database (the oracle) into the quantum computer is still under investigation; some proposals are introduced [145, 146].

²⁰ The phase oracle model has a general form as Equation (4.3).

and $G := D\mathcal{O}_f$ is Grover iterator that is applied $\sim\sqrt{N}$ times; $|\Psi_{init}\rangle$ is defined in Equation (4.14).

The action of Grover's iterations can be described as follows. First, Hadamard gates transform the zero state into an equal superposition of quantum states $|\Psi_{init}\rangle$, where every state encodes a database entry.

$$H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle := |\Psi_{init}\rangle. \quad (4.14)$$

At this point, every database entry $|j\rangle$ is equally likely to appear with probability $p = 1/N$ — if we measure all qubits. Secondly, proceeding to the oracle application, we obtain

$$\mathcal{O}_f |\Psi_{init}\rangle = \left(I - 2 \sum_{x \in X} |x\rangle\langle x| \right) |\Psi_{init}\rangle = \frac{1}{\sqrt{N}} \left(- \sum_{x \in X} |x\rangle + \sum_{y \in \mathbf{d} \setminus X} |y\rangle \right). \quad (4.15)$$

Oracle $\mathcal{O}_f$ “marks” the items found in the database by inverting the phase. Given that there are m items found in database $\mathbf{d}$, where $m \leq |X|$, we rewrite Equation (4.15) as

$$\begin{aligned} \mathcal{O}_f &= - \underbrace{\sqrt{\frac{m}{N}}}_{=:a} \underbrace{\left(\frac{1}{\sqrt{m}} \sum_{x \in X} |x\rangle \right)}_{=:|\Psi_1\rangle} + \underbrace{\sqrt{\frac{N-m}{N}}}_{=:|\Psi_0\rangle} \underbrace{\left(\frac{1}{\sqrt{N-m}} \sum_{y \in \mathbf{d} \setminus X} |y\rangle \right)}_{=:|\Psi_0\rangle} \\ &= -\sqrt{a} |\Psi_1\rangle + \sqrt{1-a} |\Psi_0\rangle = -\sin \theta |\Psi_1\rangle + \cos \theta |\Psi_0\rangle, \end{aligned} \quad (4.16)$$

where $\theta = \arcsin \sqrt{a}$, $|\Psi_1\rangle$ is the normalized sum of solution states $\forall x \in \Sigma^n$ such that $f(x) = 1$, and $|\Psi_0\rangle$ is the normalized sum of non-solution states $\forall x \in \Sigma^n$ such that $f(x) = 0$. We can see that the oracle has partitioned the Hilbert space of $\mathbf{d}$ into the solution and non-solution subspaces.

Thirdly, applying the diffusion operator D results in

$$(I - 2 |\Psi_{init}\rangle\langle\Psi_{init}|)(-\sin \theta |\Psi_1\rangle + \cos \theta |\Psi_0\rangle) = \sin 3\theta |\Psi_1\rangle + \cos 3\theta |\Psi_0\rangle, \quad (4.17)$$

which one can think of as the new initial state prepared with angle 3θ rather than θ . At this point, we have applied one Grover iterator G , and have successfully brought initial state $|\Psi_{init}\rangle$ closer to the solution by 2θ . By applying G the second time, the state $|\Psi_{init}\rangle$ will be 5θ closer to the solution. These steps are illustrated in Figure 4.5.

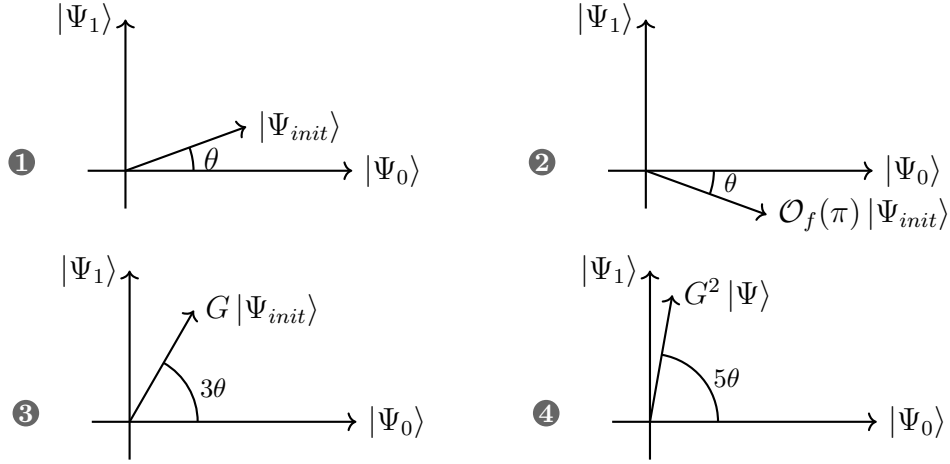


Figure 4.5: An illustration that represents the steps in Grover’s algorithm. ❶ The equal superposition of items lies in between the solution subspace spanned by $|\Psi_1\rangle$ and the non-solution subspace spanned by $|\Psi_0\rangle$. ❷ The oracle application reflects the state around $|\Psi_0\rangle$. ❸ The diffusion operation reflects the state around the initial state. Steps ❷ and ❸ is equivalent to one Grover iterator. ❹ Applying another Grover iterator results again in the addition of angle 2θ .

After one iteration, the probability of revealing the request is $\sin^2 3\theta =: p_1$; after k iterations,

$$p_k = \sin^2(\theta + 2k\theta). \quad (4.18)$$

Recall $\theta = \arcsin \sqrt{m/N}$, which is the initial angle before iterations. Thus, the highest success rate ($p_k = 1$) occurs at iteration

$$k = \frac{\pi}{4 \arcsin \sqrt{m/N}} - \frac{1}{2}. \quad (4.19)$$

Asymptotically speaking — the case of finding a needle in a haystack — $k \approx \sqrt{N}$, which gives the complexity of Grover’s algorithm $\mathcal{O}(\sqrt{N})$. Notice that, the success probability p_k is obtained only if we know m , *i.e.*, the number of solutions in the database.

When the queried item does not exist, *i.e.*, $X \cap \mathbf{d} = \emptyset$, we will have $\theta = 0$, thus, the Grover iterator is an identity. Therefore, in the case of no-instance (Equation (4.10)), the measurement outcome will be uniformly random since the final state equals to $|\Psi_{init}\rangle$.

We provide a quick demonstration of Grover’s algorithm using *Mathematica* in Section B.1. We employ a *Mathematica* library of Muñoz and Delgado [147], which is mentioned in Section A.3.

The success probability behaves sinusoidally with the number of iterations. The strategy of improving success probability by applying an operator several times is also known as *amplitude amplification*. However, this sinusoidal behaviour requires the algorithm to terminate at the right time — at the maximal success probability, which is achieved only if we know the number of solutions (non-zero) in the database.

Note that, due to rounding the value of k , we will not obtain exact probability $p_k = 1$, except for the case of finding one item in a four-entry database [148]. If N is small while not knowing the number of solutions m , we will be most likely to miss the maximal success probability; in this case, it is crucial to know the number of solutions in the database. However, if $m \ll N$, which results in an infinitely small θ , we are less likely to be far away from the maximal success probability. Therefore, Grover's algorithm performs better in finding a needle in a haystack than finding an elephant in a room.

4.3.2 Høyer's algorithm

In the previous subsection, we saw how the Grover iterator amplifies the success probability by moving the quantum state towards the solution with a constant step 2θ . Høyer provides a generalization of Grover's iterator, which allows us to approach the solution by an arbitrary step $\Theta \leq |2\theta|$, *i.e.*, within a range $\Theta \in [-2\theta, 2\theta]$. The step cannot be larger than 2θ because Grover's algorithm is optimal [144]; thus, Høyer's algorithm has the same complexity as Grover's algorithm. However, the freedom of choosing the step angles is useful to obtain an exact quantum search algorithm that is discussed in the next subsection. Høyer's amplitude amplification works as follows.

Given n qubits store all items in database $\mathfrak{d}$. The database has N items, where $2^{n-1} \leq N \leq 2^n$. The database entries can be any combination of numbers between 0 to $2^n - 1$,

$$\mathfrak{d} \in 2^{\{0,1,\dots,2^n-1\}}, \text{ where } |\mathfrak{d}| = N, \quad (4.20)$$

which is an element of a power set of $\{0, 1, \dots, 2^n - 1\}$.²¹

²¹ Compared to Grover's problem (Problem 4.3), we relax the requirement of the database content, since previously we need 2^n items with entries $0, \dots, 2^n - 1$.

First, we start with an arbitrary n -qubit product state $|i_0\rangle$.²² There exists a unitary $\mathcal{A}$ that transforms $|i_0\rangle$ into the equal superposition of all items in $\mathbf{d}$, i.e.,

$$\mathcal{A}|i_0\rangle = \frac{1}{\sqrt{N}} \sum_{j \in \mathbf{d}} |j\rangle := |\Psi_{init}\rangle, \quad (4.21)$$

which is the generalization of the initial state preparation of Grover's algorithm, where we have to set $\mathcal{A} = H^{\otimes n}$.

Second, given an oracle $f : \Sigma^n \rightarrow \Sigma$ (recall $\Sigma := \{0, 1\}$), where $\mathcal{O}_f |x\rangle = (-1)^{f(x)} |x\rangle$ for a solution x — or a yes-instance x in the promise problem formulation (Equation (4.10)). The oracle partitions the Hilbert space into the solution and non-solution subspaces.

Let us capture the solutions within the variable $X := \{x \in \mathbf{d} \mid f(x) = 1\}$, where $|X| = m$. We rewrite the initial quantum state as

$$|\Psi_{init}\rangle = \sqrt{a} |\Psi_1\rangle + \sqrt{1-a} |\Psi_0\rangle = \sin \theta |\Psi_1\rangle + \cos \theta |\Psi_0\rangle, \quad (4.22)$$

where $a := \frac{m}{N}$, $|\Psi_1\rangle := \frac{1}{\sqrt{m}} \sum_{x \in X} |x\rangle$, and $|\Psi_0\rangle := \frac{1}{\sqrt{N-m}} \sum_{y \in \mathbf{d} \setminus X} |y\rangle$.

Finally, we generalize Grover's iterator as an operator with two parameters $Q(\varphi, \psi)$, such that $Q(\varphi, \psi)$ brings the initial state $|\Psi_{init}\rangle$ closer to the solution state $|\Psi_1\rangle$ by angle $\Theta \in [-2\theta, 2\theta]$, where $\varphi, \psi \in [0, 2\pi)$. The iterator Q is defined as

$$Q(\varphi, \psi) := -D(\psi) \mathcal{O}_f(\varphi), \quad (4.23)$$

where $\mathcal{O}_f(\varphi)$ is a parameterized oracle

$$\mathcal{O}_f(\varphi) := I - (1 - e^{i\varphi}) |\Psi_1\rangle\langle\Psi_1|, \quad (4.24)$$

which “marks” every solution by adding the phase $e^{i\varphi}$, and $D(\psi)$ is the parameterized diffusion operator,

$$D(\psi) := \mathcal{A}(I - (1 - e^{i\psi}) |i_0\rangle\langle i_0|) \mathcal{A}^\dagger = I - (1 - e^{i\psi}) |\Psi_{init}\rangle\langle\Psi_{init}|, \quad (4.25)$$

where $\mathcal{A}$ is a preparation operator applied to a product state $|i_0\rangle$ — see Equation (4.21).

²² In Grover's algorithm, we are restricted to set $|i_0\rangle = |0\rangle^{\otimes n}$.

The iterator Q has the matrix form

$$Q(\varphi, \psi) = \begin{bmatrix} -a(1 - e^{i\psi}) - e^{i\psi} & (1 - e^{i\psi})e^{i\varphi}\sqrt{a(1-a)} \\ (1 - e^{i\psi})\sqrt{a(1-a)} & a(1 - e^{i\psi})e^{i\varphi} - e^{i\varphi} \end{bmatrix}, \quad (4.26)$$

which is expressed in the basis $\{|\Psi_0\rangle, |\Psi_1\rangle\}$; *e.g.*, $\langle\Psi_0|Q(\varphi, \psi)|\Psi_0\rangle = -a(1 - e^{i\psi}) - e^{i\psi}$.

Now, the question is how to configure $Q(\varphi, \psi)$ to implement the desired rotation angle. In the following, we derive the appropriate relation among variables φ, ψ , and Θ , such that $Q(\varphi, \psi)$ rotates the quantum state closer to the solution space by Θ — in a counter-clockwise sense. To do this, we treat $Q(\varphi, \psi)$ as an $SU(2)$ operator by considering $Q(\varphi, \psi)$ as a 2-dimensional unitary matrix that acts on an Hilbert space spanned by $|\Psi_0\rangle$ and $|\Psi_1\rangle$.

Given an arbitrary $SU(2)$ matrix with four parameters $\delta, \varphi_1, \varphi_2, \Theta \in [0, 2\pi)$,

$$U = e^{i\frac{\delta}{2}} \begin{pmatrix} e^{i\varphi_1} \cos(\Theta) & e^{i\varphi_2} \sin(\Theta) \\ -e^{i\varphi_2} \sin(\Theta) & e^{-i\varphi_1} \cos(\Theta) \end{pmatrix}, \quad (4.27)$$

we perform a parameter transformation $\varphi_1 = \mu + \nu$ and $\varphi_2 = \mu - \nu + \pi$, resulting in

$$U = e^{i\frac{\delta}{2}} \begin{pmatrix} e^{i\mu+i\nu} \cos(\Theta) & -e^{i\mu-i\nu} \sin(\Theta) \\ e^{i\mu-i\nu} \sin(\Theta) & e^{-i\mu-i\nu} \cos(\Theta) \end{pmatrix}. \quad (4.28)$$

Note that U has diagonal elements comprising cosine factors with different phases, whereas a counter-clockwise real rotation matrix with angle α is of the form

$$\begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}.$$

Our goal now is to represent $Q(\varphi, \psi)$ in a form of the rotation matrix above. For that, we obtain a rotation from an arbitrary $SU(2)$ matrix.

First, we impose an equality on the diagonal elements; it is satisfied when $\varphi_1 = -\varphi_1$, thus $\varphi_1 = 0$ or $\nu = -\mu$. With this setting, we obtain

$$e^{i\frac{\delta}{2}} \begin{pmatrix} \cos(\Theta) & -e^{i2\mu} \sin(\Theta) \\ e^{-i2\mu} \sin(\Theta) & \cos(\Theta) \end{pmatrix} =: \tilde{U} \quad (4.29)$$

Using variable changes $\delta/2 = \xi$ and $-2\mu = \zeta$ on $\tilde{U}$, we obtain

$$\tilde{U} = e^{i\xi} \begin{pmatrix} \cos(\Theta) & -e^{-i\zeta} \sin(\Theta) \\ e^{i\zeta} \sin(\Theta) & \cos(\Theta) \end{pmatrix}. \quad (4.30)$$

We now factorize $\tilde{U}$ separating its phases, and obtain

$$\tilde{U} = e^{i\xi} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & e^{i\zeta} \end{pmatrix}}_{P(\zeta)} \begin{pmatrix} \cos(\Theta) & -\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & e^{-i\zeta} \end{pmatrix}}_{-P(\zeta)}. \quad (4.31)$$

It is clear that $\tilde{U}$ performs a real rotation up to phase factors $P(\pm\zeta)$. Now we omit the global phase ξ . Thus, $P(-\zeta)\tilde{U}P(\zeta)$ results in a counter-clockwise real rotation matrix.

Let us discuss the same treatment on iterator $Q(\varphi, \psi)$ to obtain a rotation matrix up to phase factors. First, we set the diagonal elements of $Q(\varphi, \psi)$ in [Equation \(4.26\)](#) equal to one another

$$-a(1 - e^{i\psi}) - e^{i\psi} = a(1 - e^{i\psi})e^{i\varphi} - e^{i\varphi}. \quad (4.32)$$

By using Høyer's result in [\[127\]](#), suppose $\varphi \neq \pi$, the condition above is satisfied when

$$\tan(\varphi/2) = \tan(\psi/2)(1 - 2a). \quad (4.33)$$

We straightforwardly parameterize $Q(\varphi, \psi)$ corresponding to $\tilde{U}$, *e.g.*, by solving equation

$$\left\| (1 - e^{i\psi})\sqrt{a(1 - a)} \right\| = \sin(\Theta). \quad (4.34)$$

Using the condition in [Equation \(4.32\)](#), we obtain

$$\psi = \arccos \left(1 - \frac{\sin^2(\Theta)}{2a(1 - a)} \right) \quad (4.35)$$

$$\varphi = 2 \arctan(\tan(\psi/2)(1 - 2a)). \quad (4.36)$$

With these parameters, $Q(\varphi, \psi)$ performs a rotation through angle Θ up to phase factors $P(\pm\zeta)$. Using the form of [Equation \(4.30\)](#) and knowing ψ , phase ζ can be obtained as follows

$$\xi = \arg(-a(1 - e^{i\psi}) - e^{i\psi}) \quad (4.37)$$

$$\zeta = \arg \left((1 - e^{i\psi})\sqrt{a(1 - a)} - \xi \right). \quad (4.38)$$

Thus, using the defined angles above, Høyer's amplitude amplification $Q(\varphi, \psi)$ performs a counter-clockwise rotation through angle Θ up to phase factors $\pm\zeta$ in the basis spanned by $\{|\Psi_0\rangle, |\Psi_1\rangle\}$, namely

$$Q(\varphi, \psi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\zeta} \end{pmatrix} \begin{pmatrix} \cos \Theta & -\sin \Theta \\ \sin \Theta & \cos \Theta \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\zeta} \end{pmatrix}. \quad (4.39)$$

Grover's iterator is obtained when phases match, *i.e.*, $\psi = \varphi = \pi$, thus $G = Q(\pi, \pi)$. To obtain the correct rotation, we simply cancel the phase factors, *i.e.*,

$$P(-\zeta)Q(\varphi, \psi)P(\zeta) = -P(-\zeta)D(\psi)\mathcal{O}_f(\varphi)P(\zeta), \quad (4.40)$$

where $|\Theta| \leq 2\theta$, where $\theta = \arcsin \sqrt{m/N}$. Thus, applying $Q(\varphi, \psi)$ for k times will result in a total rotation through angle $k\Theta$; the total operator is

$$P(\zeta)Q(\varphi, \psi)^k P(-\zeta), \quad (4.41)$$

where the phase factors in between are cancelled out.

Note that the precise definitions of angles φ, ψ , and ζ are dependent of a , which is m/N . Thus, we need to know precisely the number of solution m in the database $\mathfrak{d}$, also when $m = 0$. On the contrary, Grover's algorithm can indicate if the solution is not in $\mathfrak{d}$ — as discussed in [Section 4.3.1](#).

4.4 THE EXACT GROVER-HØYER SEARCH ALGORITHM

Grover's algorithm is a fascinating algorithm to realize in an experiment since it provides a quadratic speedup compared to classical algorithms. Although Grover's algorithm is optimal to solve such a problem [\[144\]](#), its success probability varies for different cases. Here, our focus is to realize the success probability of 1, thus obtaining an exact quantum search algorithm. This setting underlies our motivation to study the growth of gate complexity by increasing the database's size ([Chapter 6](#)). Moreover, achieving certainty in a search algorithm becomes necessary in some setups such as quantum pattern recognition [\[149\]](#), distinguishing Boolean functions of different weights [\[150\]](#), searching over partial database [\[151\]](#), and database search with divide-and-verify strategy [\[152\]](#). Therefore, here, we narrow down the problem into finding a single item in a database with certainty, as formulated in [Problem 4.4](#).

Problem 4.4. *Finding an item in a database exactly. Given a database $\mathfrak{d}$, representable with n qubits, which comprises N distinct items. Find an item $x \in \mathfrak{d}$*

with success probability 1, given access to an oracle that realizes $f : \Sigma^n \rightarrow \Sigma$, where $f(z) = 1$ if $z = x$, and $f(z) = 0$ if $z \neq x$.

Grover's algorithm cannot solve [Problem 4.4](#), except when there are four items in the database [\[148\]](#), but there are others that can, which are inspired by Grover's algorithm. For instance, Long [\[153\]](#) and Liu [\[154\]](#) propose an exact quantum search algorithm that works for an arbitrary size of database and multiple marked items. Moreover, some generalizations of Grover's iterator can approach the solution arbitrarily, such as Høyer's amplitude amplification in [Section 4.3.2](#), and Long et al. that provides phase matching condition $\varphi = \psi$ — an approximation to Høyer's case. Later, Long's algorithm [\[153\]](#) has been generalized in [\[156\]](#) and can also acquire Høyer's phase condition in [Equation \(4.33\)](#).

We combine Grover's and Høyer's algorithms to reach the certainty. First, we run Grover's iterator as many times as possible without passing the solution state (the quantum final state will then be close to the solution, but not perfectly aligned). Afterward, we finish it with one application of Høyer's amplitude amplification to completely align the final state to the solutions. Note that we are talking about the state alignment shown in [Figure 4.5](#). The strategy is chosen based on the fact that Grover's algorithm is optimal [\[144\]](#), *i.e.*, the success probability is maximal for a minimal number of queries, and the iterator's gates are more straightforward than others.²³ We remark that our strategy has already been mentioned in [\[127\]](#); here, we provide an explicit algorithm, which we call *Grover-Høyer algorithm* in [Algorithm 4.1](#).

The Grover-Høyer algorithm runs on a quantum computer as follows

$$|0\rangle^{\otimes n} \equiv \boxed{\mathcal{A}} \equiv \boxed{\mathcal{O}_f} \boxed{\mathcal{D}} \equiv \cdots \equiv \boxed{\mathcal{O}_f} \boxed{\mathcal{D}} \equiv \boxed{\mathcal{O}_f(\varphi + \zeta)} \boxed{\mathcal{D}(\psi)} \equiv \boxed{\text{Measurement}}, \quad (4.42)$$

G
 G
 $Q(\varphi, \psi)$

where G signifies Grover's iterations, and $Q(\varphi, \psi)$ signifies Høyer's amplitude amplification. The curly-font means that their implementation is device-specific, for instance, a one-way quantum computer, a linear optic quantum computer, gates operation on superconducting qubits, and so forth. We call the set of device dependence operations as

$$\mathcal{B} := \{\mathcal{A}, \mathcal{O}_f, \mathcal{O}_f(\varphi + \zeta), \mathcal{D}, \mathcal{D}(\psi)\}. \quad (4.43)$$

²³ For instance, Grover's oracle marks the solutions with phase -1, whereas Høyer's oracle marks them with some phase $e^{i\varphi}$. The first is simpler to implement than the latter. This is shown in our direct count of the CNOT number for Grover and Høyer iterators in [Table 6.4](#).

Algorithm 4.1 [143] Grover-Høyer algorithm

Require: $\mathbf{d}, x$ **(1) Classical processing**1: $N \leftarrow |\mathbf{d}|$ **Ensure:** $x \in \mathbf{d}$ and $2^{n-1} < N \leq 2^n$ 2: $\theta \leftarrow \arcsin(1/\sqrt{N})$ 3: $k \leftarrow \lfloor (\frac{\pi}{2} - \theta)/2\theta \rfloor$ number of Grover runs4: $\Theta \leftarrow \frac{\pi}{2} - 2k\theta - \theta$ the remaining rotation5: $\psi, \varphi, \zeta \leftarrow \text{Equation (4.35), Equation (4.36), Equation (4.38)}.$ 6: $\mathcal{A} \leftarrow |\Psi_{in}\rangle (\langle 0|)^{\otimes n}$ 7: $\{D(\pi), D(\psi)\} \leftarrow \text{Equation (4.25)}$ 8: $\{\mathcal{O}_f(\pi), \mathcal{O}_f(\varphi + u)\} \leftarrow \text{Equation (4.24)}$ 9: Obtain a set of necessary operators $\mathcal{B}$ (Equation (4.43)), which are expressed within operations that can be done with the corresponding quantum computer.**(2) Quantum processing**10: $|\Psi\rangle \leftarrow \mathcal{A} |0\rangle^{\otimes n}$ 11: **for** $j = 1$ to k **do**12: $|\Psi\rangle \leftarrow \mathcal{D}(\pi)\mathcal{O}(\pi)|\Psi\rangle,$ 13: **end for**14: $|\Psi\rangle \leftarrow \mathcal{D}(\psi)\mathcal{O}(\varphi + \zeta)|\Psi\rangle$ 15: Measure $|\Psi\rangle$ 16: **Exit**

Operators in $\mathcal{B}$ are obtained from the following equations via some device-dependent construction. The preparation operator $\mathcal{A}$ is obtained from $\mathcal{A}$ (Equation (4.21)), the oracle $\mathcal{O}_f(\phi)$ is obtained from $\mathcal{O}_f$ (Equation (4.24)), and the diffusion operator $\mathcal{D}$ is obtained from D (Equation (4.25)).

We separate the notion of classical and quantum processing — as shown in Algorithm 4.1 — to show that there are some classical calculations going on beforehand. In the classical processing of Algorithm 4.1, the value of some variables must be known beforehand — *e.g.*, the number of the solutions — to run the algorithm.

The database choice restriction $2^{n-1} < N \leq 2^n$ is necessary only when considering resource efficiency. Choosing an index to represent an item is trivial; however, redundant qubits are costly, thus the index is chosen accordingly to minimize the number of qubits. We provide a toy model as a **Mathematica** code in Section B.3, where the Quantum package [147] is employed. As a toy model, such a resource restriction is omitted in the code; the code also works for multiple requests.

Note that we do not implement every operation composing a Høyer amplitude amplification in [Algorithm 4.1](#) (see line 14). The whole operation is supposedly $P(-\zeta)D(\psi)\mathcal{O}_f(\varphi)P(\zeta)$, as defined in [Equation \(4.40\)](#). Notice that the phase operator P ([Equation \(4.39\)](#)) and the oracle $\mathcal{O}_f$ ([Equation \(4.24\)](#)) share the same form, *i.e.*,

$$P(\zeta) = |\Psi_0\rangle\langle\Psi_0| + e^{i\zeta} |\Psi_1\rangle\langle\Psi_1| = I - (1 - e^{i\zeta}) |\Psi_1\rangle\langle\Psi_1|, \quad (4.44)$$

where here the identity operator is $I = |\Psi_0\rangle\langle\Psi_0| + |\Psi_1\rangle\langle\Psi_1|$. Thus, the phase correction $P(\zeta)$ can be absorbed into the oracle, *viz.*, $\mathcal{O}_f(\varphi)P(\zeta) = \mathcal{O}_f(\varphi + \zeta)$. Moreover, the last phase correction $P(-\zeta)$ can be omitted altogether, because at the last iteration — application of $Q(\varphi, \psi)$, the quantum state is spanned entirely by the solution space $|\Psi_1\rangle$, thus, $P(-\zeta)$ is equivalent to adding global phase $e^{-i\zeta}$.²⁴

The Grover-Høyer algorithm has the same query complexity as Grover’s algorithm, namely, $\mathcal{O}(\sqrt{N})$. If Grover’s algorithm requires k queries for a search problem, by the construction, the Grover-Høyer algorithm needs k or $k + 1$ queries for the same problem. However, if we are taking into account the real implementation resources, Høyer’s operators are more expensive to construct. For instance, we can compare how Grover’s and Høyer’s operators are built on a one-way quantum computer platform in [Figure 6.3](#), which shows that 32 nodes are needed to realize $D(\varphi)$ whereas 16 nodes are needed to realize $D(\pi)$.

4.5 SIMON’S ALGORITHM

Many quantum algorithms are tailored to solve particular problems, such that they surpass the classical counterparts, albeit those problems are artificial — we rarely find them in realistic settings. Moreover, some quantum algorithms turn out not to perform better than probabilistic classical algorithms. For example, we have already seen that this is the case of the Deutsch-Jozsa problem in [Problem 4.1](#), which does not separate **BQP** and **BPP**.

Simon’s problem [[158](#)] is the first quantum algorithm that shows a significant advantage over classical probabilistic algorithms, which separates **BQP** and **BPP** relative to an oracle. It is another black-box problem, which we formulate in [Problem 4.5](#). Although the problem appears to be unnatural, surprisingly, it is

²⁴ If we insist on applying the Høyer’s amplitude amplification entirely, together with the global phase, it means applying $-e^{-i(\zeta+\xi)}D(\psi)\mathcal{O}_f(\varphi + \zeta)$, demonstrated in [Section B.3](#), for the reason of getting an aesthetic output.

close to a significant real-world problem. It is the inspiration of Shor's integer factoring algorithm [15], which is at the heart of the RSA algorithm [137], a widely used cryptographic system today.

Problem 4.5. [157, 158] *Simon's problem.* Given black-box access to an oracle that encodes a function $f : \Sigma^n \rightarrow \Sigma^n$, the function f is promised to be either one-to-one or invariant under a nontrivial XOR-mask $s \in \Sigma^n$. The goal is to find out which category function f falls; if the second holds, determine mask s .

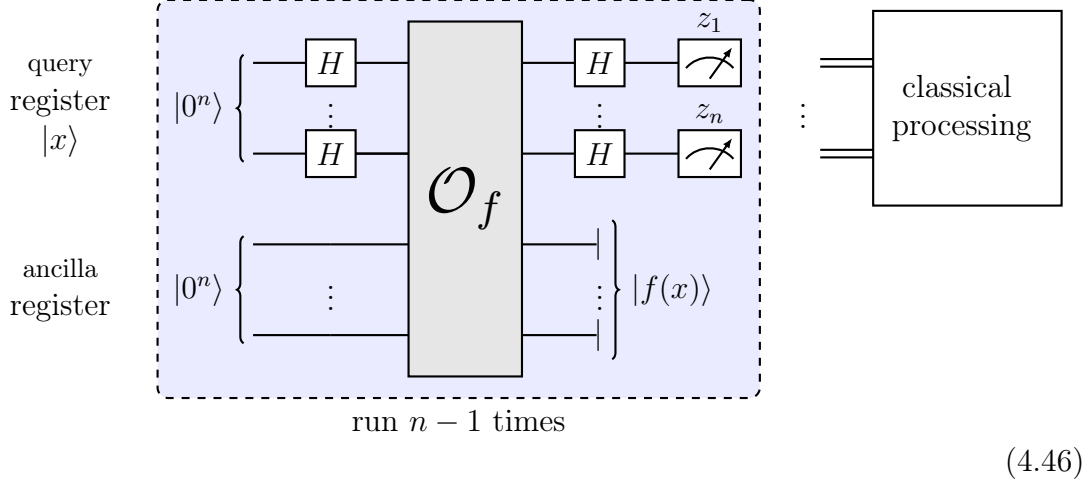
Notice that Simon's oracle does not belong to the standard oracle model (Section 4.1) since f outputs a bit string. Simon's problem is similar to Deutsch-Jozsa's problem, but with a slightly more complicated promise. The similarity is apparent in the promise problem representation,

$$\begin{array}{ll}
 \textbf{input} & f : \Sigma^n \rightarrow \Sigma^n \\
 \textbf{yes} & \exists s \in \Sigma^n \text{ and } s \neq 0, \text{ such that } \forall x, y \in \Sigma^n \text{ and } x \neq y, \\
 & f(x) = f(y) \iff x \oplus y = s \text{ (} f \text{ is two-to-one)} \\
 \textbf{no} & f \text{ is one-to-one.}
 \end{array} \tag{4.45}$$

Binary operation $\oplus$ signifies XOR (modulo 2 addition) operation. Notice that the no-instance case is equivalent to the case when the mask $s = 0^n$.

Any probabilistic TM needs an exponential number of queries, whereas a QTM — Simon's algorithm — can solve it efficiently [157]; this implies $\mathbf{BPP} \subseteq \mathbf{BQP}$ relative to the oracle. The intuition on exponential queries needed for a classical TM can be understood as follows. Consider an algorithm that queries the oracle on distinct inputs in a (uniformly) random fashion — there is no reason for sampling in a particular distribution as we only have a black-box. We tabulate every query together with its output. The algorithm halts if it outputs a value already in the table; say for inputs x and y , then f is two-to-one, and $s = x \oplus y$. Seemingly, $\mathcal{O}(\sqrt{2^n})$ queries are enough to reveal f ; this is based on the birthday paradox complexity [158, 159], i.e., the problem of finding two inputs that result in the same outputs of an unknown function has complexity $\mathcal{O}(\sqrt{2^n})$. Thus, a classical TM has complexity $\mathcal{O}(\sqrt{2^n})$, which is exponential, whereas the Simon algorithm solves

it efficiently with complexity $\mathcal{O}(n)$. Simon's algorithm is shown in Equation (4.46) below.



Simon's algorithm solves Problem 4.5 by repeatedly running the circuit in the blue box followed by some classical processing. The quantum oracle acts as

$$\mathcal{O}_f |x\rangle |y\rangle = |x\rangle |f(x) \oplus y\rangle \quad (4.47)$$

where $x, y \in \Sigma^n$; the first ket ($|x\rangle$) signifies the state assigned in the query register, and the second ket ($|y\rangle$) indicates the state assigned in the ancilla. The algorithm works as follows.

Let us examine the outcome of one iteration. The state after the oracle operation is

$$|0^n\rangle |0^n\rangle \xrightarrow{H^{\otimes n} \otimes I^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x \in \Sigma^n} |x\rangle |0^n\rangle \xrightarrow{\mathcal{O}_f} \frac{1}{\sqrt{2^n}} \sum_{x \in \Sigma^n} |x\rangle |f(x)\rangle. \quad (4.48)$$

Note that $f(x) \in \Sigma^n$, hence we cannot perform the phase kick-back.

We proceed by applying Hadamard gates to the first register; using Equation (4.8) we obtain

$$\frac{1}{2^n} \sum_{x \in \Sigma^n} \sum_{z \in \Sigma^n} (-1)^{x \cdot z} |z\rangle |f(x)\rangle, \quad (4.49)$$

where the product $x \cdot z$ is evaluated at modulo 2. Now, we are interested in the probability that corresponds to ending up in a bit string z . Thus, we rewrite the equation above to simplify the analysis,

$$\sum_{z \in \Sigma^n} |z\rangle \left(\frac{1}{2^n} \sum_{x \in \Sigma^n} (-1)^{x \cdot z} |f(x)\rangle \right). \quad (4.50)$$

First, if f belongs to **no**-instance, or equivalently, $s = 0^n$, the probability to obtain a bit string z is

$$p_z = \left\| \frac{1}{2^n} \sum_{x \in \Sigma^n} (-1)^{x \cdot z} |f(x)\rangle \right\|^2 = \frac{1}{2^{2n}} \underbrace{\left\| \sum_{x \in \Sigma^n} (-1)^{f^{-1}(x) \cdot z} |x\rangle \right\|^2}_{2^n} = \frac{1}{2^n}. \quad (4.51)$$

Since f is one-to-one, it means f is a permutation function. The uniform probability result $p_z = 1/2^n$ suggests that we will observe random output z in every iteration.

Second, if f belongs to **yes**-instance, which means $s \neq 0^n$, the situation becomes more interesting. Since f is two-to-one, every preimage has two elements. Given $f : X \rightarrow Y$, thus, for all $y \in Y$ there exists $x, \tilde{x} \in X$ where $x \neq \tilde{x}$, such that $f(x) = f(\tilde{x}) = y$; moreover $x \oplus \tilde{x} = s$. Again, we check the probability of an outcome z ,

$$\begin{aligned} p_z &= \left\| \frac{1}{2^n} \sum_{x \in \Sigma^n} (-1)^{x \cdot z} |f(x)\rangle \right\|^2 = \frac{1}{2^{2n}} \left\| \sum_{y \in Y} (-1)^{x \cdot z} |y\rangle + (-1)^{\tilde{x} \cdot z} |y\rangle \right\|^2 \\ &= \frac{1}{2^{2n}} \left\| \sum_{y \in Y} ((-1)^{x \cdot z} + (-1)^{(s \oplus x) \cdot z}) |y\rangle \right\|^2 = \frac{1}{2^{2n}} \left\| \sum_{y \in Y} ((-1)^{x \cdot z} + (-1)^{s \cdot z \oplus x \cdot z}) |y\rangle \right\|^2 \\ &= \frac{1}{2^{2n}} \underbrace{\left\| \sum_{y \in Y} (-1)^{x \cdot z} (1 + (-1)^{s \cdot z}) |y\rangle \right\|^2}_{2^{n+1}} = \begin{cases} 0, & \text{if } s \cdot z = 1 \\ \frac{1}{2^{(n-1)}}, & \text{if } s \cdot z = 0. \end{cases} \end{aligned} \quad (4.52)$$

Every z has the same probability outcome, *i.e.*, $1/2^{n-1}$, which implies that there are 2^{n-1} possible outcomes, namely the ones that fulfill condition $s \cdot z = 0$ — half of the set Σ^n . Note that the value 2^{n+1} above comes from the fact that $|Y| = 2^{n-1}$

because f is two-to-one. The probability $1/2^{n-1}$ is obtained from the following equations

$$\frac{1}{2^{2n}} \left\| \sum_{y \in Y} (-1)^{x \cdot z} |y\rangle \right\|^2 = \frac{2^2}{2^{2n}} \underbrace{\left\| \sum_{y \in Y} (-1)^{x \cdot z} |y\rangle \right\|^2}_{2^{n-1}} = \frac{1}{2^{n-1}}. \quad (4.53)$$

Now we arrive at the classical processing. Notice that for each case (whether f is one-to-one or two-to-one), it fulfills $s \cdot z = 0$ for all outcome z . This information is sufficient to infer the mask s . If we repeat the measurement $n - 1$ times, with high probability we obtain $n - 1$ distinct outputs $\{y_1, \dots, y_{n-1}\}$ and $n - 1$ linearly independent equations

$$\begin{aligned} y_1 \cdot s &= 0 \\ &\vdots \\ y_{n-1} \cdot s &= 0, \end{aligned}$$

which is efficiently solvable. Solving those $n - 1$ linearly independent equations, we obtain a candidate $\tilde{s}$. Thus, we query for x and $x \oplus \tilde{s}$ if the outputs are the same, then $s = \tilde{s}$, otherwise f is one-to-one.

Here, we only need to query the oracle $n + 1$ times in the best case, or else a few more queries will likely yield $n - 1$ linearly independent equations. Thus, the oracle complexity is $\mathcal{O}(n)$, which indicates an efficient solver. Therefore, the existence of Simon's algorithm to solve Simon's problem separates **BPP** and **BQP**. Moreover, if the oracle has internal randomness, Simon's problem becomes unsolvable classically [43]. If the oracle randomly applies different f on each query, a classical TM will output random number on each call, whereas Simon's algorithm is independent of the choice of f ; it provides a stronger separation: infinitely many classical queries will not gain any information, whereas $\Theta(n)^{25}$ quantum queries solve the problem [43]. The oracle with an internal randomness setting is useful in describing noisy oracles.

25 Theta-notation (Θ) describes the complexity of an algorithm when the upper bound ($\mathcal{O}$) and the lower bound (Ω) coincide.

BLIND ORACULAR QUANTUM COMPUTATION

Oracle construction in quantum algorithms provides an essential conceptual framework for understanding quantum speedups, *e.g.*, by allowing separation in complexity classes [20].¹ For instance, a recent result shows that, $\text{BQP}^{\mathbf{A}} \not\subseteq \text{PH}^{\mathbf{A}}$ [42] for an oracle $\mathbf{A}$, which implies quantum computers still have an advantage over classical computers even if the polynomial hierarchy collapses. Moreover, some generalizations of oracle model [43, 139] allow one to construct oracle problems that are efficiently solvable in a quantum setting, while it is impossible for infinite classical queries to gain any information. However, some quantum speedups are concluded under assumption of perfect oracles. An unexpected consequence might emerge when the oracle deviates from the ideal definition; for instance, in the Grover algorithm, the quantum speedup becomes impossible if the oracle has a small probability of failure in every call [160]. Moreover, a quantum oracle is commonly used to model a quantum Turing Machine that evaluates an unknown classical function (see Section 4.1). Thus, it is natural to separate quantum oracles from quantum algorithms.

In our work, we aim to introduce a multiparty setting where the oracle paradigm is physically meaningful; we call our scheme *Blind Oracular Quantum Computation* (BOQC). We bring the idea of oracle separation into a more realistic situation, where there is a party outside the computation, with access to the oracle. Also, we take into account privacy in our implementation.

The BOQC is a blind quantum computation (BQC) with a three-party setting. The protocol considers a client (Alice), who wants to delegate her oracular quantum algorithm to an untrustworthy server (Bob), with the help of a trustworthy third party oracle (Oscar) who has access to an oracle that evaluates a classical function. We can consider Oscar as a party with a high classical power. The BOQC protocol

¹ The notion of oracle, computation speedup, and several complexity classes are discussed briefly in Section 4.1.

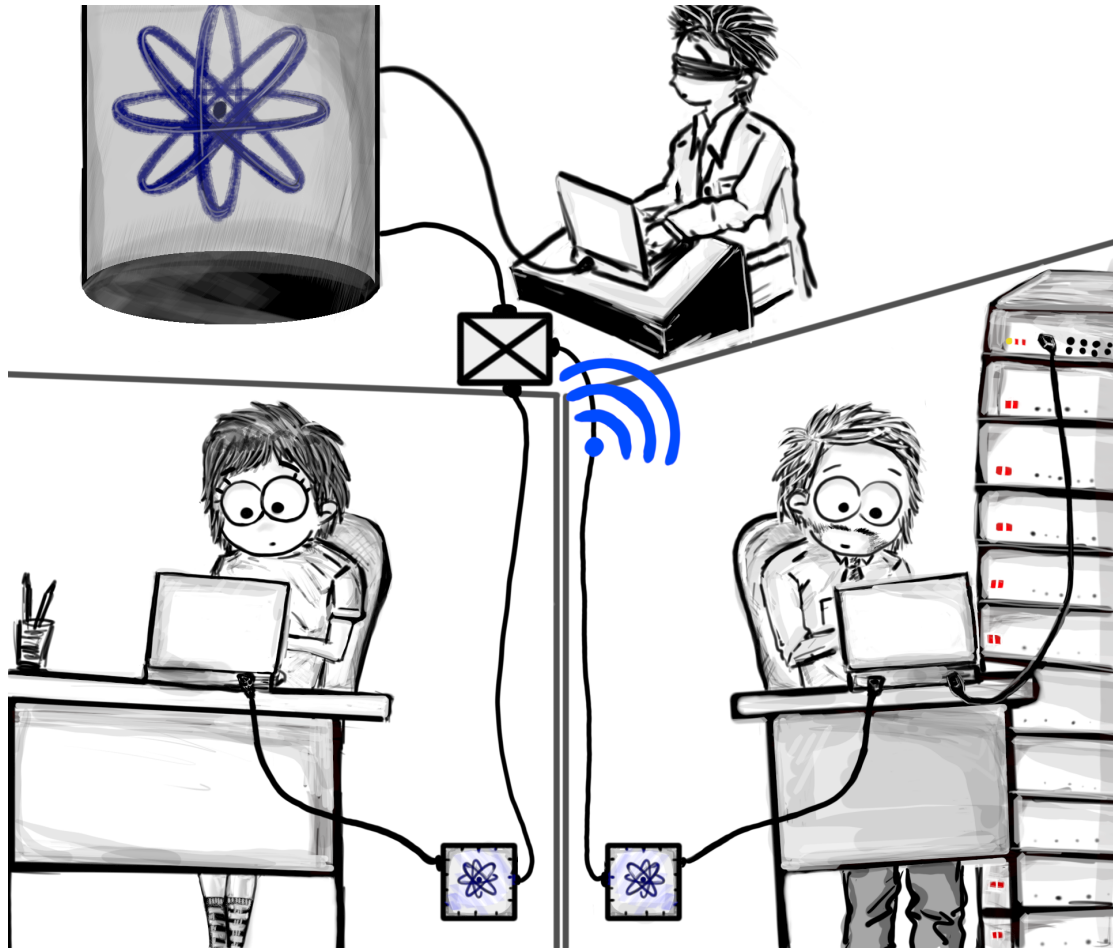


Figure 5.1: Alice (left) wants to run a private oracular quantum algorithm, but she has neither a powerful quantum computer nor the knowledge needed to construct her oracle function. On the other hand, Oscar (right) has the device that evaluates the oracle function f , e.g., it computes the prediction of stocks, gives access to a database, or does massive classical computations. Bob (top) has a quantum computer with a full quantum power, offered to Alice and Oscar as a service. However, Alice and Oscar do not trust Bob’s quantum computer, while building a reliable quantum computer is very hard. As they do not risk revealing information about their computation while running it on Bob’s quantum computer, they run their computation within the BOQC scheme, where Bob is blinded to the client’s computation. All players are connected to a classical channel — signified with the wireless symbol. It is important to remark that Alice and Oscar need a small quantum power — indicated with small boxes in the bottom, which makes them “almost classical” clients. Small quantum power is necessary for a client because blindness is impossible for a purely classical client [161].

is designed to execute a family of oracular quantum algorithms; an extensive catalog on oracular quantum algorithms can be found in [38]. Figure 5.1 illustrates a situation that pictures Alice, Bob, and Oscar running an oracular quantum algorithm in the BOQC scheme.

More details about the BOQC scheme will be elaborated in this chapter. We start with a discussion on the protocol, followed by introducing optimization in the scheme, and finally we prove the correctness and security of our protocols.

5.1 THE BOQC PROTOCOL

BOQC is a BQC with a three-party setting whose protocol is extended from the UBQC (Universal Blind Quantum Computation),² where there are two clients instead of one. Since there are two clients, we need some means to join their quantum computations with minimal communication between them, while maintaining their computation privacy. The BOQC protocol provides a means for Alice and Oscar to securely delegate their computation, where they only need to communicate once. Moreover, the BOQC protocol promises composable security.

Since the BOQC scheme works within the 1WQC paradigm,³ Alice's computation is represented as a graph state and measurement angles. We choose the 1WQC model to underlie our protocol due to the nature of unitaries on graph states, *viz.*, combining two quantum operators (graphs) is performed by directly connecting the graphs. Thus, outsourcing a quantum operation to another party means outsourcing the corresponding graph; in contrast, it is tricky to outsource a unitary in the gate model.

We assume that Alice and Oscar to possess the same level of quantum power, but Oscar is classically more powerful than Alice since he has access to oracles; this level quantum power is illustrated in Figure 5.1, portrayed with small chips of Alice and Oscar. Oscar's oracles implement some classical functions, and they admit oracle models that are discussed in Section 4.1.

In BOQC, we indicate players Alice, Bob, and Oscar with $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{O}$, respectively. Consider Alice's computation $\{(\mathcal{A}, I, O), f_{\mathcal{A}}, \phi, \rho^{in}\}$ and Oscar's computation $\{\mathcal{O}, f_{\mathcal{O}}, \psi\}$. Their computations must be independent and they do not communicate their computations. They join their computation via procedures given in Definition 5.1, *i.e.*, *pre-protocol steps*. Notice that Alice's computation

² We discuss the UBQC in Section 3.1.

³ See Section 2.1 for details of the 1WQC (One-Way Quantum Computer).

admits 1WQC computation in [Definition 2.17](#) and Oscar does not have his own input and output because he is an oracle.

Definition 5.1. *Pre-protocol steps.* Given that Oscar agreed to provide oracle information for a quantum computation that Alice wishes to run, the following steps are done via an authentic channel before starting a BOQC protocol:

- (B1) Alice and Oscar determine the size of bit string b ; the string must be long enough to indicate all possible measurement angles. The allowed angles is set to be $\Omega = \{\frac{\pi k}{2^{b-1}}\}_{0 \leq k < 2^b}$. It must hold that $\phi_i, \psi_i \in \Omega$.
- (B2) Alice and Oscar join their graphs in the following way. Given that Alice's oracular quantum algorithm needs k oracle queries, she marks each query as a black box on her graph $\mathcal{A}$. Given that Oscar's graph $\mathcal{O}$ is a graph with k components, he sends Alice $\{\mathcal{O}, f_{\mathcal{O}}\}$. Alice joins their graphs with a connection C by replacing each black box in $\mathcal{A}$ with a component of $\mathcal{O}$ according to C ; she obtains $\mathcal{G} = \mathcal{A} \cup_C \mathcal{O}$, and compute the total flow $(f, \succ)$ for the entire graph $\mathcal{G}$. The connection C is valid when the resulting graph $\mathcal{G}$ has flow.⁴
- (B3) Alice determines a total ordering $>$ that is consistent with the partial ordering $\succ$. This step is optional for the BOQC protocols ([Protocols 5.1](#) and [5.3](#)), but it is necessary for optimized protocols such as BOQCo protocols ([Protocols 5.4](#) and [5.5](#)).
- (B4) Alice publicly informs Bob of $\{(\mathcal{G}, \tilde{I}, \tilde{O}), V_{\mathcal{A}}, V_{\mathcal{O}}, \succ, >, b\}$, where $\tilde{I} \subseteq I$ is a set of nodes assigned with quantum input ρ^{in} , $\tilde{O} \subseteq O$ is a set unmeasured nodes that will be sent to Alice, $V_{\mathcal{A}} \equiv V(\mathcal{A})$, and $V_{\mathcal{O}} \equiv V(\mathcal{O})$. If $\tilde{I} \neq \emptyset$ and $\tilde{O} \neq \emptyset$, then $\tilde{I} \cap V_{\mathcal{O}} = \emptyset$ and $\tilde{O} \cap V_{\mathcal{A}} = \emptyset$, respectively; simply put, Oscar does not give quantum input and does not receive quantum output. The total order $>$ is given if step (B3) is executed.

We deviate from the convention of 1WQC computation in [Definition 2.17](#); here we introduce additional variables $\tilde{I} \subseteq I$ and $\tilde{O} \subseteq O$. Variable $\tilde{I}$ indicates the nodes assigned with quantum input; Bob needs to know $\tilde{I}$ and $\tilde{O}$ in order to prepare the quantum channel that can transmit arbitrary quantum states beyond the form of $|+\alpha\rangle$ for $\alpha \in [0, 2\pi)$. If Alice's input is entirely classical, $\tilde{I} = \emptyset$, but I cannot be $\emptyset$ in order to admit a proper flow in [Definition 2.15](#). Nodes $I \setminus \tilde{I}$ act as auxiliary nodes since they have the form of state $|+\alpha\rangle$. This convention is useful when Alice needs ancilla qubits in her computation, which add the dimension of graph state;

⁴ See [Example 5.1](#) that shows the process of this step explicitly.

here, the ancillae are counted as auxiliary nodes. Alice can also ask a subset of output nodes $\tilde{O} \subseteq O$ as quantum output that she receives; it means $O \setminus \tilde{O}$ are auxiliary nodes. The case $\tilde{O} = \emptyset$ means Alice's output is entirely classical; again, we must fulfill $O \neq \emptyset$ to admit the proper flow in [Definition 2.15](#).

For simplicity without losing generality, the input nodes $I \setminus \tilde{I}$ has zero states by default, *i.e.*, they are initialized to state $|+\rangle$. In this thesis, we provide explicit schemes for two cases: input and output are entirely quantum ([Protocols 5.1](#) and [5.4](#)) and entirely classical ([Protocols 5.3](#) and [5.5](#)).

The BOQC protocol with entirely quantum input and output is shown in [Protocol 5.1](#), where $\tilde{I} = I$ and $\tilde{O} = O$. This case is general, such that other cases can be reduced to this case. For instance, if Alice's input is a bit string $c_1 c_2 \dots c_n$ where $c_i \in \{0, 1\}$, we encode it as a state $\bigotimes_i c_i |+\rangle$ in the implementation, but $\rho_A^{in} = \bigotimes_i |c_i\rangle\langle c_i|$ in the security model.

In the language of AC (Abstract Cryptography), we denote the BOQC protocol ([Protocol 5.1](#)) as $\pi_{boqc} = \{\pi_A, \pi_B, \pi_O\}$, where π_A , π_B , and π_O are converters of Alice, Bob, and Oscar, respectively; these converters can be considered as the process carried out by the corresponding party in [Protocol 5.1](#).

We explicitly include the adaptive measurement scheme in [Protocol 5.1](#), where some notation is taken over from [88]. [Protocol 5.1](#) takes input from Alice $\{(\mathcal{G}, I, O), f, \phi, \rho_A^{in}\}$, and input from Oscar $\{\psi\}$; note that flows f_A and f_O are captured in f . Alice prepares the input state ρ_A^{in} on nodes $\tilde{I}$ and receives the final state of nodes $\tilde{O}$; in this protocol, we set $\tilde{I} = I$ and $\tilde{O} = O$. The algorithm's outcome is $\mathcal{E}(\rho_A^{in})$, where $\mathcal{E}$ represents the resulting CPTP map of the entire computation.

The BOQC protocol employs resources $\mathcal{R}$ (AC notation) comprising a secure key channel between Alice and Oscar, two-way classical channels between clients and Bob, and two-way quantum channels between clients and Bob. Note that, if $\tilde{O} = \emptyset$, one-way quantum channels are sufficient.⁵

The heart of our strategy in joining computations of Alice and Oscar is the establishment of symmetric keys r and t between Alice and Oscar. The key r hides measurement outcomes and t hides the quantum input state⁶ by randomly flipping

¹ It means the loop goes through all nodes in $V \setminus O$ following a total ordering $>$ that is consistent with $\succ$; the total ordering $>$ is implicit there.

² Operator tr_j is a partial trace, where subsystem j is traced-out; thus $\text{tr}_{I \setminus i}[\rho_A^{in}]$ indicates that subsystem i remains, where $i \in I$.

⁵ The details on requirements of quantum power will be discussed in [Section 5.3](#).

⁶ This hiding strategy is called QOTP (Quantum One Time Pad), which is discussed in [Section 3.1](#).

Protocol 5.1 BOQC ($\pi_{boqc} = \{\pi_A, \pi_B, \pi_O\}$)

Alice's input: $\{(\mathcal{G}, I, O), f, \phi, \rho_A^{in}\}$ $\triangleright \tilde{I} = I$ and $\tilde{O} = O$

Oscar's input: $\{\psi\}$

Alice's output for an honest Bob: $\rho_A^{out} = \mathcal{E}(\rho_A^{in})$

Assumptions and conventions:

(I) Alice ($\mathcal{A}$) and Oscar ($\mathcal{O}$) have performed pre-protocol steps in [Definition 5.1](#); Bob knows $\{(\mathcal{G}, \tilde{I}, \tilde{O}), V_A, V_O, \succ, >, b\}$. Here, we set $\tilde{I} = I$ and $\tilde{O} = O$. Recall $\tilde{O} \cap V_O = \emptyset$ (quantum outputs are held by Alice) and $\Omega = \{\frac{\pi k}{2^{b-1}}\}_{0 \leq k < 2^b}$.

(II) $s_{invf(i)} = 0, \forall i \in I$ and $t_i = 0, \forall i \in I^c$.

(III) $invf(i) \equiv f^{-1}(i)$, $z(i) := \bigoplus_{k \prec i, i \in N_{\mathcal{G}}(f(k))} s_k$, and $t(i) := \bigoplus_{k \in I, i \in N_{\mathcal{G}}(k)} t_k$.

0 Pre-preparation

- 1: Alice and Oscar receive keys r, t via a secure key channel, where $r_i \in \{0, 1\}$, $i \in O^c$ and $t_j \in \{0, 1\}$, $j \in I$.

1 State preparation

- 2: **for** $i \in V \setminus O$ following partial ordering $\succ$ **do**¹ $\triangleright$ it may follow ordering $>$
- 3: **if** $i \in V_A$ **then**
- 4: Alice chooses $\alpha_i \in \Omega$ at random.
- 5: **if** $i \in I$ **then**
- 6: Alice applies $Z_i(\alpha_i)X_i^{t_i}$ to $\text{tr}_{I \setminus i}[\rho_A^{in}]$ and sends it to Bob.²
- 7: Alice updates angles:

$$\begin{aligned} \phi_i &= (-1)^{t_i} \phi_i \\ \phi_j &= \phi_j + t_i \pi, \quad \forall j \in N_{\mathcal{G}}(i) \cap V_A. \end{aligned}$$

- 8: Oscar updates angles $\psi_j = \psi_j + t_i \pi$, $\forall j \in N_{\mathcal{G}}(i) \cap V_O$.
- 9: **else**
- 10: Alice prepares $|+\alpha_i\rangle_i$ and sends it to Bob.
- 11: **end if**
- 12: **else if** $i \in V_O$ **then**
- 13: Oscar prepares $|+\beta_i\rangle_i$ and sends it to Bob, where $\beta_i \in \Omega$ is chosen at random.
- 14: **end if**
- 15: **end for**
- 16: For all $i \in O$, Bob prepares $|+\rangle_i$.

2 Graph state formation

- 17: Bob applies entangling operator $E_{\mathcal{G}}$ defined in [Equation \(2.47\)](#).

3 Classical interaction and measurement

```

18: for  $i \in V \setminus O$  which follows partial ordering  $\succ$  do
19:   if  $i \in V_{\mathcal{A}}$  then
20:     Alice computes  $\phi'_i = (-1)^{s_{\text{invf}(i)}} \phi_i + z(i)\pi$ .
21:     Alice computes  $\delta_i := \phi'_i + \pi r_i + \alpha_i$ , and sends Bob  $\delta_i$ .
22:   else if  $i \in V_{\mathcal{O}}$  then
23:     Oscar computes  $\psi'_i = (-1)^{s_{\text{invf}(i)}} \psi_i + z(i)\pi$ .
24:     Oscar computes  $\delta_i := \psi'_i + \pi r_i + \beta_i$ , and sends Bob  $\delta_i$ .
25:   end if
26:   Bob measures qubit  $i$  in  $|\pm_{\delta_i}\rangle$  basis, then sends Alice and Oscar the outcome  $\tilde{s}_i$ .
27:   Alice and Oscar set  $s_i = \tilde{s}_i \oplus r_i$ .
28: end for
4 Output transmission and correction
29: Bob sends Alice output qubits  $\rho_{\mathcal{B}}^{\text{out}}$  (all qubits  $i \in O$ ).
30: Alice corrects the final output  $P(\rho_{\mathcal{B}}^{\text{out}}) =: \rho_{\mathcal{A}}^{\text{out}}$ , where  $P \equiv \bigotimes_{i \in O} X_i^{s_{\text{invf}(i)} + t_i} Z_i^{z(i) + t(i)}$ .

```

the input qubits; thus, key t is needed if the input is quantum, *i.e.*, $\tilde{I} \neq \emptyset$. The minimal communication between Alice and Oscar can be identified from the use of resource $\mathcal{R}$, which is pictured in Figure 5.2 as an interaction diagram. The figure shows that there is almost no communication between Alice and Oscar, except for establishing a symmetric key between them in the very beginning. Therefore, the random key r, t is the only secret shared between Alice and Oscar.

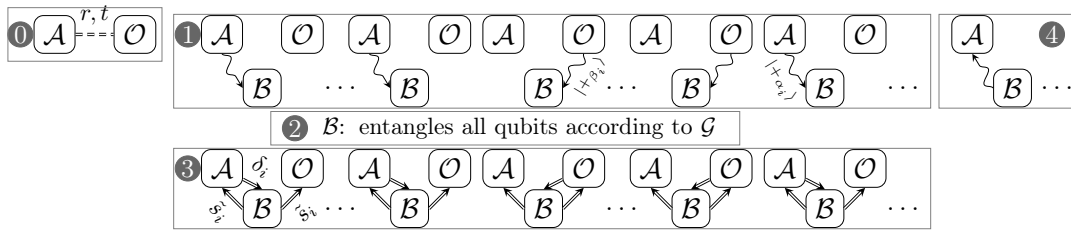
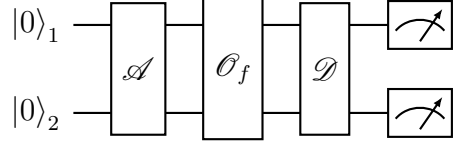


Figure 5.2: [143] Communication diagram in the BOQC scheme. It also gives details of how the resources in Protocol 5.1 are used; these resources correspond to real-world resources $\mathcal{R}$ discussed in Section 5.2. The initials $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{O}$ indicate players Alice, Bob, and Oscar, respectively. The double dashed line represents a secure key channel, the double lines denote insecure classical channels, and the wavy lines denote quantum channels. Every circled number i corresponds to a part in Protocol 5.1 identified with number i . Notice that, Alice needs a two-way quantum channel as she receives the final quantum output.

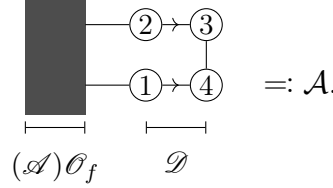
Example 5.1. Alice wants to run a 2-qubit Grover algorithm (see [Section 4.3](#)) within a BOQC scheme and Oscar has the 4-element database. Alice and Oscar run the pre-protocol steps from [Definition 5.1](#) as shown below.

The 2-qubit Grover algorithm consists of only one Grover iteration that is realized with circuit

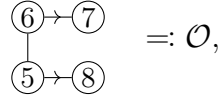


where $\mathcal{A}$, $\mathcal{O}_f(\pi)$ and $\mathcal{D}(\pi)$ indicate preparation, oracle, and diffusion unitaries respectively; we refer to [Section 4.3](#) for details on these unitaries.

It turns out that Alice does not need any input from herself, thus she sets $\tilde{I} = \emptyset$. Also, she sets $\tilde{O} = \emptyset$ since the output is classical. The following figure shows Alice's graph state, where the oracle is marked as a black-box and the flow is denoted by arrows,

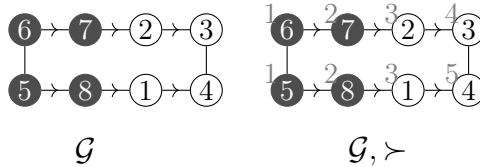


First Alice and Oscar agreed upon the bit string size, e.g., $b = 4$. Secondly, Oscar tells Alice his graph and flow $\{\mathcal{O}, f_{\mathcal{O}}\}$, where $\mathcal{O}$ is graph with one component as shown below,



and the flow is denoted with arrows.

Now, we have $\mathcal{A} = (\{1, 2, 3, 4\}, \{(2, 3), (1, 4), (3, 4)\})$ and $\mathcal{O} = (\{5, 6, 7, 8\}, \{(2, 3), (3, 4), (1, 4)\})$. Alice obtains the total graph $\mathcal{G} = \mathcal{A} \cup_C \mathcal{O}$ with $C = \{(7, 2), (8, 1)\}$ and the total flow $(f, \succ)$, pictorially



where $I = \{5, 6\}$, $\tilde{I} = \emptyset$, $O = \{3, 4\}$, $\tilde{O} = \emptyset$, $V_A = \{1, 2, 3, 4\}$, $V_O = \{5, 6, 7, 8\}$, partial ordering $\{5, 6\} \succ \{7, 8\} \succ \{1, 2\}, \succ \{3, 4\}$, and a total ordering, e.g., $5 < 6 < 7 < 8 < 1 < 2 < 3 < 4$. Finally, Alice publicly tells Bob $\{(\mathcal{G}, \emptyset, \emptyset), V_A, V_O, \succ, \succ, 4\}$.

In this case, neither Alice nor Oscar provide classical or quantum input, and $I \subset V_O$; this is consistent with [Definition 5.1](#) since $\tilde{I} = \emptyset$. Instead, here the input is implicit, i.e., state zeros $|00\rangle\langle 00|$.

Using the AC framework defined in [Section 3.2](#), a composable security definition of a protocol is obtained when we achieve: correctness ([Definition 3.2](#)) and security ([Definition 3.3](#)). Correctness is the statement of every party being honest, whereas security is the statement in the presence of an adversary — when Bob is cheating. One may define the correctness arbitrarily; for us, BOQC protocol is correct when outputs of the BOQC scheme are identical to outputs of the 1WQC scheme for the same inputs. Our correctness statement is similar to the statement of the UBQC correctness [[113](#)]. The formal correctness statement is provided in [Theorem 5.2](#); however, we first consider the following theorem ([Theorem 5.1](#)).

Theorem 5.1. *The BOQC protocol π_{boqc} defined in [Protocol 5.1](#) delegates a computation with the isometry defined in [Theorem 2.1](#) ([Section 2.4](#)) for the same computation, without requiring Alice and Oscar to communicate their computations to each other.*

Proof. Let Alice's input be $\{(\mathcal{G}, I, O), f, \phi, \rho_A^{in}\}$ and Oscar's input be $\{\psi\}$. If we join both computations, the total algorithm will be $\{(\mathcal{G}, I, O), f, \theta, \rho_A^{in}\}$, where $\theta = \phi \cup \psi$ contains measurement angles for all nodes $i \in O^c$. To achieve the same isometry as the one defined in [Theorem 2.1](#), we need to obtain an identical pattern as in [Theorem 2.1](#). Thus, we need to show two things here: for an input $\{(\mathcal{G}, I, O), f, \theta, \rho_A^{in}\}$ of [Protocol 5.1](#), the resulting pattern must be

$$\bigotimes_{i \in O^c} (X_{f(i)}^{s_i} \prod_{k \in N_{\mathcal{G}}(f(i)) \setminus \{i\}} Z_k^{s_i} M_i^{\theta_i}) E_{\mathcal{G}} N_{I^c}^0 =: \mathfrak{P}_{1wqc},$$

while Alice and Oscar keep their computation themselves during the whole process.

First, we consider Alice and Oscar as one client called super-Alice. Omitting the randomness $r_i = \alpha_i = \beta_i = 0$,⁷ running a computation with Protocol π_{boqc} yields pattern

$$\mathfrak{P}_{boqc} = \bigotimes_{j \in O} X_j^{s_{\text{inv}f(j)}} Z_j^{z(j)} \succcurlyeq \prod_{i \in O^c} M_i^{\theta_i} X_i^{s_{\text{inv}f(i)}} Z_i^{z(i)} E_{\mathcal{G}} N_{I^c}^0 \stackrel{\text{Lem. 2.1}}{=} \mathfrak{P}_{1wqc},$$

where $z(i) := \bigoplus_{k \prec i, i \in N_{\mathcal{G}}(f(k))} s_k$ and Lemma 2.1 is stated in Section 2.5. Thus, in the absence of randomness, $\mathfrak{P}_{boqc} = \mathfrak{P}_{1wqc}$.

Second, consider $\gamma = \alpha \cup \beta$ that contains random angles produced by Alice and Oscar. Including γ and r in the computation,⁸ we obtain pattern

$$\mathfrak{P}_{boqc} := \succcurlyeq \prod_{i \in O^c} (X_{f(i)}^{s_i} \prod_{k \in N_{\mathcal{G}}(f(i)) \setminus \{i\}} Z_k^{s_i + r_i} M_i^{\theta_i + \gamma_i + \pi r_i}) E_{\mathcal{G}} N_{I^c}^{\gamma}. \quad (5.1)$$

Thus, using the rules in Equation (2.32) from Section 2.3 ($M_i^{\theta} X_i^s = M_i^{(-1)^s \theta}$ and $M_i^{\theta} Z_i^s = M_i^{\theta + s\pi}$) and the commutation relation $E_{\mathcal{G}} N_i^{\gamma_i} = E_{\mathcal{G}} Z_i(\gamma_i) N_i^0 = Z_i(\gamma_i) E_{\mathcal{G}} N_i^0$, random variable γ_i is cancelled out.⁹ The random angle πr_i is canceled out by flipping the measurement outcome, $s_i = \tilde{s}_i \oplus r_i$, as it appears in the superscript of Z -corrections. Thus, equality $\mathfrak{P}_{boqc} = \mathfrak{P}_{1wqc}$ is maintained.

Third, we show that the encryption of the quantum inputs $X_i^{t_i} Z_i(\alpha_i)(\rho_{\mathcal{A}}^{in})$ is perfectly decrypted during the process. Random variable α_i is cancelled out in the same manner as removing γ above. The random variable t_i is cancelled out by flipping the sign of the measurement angle¹⁰ $M_i^{\theta} X_i \rho_{\mathcal{A}}^{in} = M_i^{-\theta} \rho_{\mathcal{A}}^{in}$. Thus, we conclude that $\rho_{\mathcal{A}}^{in}$ is intact.

Finally, we divide super-Alice into Alice and Oscar. Calculating corrected angles is done separately, since they know the real measurement outcomes $s_i = \tilde{s}_i \oplus r_i$. Cancelling out the z -rotations $R_i^z(\alpha_i)$ or $R_i^z(\beta_i)$ is also done separately, that is clearly shown from lines 21 and 24. Thus, Alice and Oscar keep their measurement angles and the random variables to themselves. $\square$

Theorem 5.1 confirms that the BOQC protocol indeed realizes the desired 1WQC computation. Alice and Oscar only need to establish symmetric keys r and t in the

⁷ In this condition, the computation is within the 1WQC scheme.

⁸ At this stage, it can be seen as a UBQC computation with graph $\mathcal{G}$ instead of the brickwork graph (see Figure 3.4 in Section 3.1).

⁹ Within the gate representation, measuring i in γ means $M_i^{\gamma} \rho = \text{tr}[P_i H_i Z_i(-\gamma) \rho]$, where P is measurement projector in the computational basis; thus, $M_i^{\gamma} N_i^{\gamma} = M_i^0 N_i^0$.

¹⁰ It can be considered as a measurement that is X -dependent.

beginning to join their computations; if $\tilde{I} = \emptyset$, string t is omitted. If we compare resources of BOQC and UBQC schemes, the BOQC requires an additional secure-key channel due to the presence of Oscar. There are many mature symmetric key exchange protocols, some of which are used on a daily basis of our work, such as DES, 3DES, AES, IDEA, RC4, and RC5 (classical) [162], and QKD (quantum assisted) [12]. Thus, the additional requirement of the BOQC is feasible to realize.

5.2 SECURITY OF THE BOQC

The goal of this section is to provide the statement “the BOQC protocol is unconditionally blind”. For that, we use the AC framework (reviewed in Section 3.2) to prove statements on correctness (Definition 3.2) and security (Definition 3.3). We promise the security of BOQC as *blindness*, *i.e.*, Bob cannot learn anything about the computation apart from public information that is provided by Alice and Oscar.

Denote the BOQC protocol in Protocol 5.1 as π_{boqc} and real-world resources used in the protocol as $\mathcal{R}$. In Protocol 5.1, resource $\mathcal{R}$ comprises a secret key channel, two-way quantum channels, and two-way (insecure) classical channels; the interactions set up via these channels can be observed from Figure 5.2. We define the ideal-world resource of the BOQC protocol $\mathcal{S}$ in Figure 5.3.

The ideal resource $\mathcal{S}$ has the same interface set as the BOQC, *i.e.*, $\mathcal{I} = \{\mathcal{A}, \mathcal{O}, \mathcal{B}\}$, where $\mathcal{A}$, $\mathcal{O}$, and $\mathcal{B}$ indicate Alice, Oscar, and Bob respectively. Also, notice that $\mathcal{S}$ possesses the same input output format as the BOQC: input $\{(\mathcal{G}, I, O), f, \phi, \rho_{\mathcal{A}}^{in}\}$ from Alice, input $\{\psi\}$ from Oscar, and output $\mathcal{E}(\rho_{\mathcal{A}}^{in})$ is sent to Alice. In the ideal model $\mathcal{S}$, we allow some leak of classical information $\ell^{\mathcal{AO}}$ on Bob’s interface, where

$$\ell^{\mathcal{AO}} := \{(\mathcal{G}, \tilde{I}, \tilde{O}), V_{\mathcal{A}}, V_{\mathcal{O}}, \succ, >, b\}. \quad (5.2)$$

We refer to Definition 5.1 for the notation.

Information $\ell^{\mathcal{AO}}$ is publicly available, and is necessary to run the protocol. Notice that $\ell^{\mathcal{AO}}$ is obtained from the pre-protocol procedure in Definition 5.1, where $\tilde{I} = I$ and $\tilde{O} = O$. The open graph $(\mathcal{G}, \tilde{I}, \tilde{O})$ is necessary for Bob to obtain the graph state and to prepare transmission channel, sets $V_{\mathcal{A}}$ and $V_{\mathcal{O}}$ are necessary for the transmission switching (whether connected to Alice or to Oscar). Orderings $\succ, >$ indicate the measuring order. The integer b — equivalently the set of angles Ω — is necessary for Bob to prepare measurement devices.

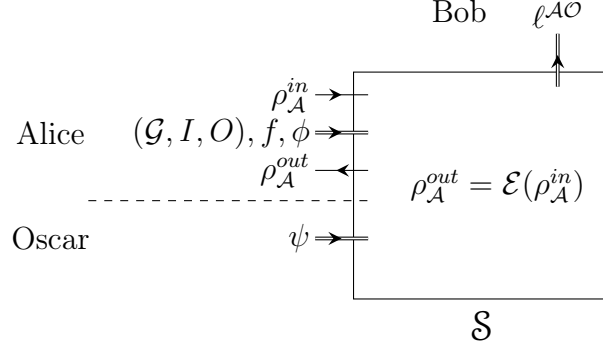


Figure 5.3: The ideal BOQC resource $\mathcal{S}$ without adversary. Alice and Oscar have an interface on the left side, whereas Bob's interface is on the top. Single-line arrows signify quantum information, and double-line arrows indicate classical information. Resource $\mathcal{S}$ receives inputs from Alice: an open graph $(\mathcal{G}, I, O)$ that has flow $(f, \succ)$ together with the measurement angles ϕ , and a quantum input ρ_A^{in} , where $\phi_i \in \Omega$ and $\Omega = \{\frac{k\pi}{2^{b-1}}\}_{0 \leq k < 2^b}$ for an integer b . Also, it receives inputs ψ from Oscar, i.e., the measurement angles for his nodes, where $\psi_i \in \Omega$. Resource $\mathcal{S}$ does not receive any honest input from Bob. Alice receives the final output state $\rho_B^{out} = \mathcal{E}(\rho_A^{in})$, where $\mathcal{E}$ is defined as the resulting superoperator with an isometry defined in [Theorem 2.1](#) in [Section 2.4](#). The resource leaks classical information $\ell^{AO} = \{(\mathcal{G}, \tilde{I}, \tilde{O}), V_A, V_O, \succ, >, b\}$ on Bob's interface, which are necessary to prepare and to run the protocol. The leak ℓ^{AO} corresponds to public information from [Definition 5.1](#).

Thus, to achieve perfect correctness ($\varepsilon = 0$), we must prove the statement

$$\mathcal{R} \xrightarrow{\pi_{boqc}, 0} \mathcal{S}, \quad (5.3)$$

where Bob is honest; this statement is formalized and proven in [Theorem 5.2](#) below.

Theorem 5.2. The BOQC protocol $\pi_{boqc} = (\pi_A, \pi_O, \pi_B)$ defined in [Protocol 5.1](#) is perfectly correct and emulates the ideal resource $\mathcal{S}$ defined in [Figure 5.3](#).

Proof. Protocol π_{boqc} is correct if it satisfies [Definition 3.2](#): $d(\pi_A \pi_O \mathcal{R} \pi_B, \mathcal{S}) = 0$, i.e., resources $\pi_A \pi_O \mathcal{R} \pi_B$ and $\mathcal{S}$ must be perfectly indistinguishable, where d is a pseudo-metric with properties defined in [Equation \(3.7\)](#). Which means, we show that the distinguishing advantage (defined in [Equation \(3.9\)](#)) is zero. For that, we show that both resources have the same — or statistically the same — inputs and outputs.

First, we show that π_{boqc} and $\mathcal{S}$ have an identical description of inputs. As shown in [Protocol 5.1](#), the protocol receives inputs $\{(\mathcal{G}, I, O), f, \phi, \rho_A^{in}\}$ from Alice, ψ from Oscar, and no honest input from Bob. These inputs are identical to the inputs of $\mathcal{S}$.

Second, $\mathcal{S}$ sends an output $\rho_A^{out} = \mathcal{E}(\rho_A^{in})$, where $\mathcal{E}$ is the superoperator with an isometry given in [Theorem 2.1](#); using [Theorem 5.1](#), BOQC also implements that isometry.

Finally we show that BOQC leaks the same information as $\mathcal{S}$, which is ℓ^{AO} . Bob receives information $\{(\mathcal{G}, \tilde{I}, \tilde{O}), V_A, V_O, \succ, >, b\}$ in the protocol, which is a public information obtained from the pre-protocol steps defined in [Definition 5.1](#) before the protocol starts. The public information is identical to leak ℓ^{AO} . Bob is not curious in this setting, thus there is no additional information obtained beyond ℓ^{AO} . However, here, he does not erase the received information, as it is required to run the protocol and to provide the bill for his clients.¹¹ $\square$

As [Theorem 5.2](#) proves that BOQC is correct, it remains to show what functionalities are realized in the presence of an adversary, namely when Bob is malicious and he performs arbitrary operations, deviating from his protocol. When Bob is dishonest, the problem becomes more interesting. In such a situation, Alice and Oscar must always send their (encrypted) information to Bob, wishing for Bob to perform their computation, even though he is untrustworthy.

In principle, we use the same technique as proving correctness to prove security: we determine the ideal resource while Bob is cheating, then determine how close it is to the real resource, as shown in [Definition 3.3](#). In this setting, we define the same real resource as before, *i.e.*, resource $\mathcal{R}$. Since Bob may deviate from [Protocol 5.1](#), we eliminate his converter (π_B). Now, we are trying to achieve the statement

$$\mathcal{R} \xrightarrow{\pi_{boqc}, 0} \mathcal{S}', \quad (5.4)$$

where the ideal resource $\mathcal{S}'$ defined in [Figure 5.4](#) provides blindness. As we wish to achieve an unconditional blindness, we set $\varepsilon = 0$.

Resource $\mathcal{S}'$ leaks the same information as resource $\mathcal{S}$, namely ℓ^{AO} . It means, we are constructing an ideal resource, such that Bob will not learn anything beyond the public information ℓ^{AO} even when he is cheating. Bob's malicious activity is captured by inputting dishonest information $\mathbb{B}$ from Bob, which allows him to deviate from his tasks. As Bob may not follow his protocol, we remove his

¹¹ Bob is not curious, but he needs to record some information for his clients to pay. For example, the dense Ω may cost more than the sparse Ω .

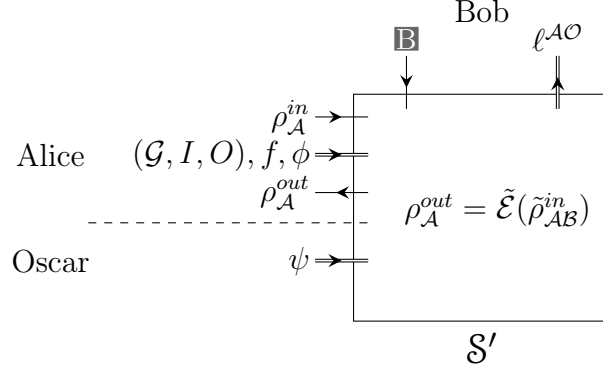


Figure 5.4: Ideal resource S' that provides unconditional blindness while Bob is malicious. On the clients' interface, the inputs and outputs are identical to [Protocol 5.1](#) (also to S in [Figure 5.3](#)). Resource S' does not take any honest input from Bob, but he may entangle Alice's input with a state of his choice ($\tilde{\rho}_{AB}^{in}$) and apply any superoperator as he wishes ($\tilde{\mathcal{E}}$), obtaining $\tilde{\mathcal{E}}(\tilde{\rho}_{AB}^{in})$. Bob's dishonest inputs are captured in $\mathbb{B}$. The output ρ_A^{out} depends on the inputs at $\mathbb{B}$; more specifically, these inputs are used to send the extension ρ_B^{in} to S' and to determine what map $\tilde{\mathcal{E}}$ is applied. See [Figure 5.3](#) for details on the rest of notations.

converter, and examine what he can do by finding a relaxation such that it is indistinguishable to the BOQC protocol with deviating Bob (see [Definition 3.3](#)), given that any relaxation is acceptable in the ideal world (see [Section 3.2](#)). Such a relaxation indeed exists, and is provided in [Protocol 5.2](#). We formally prove the blindness of BOQC in [Theorem 5.3](#).

Theorem 5.3. *The BOQC protocol with dishonest Bob $\pi_{boqc} = \{\pi_A, \pi_O\}$, defined in [Protocol 5.1](#) is perfectly blind and realizes the ideal resource S' defined in [Figure 5.4](#).*

Proof. Applying [Definition 3.3](#), $\mathcal{R} \xrightarrow{\pi_{boqc}, 0} S'$ if there exists a simulator σ_B such that $d(\pi_A \pi_O \mathcal{R}, S' \sigma_B) = 0$; thus, we must find a relaxation $S' \sigma_B$ that is perfectly indistinguishable from $\pi_A \pi_O \mathcal{R}$. Suppose the relaxation $S' \sigma_B$ is defined in [Protocol 5.2](#), then we proceed to prove that it is indistinguishable from $\pi_A \pi_O \mathcal{R}$.

To simplify the problem, we first reduce the protocol to a one-client protocol as follows. Consider a pictorial representation of π_{boqc} (in the absence of Bob) in [Figure 5.5](#); it clearly shows that the common information between Alice and Oscar is the keys s, t , shared via a secure key channel. Since it is known that a secure channel guarantees secrecy and authenticity, Alice and Oscar obtain their keys without leaking any information to Bob — apart from the keys' sizes. Therefore, we may think that Alice and Oscar have already shared the keys before the protocol

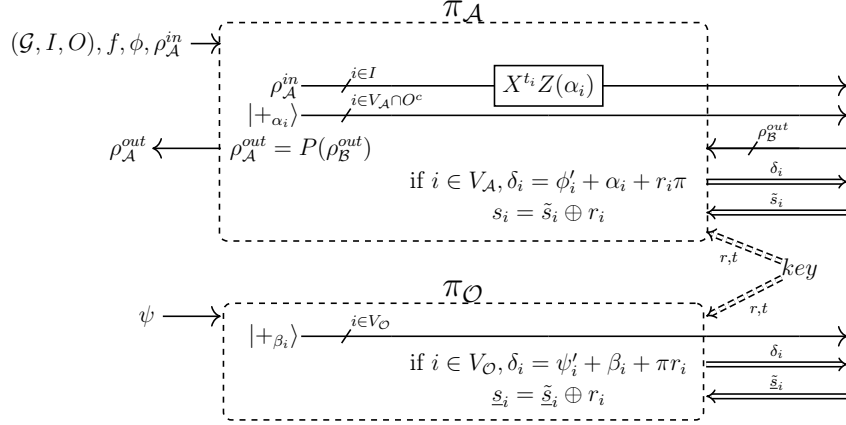


Figure 5.5: Pictorial representation of $\pi_A \pi_O \mathcal{R}$. Each variable corresponds to the one in [Protocol 5.1](#). We distinguish measurement outcomes on Alice’s interface ($\tilde{s}_i$) and on Oscar’s interface ($\underline{s}_i$), because Bob is allowed to be dishonest, thus $\tilde{s}_i$ and $\underline{s}_i$ may be different. The left-hand side denotes the outside interface of the converters. The right-hand side indicates the inside interface connected to resources: secure key channel (dashed double arrow), classical channel (single arrow), and quantum channel (double arrow). The strike-through line indicates multiple qubits.

starts. Thus, protocol $\pi_A \pi_O$ is reducible to a one-client protocol π_{AO} , shown in [Figure 5.6](#), where the alternating part between Alice and Oscar is captured within functions $\theta(i)$, $s(i)$, and $\delta(i)$.

Relaxation $\mathcal{S}'\sigma_B$ defined in [Protocol 5.2](#) is pictured in [Figure 5.7](#). Thus, we now can prove statement $d(\pi_A \pi_O \mathcal{R}, \mathcal{S}'\sigma_B) = 0$ by showing that [Figure 5.6](#) and [Figure 5.7](#) are indistinguishable.

Notice that [Figures 5.5](#) and [5.6](#) clearly show identical inputs and outputs, indicated by the same configuration of arrows. Thus, it now remains for us to prove that the arrows with a circled letter are (statistically) the same.

Remark that we focus only on blindness without verifiability. Since verification is not involved, Alice does not care whether her computation is correct or not. That is, some information related to computation can be arbitrary, such as ρ_B^{out} , $\tilde{s}_i$, $\underline{s}_i$, and δ_i .

Consider the information sent by Bob: **c** ρ_B^{out} , **e** $\tilde{s}_i$, and **f** $\underline{s}_i$. First, in both figures, ρ_B^{out} is an arbitrary state chosen by Bob (simulator). Second, in [Figure 5.6](#), $\tilde{s}_i$ and $\underline{s}_i$ signify measurement outcomes seen by Bob, which is random information independent of the actual measurement outcomes: $s_i = \tilde{s}_i \oplus r_i$ and $\underline{s}_i = \tilde{s}_i \oplus r_i$. The same is true in [Figure 5.7](#); $\tilde{s}_i, \underline{s}_i$ is arbitrary information inputted to Bob’s interface in $\mathcal{S}'$.

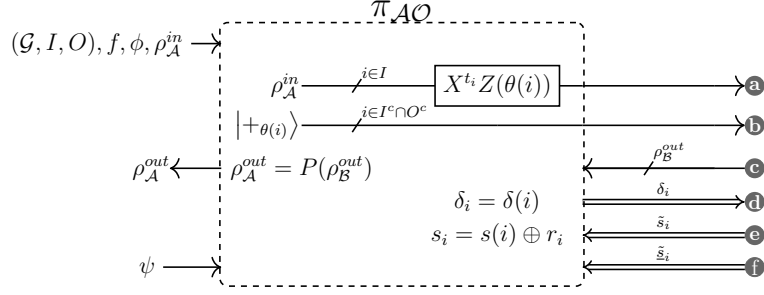


Figure 5.6: Pictorial representation of $\pi_{\mathcal{AO}}\mathcal{R}$. See Figure 5.5 for notation; additionally,

$$\theta(i) = \begin{cases} \alpha_i, & \text{if } i \in V_{\mathcal{A}} \\ \beta_i, & \text{if } i \in V_{\mathcal{O}}, \end{cases} \quad s(i) = \begin{cases} \tilde{s}_i, & \text{if } i \in V_{\mathcal{A}} \\ \tilde{s}_i, & \text{if } i \in V_{\mathcal{O}}, \end{cases} \quad \text{and } \delta(i) = \begin{cases} \phi'_i + \alpha_i + \pi r_i, & \text{if } i \in V_{\mathcal{A}} \\ \psi'_i + \beta_i + \pi r_i, & \text{if } i \in V_{\mathcal{O}}. \end{cases}$$

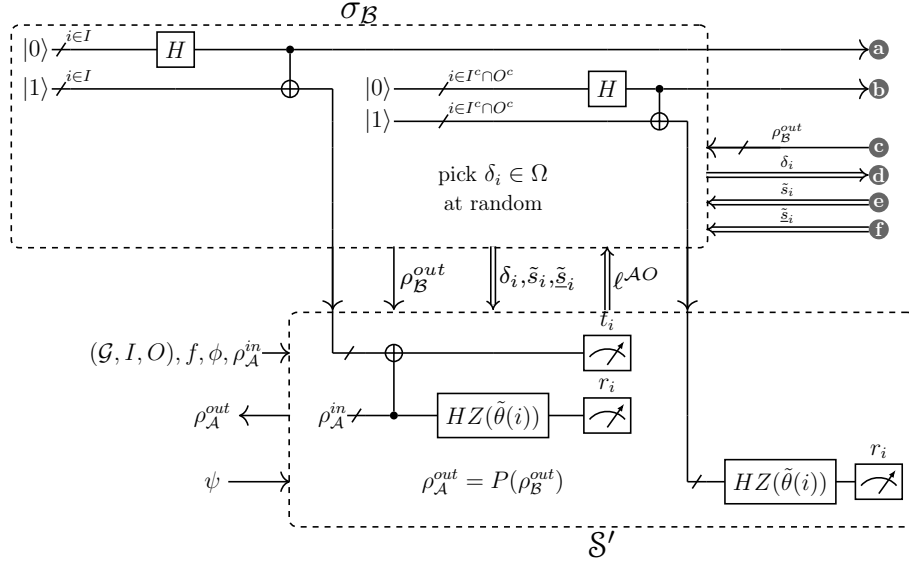


Figure 5.7: Pictorial representation of $\mathcal{S}'_{\mathcal{B}}$ (Protocol 5.2). The variables correspond to those defined in Protocol 5.2. See Figure 5.5 for the notation; additionally, $\tilde{\theta}(i) = \begin{cases} \delta_i - \phi'_i, & \text{if } i \in V_{\mathcal{A}} \\ \delta_i - \psi'_i, & \text{if } i \in V_{\mathcal{O}}. \end{cases}$

We now analyze the information received by Bob: **a**, **b**, and **d** δ_i . First, in both figures, δ_i are uniformly distributed random angles, independent of the actual measurement angles, ϕ'_i, ψ'_i . Second, consider the information at **a**, namely at input node $i \in I$. In Figure 5.6, **a** is an encrypted input state $X^{t_i}Z_i(\alpha_i)(\rho_i)$, where $\rho_i := \text{tr}_{I \setminus i}[\rho_{\mathcal{A}}^{\text{in}}]$. In Figure 5.7, **a** is an uncorrected teleported state $X_i^{t_i}Z_i^{r_i}Z_i(\delta_i - \phi'_i)(\rho_i) = X_i^{t_i}Z_i(\delta_i - \phi'_i - \pi r_i)(\rho_i) = X_i^{t_i}Z_i(\alpha_i)(\rho_i)$, which is identical to **a** in Figure 5.6. Third, for **b**, consider $i \in V_{\mathcal{A}}$. In Figure 5.6, **b** is $|+\alpha_i\rangle$. In Figure 5.7, **b** is a remote state preparation[163] of $Z^{r_i}|+\delta_i - \phi'_i\rangle = |+\delta_i - \phi'_i - \pi r_i\rangle = |+\alpha_i\rangle$. This is also clearly true for $i \in V_{\mathcal{O}}$.

Finally, since both figures have the same inputs and outputs, it remains to show that the leak is given by ℓ^{AO} . The leak of information ℓ^{AO} is inputted to the simulator from Bob's interface. The simulator does not use the leak to create any useful information. Now we need to prove that the simulator does not learn anything beyond ℓ^{AO} . For that, the information received by the simulator must be independent of the computation. First, δ_i is an arbitrary angle, thus independent of the computation. Second, **a** and **b** are completely mixed states because of the randomness α_i, β_i ; thus, the simulator cannot guess α_i or β_i with complete certainty. A curious Bob might correctly guess other information such as the flow f , however it gives no advantage to him. Therefore, there is no more leak than ℓ^{AO} throughout the protocol, which concludes the proof. $\square$

The construction of our simulator σ_B in [Protocol 5.2](#) comes to us from the security proof of the UBQC by Dunjko et al. in [\[113\]](#). However, its usefulness in our case is only guaranteed by [Theorem 5.1](#).

Here we consider only the general case when input and output are entirely quantum, *i.e.*, $\tilde{I} = I$ and $\tilde{O} = O$. Moreover, the security is maintained for the case $\tilde{I} \subset I$ and $\tilde{O} \subset O$, *i.e.*, input and output are partially or entirely classical; this is true if it admits the same security model, which includes $\mathcal{S}, \mathcal{S}'$, and π_{boqc} . Let the input be partially classical, we distinguish Alice's quantum input (ρ^{in}) and the quantum input in the security model (ρ_A^{in}). Thus, $\rho^{in} \in \mathcal{H}_{\tilde{I}}$ and $\rho_A^{in} \in \mathcal{H}_I$, where $\mathcal{H}_{\tilde{I}} \subset \mathcal{H}_I$. Suppose the classical input is a bit string c , then we encode c into the quantum state $\rho^c = \bigotimes_{i \in I \setminus \tilde{I}} |c_i\rangle\langle c_i|$. Thus, we can set the input state in the security model as $\rho_A^{in} = \rho^{in} \otimes \rho^c$. For the entirely classical input, $\tilde{I} = \emptyset$, input ρ_A^{in} is a diagonal matrix. If the output is classical, Bob measures the output nodes, and superoperator $\mathcal{E}$ can still incorporate classical outcomes. Therefore, the security is maintained for the partially or fully classical input and output.

We have shown that the BOQC scheme realizes $\mathcal{S}'$ and is perfectly blind in a composable definition, where Bob cannot learn anything about his clients' computation apart from the public information ℓ^{AO} . Moreover, since the security definition is composable, the BOQC protocol is still secure as a subsystem of a larger cryptographic protocol (see composition rules in [Section 3.2](#)). However, the clients are oblivious to Bob's malicious activities. Thus, verifiability is the next step towards a stronger security definition for a secure delegated oracular quantum computation.

Protocol 5.2 A relaxation $\mathcal{S}'\sigma_B$

Conventions:

- (I) Given public information $\{(\mathcal{G}, \tilde{I}, \tilde{O}), V_{\mathcal{A}}, V_{\mathcal{O}}, \succ, >, b\}$, where $\tilde{I} = I$, $\tilde{O} = O$, $\mathcal{G} = (V, E)$, $V = V_{\mathcal{A}} \cup V_{\mathcal{O}}$, and $\Omega = \{\frac{\pi k}{2^{b-1}}\}_{0 \leq k < 2^b}$.
- (II) $s_{\text{invf}(i)} = 0$, $\underline{s}_{\text{invf}(i)} = 0$ for all $i \in I$ and $t_i = 0$ for all $i \in I^c$.
- (III) $\text{invf}(i) \equiv f^{-1}(i)$, $z(i) := \bigoplus_{k < i, i \in N_{\mathcal{G}}(f(k))} s_k$, $\underline{z}(i) := \bigoplus_{k < i, i \in N_{\mathcal{G}}(f(k))} \underline{s}_k$, and $t(i) := \sum_{k \in I, i \in N_{\mathcal{G}}(k)} t_k$.
- (IV) Measurements are performed in the computational basis.

The simulator σ_B

- 1: Prepares an EPR pair $(|00\rangle + |11\rangle)/\sqrt{2}$ for every node $i \in O^c$ and outputs its half.
- 2: Picks random angles $\{\delta_i \in \Omega \mid i \in O^c\} =: \delta$ and outputs it.
- 3: Receives responses $\{\tilde{s}_i \in \{0, 1\} \mid i \in O^c \cap V_{\mathcal{A}}\} =: \tilde{s}$ and $\{\tilde{s}_i \in \{0, 1\} \mid i \in O^c \cap V_{\mathcal{O}}\} =: \underline{\tilde{s}}$.
- 4: Receives the corresponding output qubits ρ_B^{out} (all qubits $i \in O$).
- 5: Sends $\mathcal{S}'$ the other halves of EPR pairs $(\{\text{half-EPR-}i \mid i \in O^c\} =: \text{EPRs}), \delta, \tilde{s}, \underline{\tilde{s}}$, and ρ_B^{out} .

The ideal BOQC resource $\mathcal{S}'$

- 6: Receives $\{(\mathcal{G}, I, O), f, \rho_{\mathcal{A}}^{\text{in}}, \phi\}$ at Alice's interface and ψ at Oscar's interface, and information from step 5 — EPRs, $\delta, \tilde{s}, \underline{\tilde{s}}$, and ρ_B^{out} .
- 7: Applies CNOT gates between $\text{tr}_{I \setminus i}[\rho_{\mathcal{A}}^{\text{in}}]$ (as control) and half-EPR- i , for all $i \in I$, then measures the half-EPR, stores the outcome as t_i , and updates measurement angles:

$$\begin{aligned} \forall i \in I, \phi_i &= (-1)^{t_i} \phi_i, \\ \forall i \in I, \forall j \in N_{\mathcal{G}}(i) \cap V_{\mathcal{A}}, \phi_j &= \phi_j + t_i \pi \\ \forall i \in I, \forall j \in N_{\mathcal{G}}(i) \cap V_{\mathcal{O}}, \psi_j &= \psi_j + t_i \pi. \end{aligned}$$

- 8: **for** $i \in O^c$ which follows ordering $>$ **do**
- 9: **if** $i \in V_{\mathcal{A}}$ **then**
- 10: Computes $\phi'_i = (-1)^{s_{\text{invf}(i)}} \phi_i + z(i)\pi$.
- 11: Computes $\theta'_i = \delta_i - \phi'_i$.
- 12: **else if** $i \in V_{\mathcal{O}}$ **then**
- 13: Computes $\psi'_i = (-1)^{\underline{s}_{\text{invf}(i)}} \psi_i + \underline{z}(i)\pi$.
- 14: Computes $\theta'_i = \delta_i - \psi'_i$.
- 15: **end if**

```

16:   if  $i \in I$  then
17:       Applies  $HZ_i(\theta_i)$  to an input qubit  $\text{tr}_{I \setminus i}[\rho_{\mathcal{A}}^{\text{in}}]$  followed by measurement.
18:   else
19:       Applies  $HZ_i(\theta_i)$  to the half-EPR- $i$  followed by measurement
20:   end if
21:   Stores the measurement outcome as  $r_i$  and sets  $s_i = \tilde{s}_i \oplus r_i$  and  $\underline{s}_i = \tilde{\underline{s}}_i \oplus r_i$ .
22: end for
23: Corrects the output  $P(\rho_{\mathcal{B}}^{\text{out}}) =: \rho_{\mathcal{A}}^{\text{out}}$ , where  $P \equiv \bigotimes_{i \in O} X_i^{s_{\text{invf}(i)} + t_i} Z_i^{z(i) + t(i)}$ , then
    outputs it on Alice's interface.

```

5.2.1 The BOQC-compatible graphs

The BOQC protocol promises blindness, even with malicious activities of Bob, *i.e.*, Bob can only learn public information (the leak) $\ell^{AO} := \{(\mathcal{G}, \tilde{I}, \tilde{O}), V_{\mathcal{A}}, V_{\mathcal{O}}, \succ, \succ, b\}$. One can certainly not infer the computation from the leak, however, here is the catch.

Consider a real-life situation, where Oscar's oracle company is well-known as one that stores massive confidential databases. His clients, *e.g.*, Alice, uses the company's service to request some data. Thus, Bob knows his clients are running a quantum search algorithm. If Oscar provides a different graph for every state of this database, Bob might learn that state from the graph itself, *e.g.*, a graph that seems simple might perform single unitary gates. Hence, Oscar needs to blind his oracles by predetermining a standard graph for all database states, such that it only requires him to adjust the measurement angles according to the state of his database. Such standard oracle graphs we call *BOQC-compatible* graphs.

We propose two ways to construct BOQC-compatible graphs. First, using a universal graph state such as brickwork state [25] (Figure 3.4) with a fixed size. But since using a brickwork state can be expensive,¹² we propose graph optimization techniques constrained to all possible requests. We implemented such strategy in [143] to run 2- and 3-qubit exact quantum search algorithms, obtaining much more compact graph states than given by brickwork states. The strategy relies on numerical optimizations, which we elaborate in Section 6.3.

¹² For example, implementing a CNOT gate on a brickwork state needs 10 nodes (see Figure 3.5), whereas Equation (2.13) uses 4 nodes.

5.3 BOQC WITH CLASSICAL INPUT-OUTPUT AND QUANTUM POWER

One of the key features in the UBQC scheme (reviewed in [Section 3.1](#)) is the clear separation of the distributed quantum power between a client and a server: the client's quantum power is limited to generating a family of quantum states, whereas a server has unlimited quantum power. In the process of extending the UBQC scheme to the BOQC scheme, we maintain this clear separation; hence, Alice and Oscar possess the same quantum power as the client in the UBQC, and Bob possesses unlimited quantum power. In practice, Bob does not need unlimited quantum power, but this assumption is used for the security analysis. The quantum power distribution of the BOQC scheme can be easily recognized in our illustration in [Figure 5.1](#).

Here, we use the notation from [Definition 5.1](#). If we watch every step in the BOQC protocol ([Protocol 5.1](#)), estimating the quantum power, the requirement is smaller when $\tilde{I} = \tilde{O} = \emptyset$, *i.e.*, input and output are entirely classical. For instance, if $\tilde{O} \neq \emptyset$, Alice needs a device that can receive some qubits and perform Pauli corrections on them; such a device would be unnecessary if $\tilde{O} = \emptyset$. The BOQC protocol in [Protocol 5.1](#) provides a general scheme where $\tilde{I} = I$ and $\tilde{O} = O$; it has the highest demand compared with other cases, *e.g.*, $\tilde{O} = \emptyset$. The increased requirement of quantum power applies to resources¹³ as well, *e.g.*, a two-way quantum channel between Alice and Bob is needed to transmit final outputs. We tabulate the requirements of quantum power and resources in [Table 5.1](#) for every combination of input-output types.

A few changes in [Protocol 5.1](#) are needed if $\tilde{I} \subset I$ or $\tilde{O} \subset O$. For instance [Protocol 5.3](#) provides the BOQC protocol when input and output are entirely classical, *i.e.*, $\tilde{I} = \tilde{O} = \emptyset$. If $\tilde{I} = \emptyset$, the QOTP¹⁴ encryption on input qubits is omitted, Alice's input preparation is done as the following instead. Let her input be a binary string $c = c_1 c_2 \dots c_n$, encoded as $\bigotimes_{i \in I} Z^{c_i} |+\rangle_i$; now, her input states are in $|\pm_\alpha\rangle$ basis. Since we omit the QOTP step, lines 6 – 8 in [Protocol 5.1](#) are replaced by a single line

$$\text{“Alice prepares } |+\alpha_i + \pi c_i\rangle\text{”};$$

recall that $Z |+\rangle = |+\pi\rangle$. Since QOTP is omitted, random variable t is unnecessary, or equivalently setting $t_i = 0, \forall i$; this eliminates requirement **(C3)** in [Table 5.1](#). On the server's side, the requirement to receive arbitrary quantum states becomes optional, which eliminates **(S3)**.

¹³ The notion resource here refers to the real-world resource in abstract cryptography ([Section 3.2](#)).

¹⁴ QOTP is discussed in the extra box in [Section 3.1](#)

I	O	Client	Server	Resource
c	c	(C1) creates $ +\theta\rangle$, then transmits it to Bob	(S1) receives $ +\theta\rangle$, performs CPHASE gates, and measures qubits in the xy -plane bases	$\mathcal{K}, \mathcal{C}, \mathcal{Q}$
c	q	(C1), (C2) receives ρ_B^{out} from Bob, then performs Pauli correction on it	(S1), (S2) create $ +\rangle \forall i \in O$ and sends Alice the final outputs ρ_B^{out}	$\mathcal{K}, \mathcal{C}, \mathcal{Q}, \mathcal{Q}2$
q	c	(C1), (C3) creates quantum input ρ_A^{in} , performs QOTP (see the extra box in Section 3.1), then transmits it to Bob	(S1), (S3) receives input states (arbitrary) from Alice	$\mathcal{K}, \mathcal{C}, \mathcal{Q}$
q	q	(C1), (C2), (C3)	(S1), (S2), (S3)	$\mathcal{K}, \mathcal{C}, \mathcal{Q}, \mathcal{Q}2$

Table 5.1: Minimal quantum power requirements of client and server to run the BOQC protocol (also the UBQC). Initial “c” indicates “entirely classical” and initial “q” indicates “entirely quantum”. Resources $\mathcal{K}, \mathcal{C}, \mathcal{Q}$, and $\mathcal{Q}2$ are respectively signifying a secure key channel, an insecure classical channel, a one-way quantum channel (client to server), and a two-way quantum channel. Other notations follow the notations in [Protocol 5.1](#).

Bob will measure all qubits for the classical output case, and the following changes need to be made for the BOQC protocol (general case). First, Bob does not need to prepare the output states; we remove line 16 in [Protocol 5.1](#), which eliminates requirement (S2). Hence, the loop of step ③ in line 18 is replaced by

“for $i \in V$ which follows partial ordering $\prec$ ”,

which dismisses the whole step ④, eliminating requirement (C2).

The lowest quantum power requirement applies when input and output are classical, where the client requires only (C1), and the server requires only (S1). The BOQC protocol with the lowest requirement of quantum power is shown in [Protocol 5.3](#).

Several qubit implementations on physical quantum systems have been studied today, where each of them uniquely possesses features and limitations. Categorizing quantum power requirements can help us classify a class of solvable problems by a particular quantum system.

Protocol 5.3 BOQC with classical input-output

Alice's input: $\{(\mathcal{G}, I, O), f, \phi, c = c_1 c_2, \dots, c_n\} \triangleright \tilde{I} = \tilde{O} = \emptyset, \rho_{\mathcal{A}}^{in} = \prod_{i=1}^n |c_i\rangle\langle c_i|$

Oscar's input: $\{\psi\}$

Alice's output for an honest Bob: $\rho_{\mathcal{A}}^{out} = \mathcal{E}(\rho_{\mathcal{A}}^{in}) \triangleright \rho_{\mathcal{A}}^{out}$ is a diagonal matrix

Assumptions and conventions:

(I) Alice ($\mathcal{A}$) and Oscar ($\mathcal{O}$) have performed pre-protocol steps in [Definition 5.1](#); Bob knows $\{(\mathcal{G}, \tilde{I}, \tilde{O}), V_{\mathcal{A}}, V_{\mathcal{O}}, \succ, >, b\}$. Here, we set $\tilde{I} = \tilde{O} = \emptyset$, and recall $\Omega = \{\frac{\pi k}{2^{b-1}}\}_{0 \leq k < 2^b}$.

(II) $s_{invf(i)} = 0, \forall i \in I$.

(III) $invf(i) \equiv f^{-1}(i)$ and $z(i) := \bigoplus_{k \prec i, i \in N_{\mathcal{G}}(f(k))} s_k$.

0 Pre-preparation

1: Alice and Oscar receive a key r via a secure key channel, where $r_i \in \{0, 1\}$, for $i \in O^c$.

1 State preparation

2: **for** $i \in V$ which follows partial ordering $\succ$ **do**

3: **if** $i \in V_{\mathcal{A}}$ **then**

4: **if** $i \in I$ **then**

5: Alice prepares $|+\alpha_i + c_i \pi\rangle_i$ and sends it to Bob; $\alpha_i \in \Omega$ is chosen at random.

6: **else**

7: Alice prepares $|+\alpha_i\rangle_i$ and sends it to Bob, where $\alpha_i \in \Omega$ is chosen at random

8: **end if**

9: **else if** $i \in V_{\mathcal{O}}$ **then**

10: Oscar prepares $|+\beta_i\rangle_i$ and sends it to Bob; $\beta_i \in \Omega$ is chosen at random.

11: **end if**

12: **end for**

2 Graph state formation

13: Bob applies entangling operator $E_{\mathcal{G}}$ defined in [Equation \(2.47\)](#).

3 Classical interaction and measurement

14: **for** $i \in V$ which follows partial ordering $\succ$ **do** $\triangleright$ e.g., it follows $>$

15: **if** $i \in V_{\mathcal{A}}$ **then**

16: Alice computes $\phi'_i = (-1)^{s_{invf(i)}} \phi_i + z(i)\pi$.

17: Alice computes $\delta_i := \phi'_i + \pi r_i + \alpha_i$, and sends Bob δ_i .

18: **else if** $i \in V_{\mathcal{O}}$ **then**

19: Oscar computes $\psi'_i = (-1)^{s_{invf(i)}} \psi_i + z(i)\pi$.

20: Oscar computes $\delta_i := \psi'_i + \pi r_i + \beta_i$, and sends Bob δ_i .

21: **end if**

22: Bob measures qubit i in basis $|\pm_{\delta_i}\rangle$, then sends Alice and Oscar the outcome $\tilde{s}_i$.

23: Alice and Oscar set $s_i = \tilde{s}_i \oplus r_i$.

24: **end for**

5.4 BOQCO IS BOQC ON SOLID-STATE QUBITS

The BOQC scheme is our contribution for a realization towards a physically meaningful quantum oracle in a client-server cryptographic setting, which is the setting that is believed to be prevailing in future. We reckon solid-state qubits as a strong candidate for the BOQC implementation, due to their progress in individual qubit control, even though they are only available in few numbers of high fidelity qubits. We thus introduce the lazy 1WQC (Section 2.5) to reduce the number of physical qubits, which works for qubits with quantum memories, *i.e.*, qubits that possess permanence and can be rapidly re-initialized after being measured. The lazy 1WQC has successfully bounded the required number of physical qubits to the number of output nodes plus one (Conjecture 2.1). Therefore, we now introduce a protocol that combines the BOQC with the lazy 1WQC, that we call *BOQC-optimized* (BOQCo). We hope our BOQCo protocol will inspire some solid-state qubit experimentalists to conduct exciting experiments, performing oracular quantum algorithms within a setting in which the oracles have a physical meaning.

We define the BOQCo protocol (π_{boqco}) in Protocol 5.4. The protocol implements the same strategy as the BOQC in achieving blindness, which is captured by the presence of random variables r, t, α , and β . The measurement commands (qubit transmissions, entanglements, and measurements) are distributed according to the lazy 1WQC.

Protocol 5.4 BOQCo ($\pi_{boqco} = \{\pi_{\mathcal{A}}, \pi_{\mathcal{B}}, \pi_{\mathcal{O}}\}$)

Alice's input: $\{(\mathcal{G}, I, O), f, \phi, \rho_A^{in}\}$ $\triangleright \tilde{I} = I$ and $\tilde{O} = O$

Oscar's input: $\{\psi\}$

Alice's output for an honest Bob: $\rho_A^{out} = \mathcal{E}(\rho_A^{in})$

Assumptions and conventions:

- (I) Alice ($\mathcal{A}$) and Oscar ($\mathcal{O}$) have performed pre-protocol steps in Definition 5.1; Bob knows $\{(\mathcal{G}, \tilde{I}, \tilde{O}), V_{\mathcal{A}}, V_{\mathcal{O}}, \succ, >, b\}$. Here, we set $\tilde{I} = I$ and $\tilde{O} = O$. Recall $\tilde{O} \cap V_{\mathcal{O}} = \emptyset$ (quantum outputs are held by Alice) and $\Omega = \{\frac{\pi k}{2^{b-1}}\}_{0 \leq k < 2^b}$.
- (II) $s_{invf(i)} = 0, \forall i \in I$ and $t_i = 0, \forall i \in I^c$.
- (III) $invf(i) \equiv f^{-1}(i)$, $z(i) := \bigoplus_{k \prec i, i \in N_{\mathcal{G}}(f(k))} s_k$, and $t(i) := \bigoplus_{k \in I, i \in N_{\mathcal{G}}(k)} t_k$.

0 Pre-preparation

- 1: Alice and Oscar receive keys r, t via a secure key channel, where $r_i \in \{0, 1\}$, $i \in O^c$ and $t_j \in \{0, 1\}$, $j \in I$.
-

1 BOQC by parts

```

2: for  $i \in V \setminus O$  with ordering  $>$  do
3:   for  $k \in A(i) \cup I$  do ▷ Equation (2.48), Section 2.1
4:     if  $k \in I$  then ▷ input qubit
5:       Alice applies  $Z_k(\alpha_k)X_k^{t_k}$  to  $\text{tr}_{I \setminus k}[\rho_{\mathcal{A}}^{in}]$  and sends it to Bob,  $\alpha_k \in \Omega$  is
       chosen at random.
6:       Alice updates angles:
           
$$\phi_k = (-1)^{t_k} \phi_k$$

           
$$\phi_j = \phi_j + t_k \pi, \forall j \in N_{\mathcal{G}}(k) \cap V_{\mathcal{A}}.$$

7:       Oscar updates angles:  $\psi_j = \psi_j + t_k \pi, \forall j \in N_{\mathcal{G}}(k) \cap V_{\mathcal{O}}.$ 
8:     else if  $k \in O$  then ▷ output qubit
9:       Bob prepares  $|+\rangle_k$ .
10:    else ▷ auxiliary qubit
11:      if  $k \in V_{\mathcal{A}}$  then
12:        Alice prepares  $|+\alpha_k\rangle_k$ , sends it to Bob,  $\alpha_k \in \Omega$  is chosen at
        random.
13:      else if  $k \in V_{\mathcal{O}}$  then
14:        Oscar prepares  $|+\beta_k\rangle_k$  and send it to Bob,  $\beta_k \in \Omega$  is chosen at
        random.
15:      end if
16:    end if
17:  end for
18:  Bob applies entangling operations  $E_{iN_{\mathcal{G}}(i)}^>$ . ▷ Equation (2.47), Section 2.1
19:  if  $i \in O^c$  then
20:    if  $i \in V_{\mathcal{A}}$  then
21:      Alice computes  $\phi'_i = (-1)^{s_{\text{invf}}(i)} \phi_i + z(i)\pi$ .
22:      Alice computes  $\delta_i := \phi'_i + \pi r_i + \alpha_i$  and sends Bob  $\delta_i$ .
23:    else if  $i \in V_{\mathcal{O}}$  then
24:      Oscar computes  $\psi'_i = (-1)^{s_{\text{invf}}(i)} \psi_i + z(i)\pi$ .
25:      Oscar computes  $\delta_i := \psi'_i + \pi r_i + \beta_i$  and sends Bob  $\delta_i$ .
26:    end if
27:    Bob measures  $i$  in  $|\pm_{\delta_i}\rangle$  basis, sends Alice and Oscar the outcome  $\tilde{s}_i$ .
28:    Alice and Oscar set  $s_i = \tilde{s}_i \oplus r_i$ .
29:  else ▷ output qubit transmissions and corrections
30:    Bob sends Alice qubit  $i$ .
31:    Alice corrects qubit  $i$  by applying  $X_i^{s_{\text{invf}}(i)+t_i} Z_i^{z(i)+t(i)}$ .
32:  end if
33: end for

```

The BOQCo comprises two steps: ❶ pre-preparation identical to the BOQC and ❷ BOQC by parts. The protocol has the same configuration of inputs and outputs as the BOQC, *i.e.*, it receives input from Alice as $\{(\mathcal{G}, I, O), f, \phi, \rho_A^{in}\}$, input from Oscar as $\{\psi\}$, and sends the final output ρ_B^{out} to Alice. Essentially, BOQCo is BOQC within the lazy 1WQC scheme. We introduce the same definitions of correctness and security of BOQCo as BOQC: the BOQCo protocol is correct if the resulting map is identical to the 1WQC scheme for the same inputs, and the BOQCo is unconditionally blind within a composable definition; these are discussed in the next subsection. Before that, let us first consider [Theorem 5.4](#) as follows.

Theorem 5.4. *The BOQCo protocol π_{boqco} defined in [Protocol 5.4](#) delegates a computation with the isometry defined in [Theorem 2.1](#) ([Section 2.4](#)) for the same computation, without requiring Alice and Oscar to communicate their computation to each other.*

Proof. We use the same reasoning as in the proof of [Theorem 5.1](#) by showing that the resulting pattern from [Protocol 5.4](#) can be reduced to the pattern in [Theorem 2.1](#). Note that, in this context, converters π_A, π_O , and π_B correspond to procedures of the BOQCo protocol defined in [Protocol 5.4](#); real resource $\mathcal{R}$ comprises the same elements as the BOQC *viz.*, a secure key, two-way classical channels, and two-way quantum channels.

Consider Alice and Oscar as one party, which we call super-Alice, who has all information about the angles ϕ, ψ and the random variables α, β, r, t . We denote $\gamma := \alpha \cup \beta$ and $\theta := \phi \cup \psi$.

First, we omit randomness, so that $r_i = t_i = \alpha_i = \beta_i = 0$ for all i ; the resulting BOQCo pattern ($\mathfrak{P}_{boqco}$) can be written as

$$\mathfrak{P}_{boqco} := \bigotimes_{j \in O} X_j^{s_{invf(j)}} Z_j^{z(j)} E_O \mathop{\mathbb{H}}_{i \in O^c}^> M_i^{\theta_i} X_i^{s_{invf(i)}} Z_i^{z(i)} E_{iN_G(i)}^> N_{A(i)}^0 = \mathfrak{P}_{lazy} \stackrel{\text{Theo. 2.4}}{=} \mathfrak{P}_{1wqc},$$

where $z(i) := \bigoplus_{k < i, i \in N_G(f(k))} s_k$, $invf(i) \equiv f^{-1}(i)$, $\mathfrak{P}_{lazy}$ is the resulting pattern of lazy 1WQC ([Equation \(2.50\)](#)), and $\mathfrak{P}_{1wqc}$ is the resulting pattern of 1WQC ([Equation \(2.43\)](#)), where

$$\mathfrak{P}_{1wqc} := \mathop{\mathbb{H}}_{i \in O^c}^> (X_{f(i)}^{s_i} \prod_{k \in N_G(f(i)) \setminus \{i\}} Z_i^{s_i} M_i^{\theta_i}) E_G N_{I^c}^0.$$

Thus, $\mathfrak{P}_{boqco} = \mathfrak{P}_{1wqc}$ holds in the absence of randomness. Note that total ordering $>$ is consistent with partial ordering $\succ$ (partial ordering induced by the flow); thus

we can interchangeably use both. Recall that operator $\otimes$ allows for concurrency, whereas $\prod$, $\mathbb{H}$, and $\mathbb{H}$ indicates serial operations.

Second, we introduce random variables γ and r into the protocol; now, pattern $\mathfrak{P}_{boqco}$ becomes

$$\mathfrak{P}_{boqco} = \mathbb{H}_{i \in O^c} (X_{f(i)}^{s_i} \prod_{k \in N_G(f(i)) \setminus \{i\}} Z_k^{s_i + r_i} M_i^{\theta_i + \gamma_i + \pi r_i}) E_G N_{I^c}^\gamma, \quad (5.5)$$

which is identical to Equation (5.1). Thus, from this point, the proof proceeds identically to that of Theorem 5.1 from Equation (5.1). $\square$

From Theorem 5.4, we know that BOQCo and BOQC result in identical computations. However, in the implementation, BOQCo requires many fewer physical qubits on Bob's side due to the lazy 1WQC paradigm. Also, the requirements of quantum power and resource are identical to the BOQC, which is discussed in Section 5.3. The BOQCo protocol when input and output are entirely classical is shown in Protocol 5.5, which has the lowest demand for quantum power.

Protocol 5.5 BOQCo with classical input-output

Alice's input: $\{(\mathcal{G}, I, O), f, \phi, c = c_1 c_2, \dots, c_n\} \triangleright \tilde{I} = \tilde{O} = \emptyset, \rho_{\mathcal{A}}^{in} = \prod_{i=1}^n |c_i\rangle\langle c_i|$

Oscar's input: $\{\psi\}$

Alice's output for an honest Bob: $\rho_{\mathcal{A}}^{out} = \mathcal{E}(\rho_{\mathcal{A}}^{in}) \triangleright \rho_{\mathcal{A}}^{out}$ is a diagonal matrix

Assumptions and conventions:

(I) Alice ($\mathcal{A}$) and Oscar ($\mathcal{O}$) have performed pre-protocol steps in Definition 5.1; Bob knows $\{(\mathcal{G}, \tilde{I}, \tilde{O}), V_{\mathcal{A}}, V_{\mathcal{O}}, \succ, >, b\}$. Here, we set $\tilde{I} = \tilde{O} = \emptyset$, and recall $\Omega = \{\frac{\pi k}{2^{b-1}}\}_{0 \leq k < 2^b}$.

(II) $s_{inv f(i)} = 0, \forall i \in I$.

(III) $inv f(i) \equiv f^{-1}(i)$ and $z(i) := \bigoplus_{k \prec i, i \in N_G(f(k))} s_k$.

0 Pre-preparation

- 1: Alice and Oscar receive a key r via a secure key channel, where $r_i \in \{0, 1\}$, for $i \in O^c$.

1 BOQC by parts

```

2: for  $i \in V \setminus O$  with ordering  $>$  do
3:   for  $k \in A(i) \cup I$  do ▷ see Equation (2.48) in Section 2.1
4:     if  $k \in I$  then ▷ input qubit
5:       Alice prepares  $|+\alpha_i + c_i \pi\rangle_i$  and sends it to Bob;  $\alpha_i \in \Omega$  is chosen at
       random.
6:     else ▷ auxiliary qubit
7:       if  $k \in V_A$  then
8:         Alice prepares  $|+\alpha_k\rangle_k$  and sends it to Bob,  $\alpha_k \in \Omega$  is chosen at
         random.
9:       else if  $k \in V_O$  then
10:        Oscar prepares  $|+\beta_k\rangle_k$  and send it to Bob,  $\beta_k \in \Omega$  is chosen at
        random.
11:      end if
12:    end if
13:  end for
14:  Bob applies entangling operation  $E_{iN_G(i)}^>$ . ▷ See Equation (2.47) in
  Section 2.1
15:  if  $i \in V_A$  then
16:    Alice computes  $\phi'_i = (-1)^{s_{invf(i)}} \phi_i + z(i)\pi$ .
17:    Alice computes  $\delta_i := \phi'_i + \pi r_i + \alpha_i$  and sends Bob  $\delta_i$ .
18:  else if  $i \in V_O$  then
19:    Oscar computes  $\psi'_i = (-1)^{s_{invf(i)}} \psi_i + z(i)\pi$ .
20:    Oscar computes  $\delta_i := \psi'_i + \pi r_i + \beta_i$  and sends Bob  $\delta_i$ .
21:  end if
22:  Bob measures  $i$  in  $|\pm_{\delta_i}\rangle$  basis, then sends Alice and Oscar the outcome  $\tilde{s}_i$ .
23:  Alice and Oscar set  $s_i = \tilde{s}_i \oplus r_i$ .
24: end for

```

5.4.1 Security of the BOQCo

Since BOQCo is BOQC within the lazy 1WQC paradigm, where the main difference is the order of computation process (transmission, entanglement, and measurement); we expect BOQCo to inherit security properties from the BOQC. Here, we prove that BOQCo is indeed as secure as BOQC.

We define ideal and real resources of BOQCo to be identical with the BOQC: ideal resource with an honest Bob $\mathcal{S}$ is defined in Figure 5.3, the ideal resource with a dishonest Bob $\mathcal{S}'$ is defined in Figure 5.4, and real resource $\mathcal{R}$ for both cases (honest and dishonest Bob) comprises a secure key channel between Alice and Oscar, two-way classical insecure channels between each client and Bob, a one-way

quantum channel between Oscar and Bob, and a two-way quantum channel between Alice and Bob. Note that, if the final output is classical, one-way quantum channel is sufficient for Alice.

We construct the same reasoning as when we proved the BOQC security, now to prove the BOQCo security. The correctness is achieved by obtaining $\mathcal{R} \xrightarrow{\pi_{boqco}, 0} \mathcal{S}$, where $\pi_{boqco} = \{\pi_{\mathcal{A}}, \pi_{\mathcal{O}}, \pi_{\mathcal{B}}\}$ (honest Bob). The perfect security (unconditionally blind, $\varepsilon = 0$) is achieved by obtaining $\mathcal{R} \xrightarrow{\pi_{boqco}, 0} \mathcal{S}'$, where $\pi_{boqco} = \{\pi_{\mathcal{A}}, \pi_{\mathcal{O}}\}$ (malicious Bob). The correctness is formalized in [Theorem 5.5](#) and the blindness in formalized in [Theorem 5.6](#).

Theorem 5.5. *The BOQCo protocol $\pi_{boqco} = (\pi_{\mathcal{A}}, \pi_{\mathcal{O}}, \pi_{\mathcal{B}})$, defined in [Protocol 5.4](#) is perfectly correct, and emulates ideal resource $\mathcal{S}$ defined in [Figure 5.3](#).*

Proof. Using [Definition 3.2](#), the correctness is achieved when $d(\pi_{\mathcal{A}}\pi_{\mathcal{O}}\mathcal{R}\pi_{\mathcal{B}}, \mathcal{S}) = 0$. We prove this condition by reducing BOQCo to BOQC.

First, [Protocol 5.4](#) (BOQCo) has the same inputs as [Protocol 5.1](#) (BOQC). Applying [Theorem 5.4](#), BOQCo also results in the same computation as BOQC, thus, the same output. Secondly, BOQCo and BOQC differ only in the ordering among transmissions, entanglements, and measurements; thus, there is no additional leak introduced beyond ℓ^{AO} (the leak in the BOQC, [Equation \(5.2\)](#)). In terms of correctness, BOQCo is reducible to BOQC. Finally, since BOQC is perfectly correct within the composable definitions, BOQCo is also perfectly correct within the composable definition. $\square$

At this point, the blindness of the BOQCo can be straightforwardly derived from the BOQC security statement; this is because the two protocols differ only in the measurement ordering commands, *viz.*, qubit transmissions, entanglements, and measurements.

Theorem 5.6. *The BOQCo protocol with dishonest Bob $\pi'_{boqco} = (\pi_{\mathcal{A}}, \pi_{\mathcal{O}})$, defined in [Protocol 5.4](#) is perfectly blind, and realizes ideal resource $\mathcal{S}'$ defined in [Figure 5.4](#).*

Proof. By [Definition 3.3](#), statement $\mathcal{R} \xrightarrow{\pi_{boqco}, 0} \mathcal{S}'$ is achieved if there exists a simulator $\sigma_{\mathcal{B}}$ such that $d(\pi_{\mathcal{A}}\pi_{\mathcal{O}}\mathcal{R}, \mathcal{S}'\sigma_{\mathcal{B}}) = 0$. Such a relaxation can be straightforwardly derived from [Protocol 5.2](#) by rearranging the process order, *viz.*, qubit transmissions, entanglements, and measurements according to the lazy computation in [Algorithm 2.2](#) (the relaxation is shown explicitly in [Protocol 5.6](#)).

Since BOQCo and BOQC differ only by the process order, they have the same configuration of inputs and outputs. Therefore, $\pi_{\mathcal{A}}\pi_{\mathcal{O}}\mathcal{R}$ and $\mathcal{S}'\sigma_{\mathcal{B}}$ have pictorial representation as [Figure 5.6](#) and [Figure 5.7](#), respectively, where the figures are proven to be indistinguishable in [Theorem 5.3](#). The difference in the process order will not reveal any information about the computation because the ordering is determined before the protocol starts, arranged in pre-protocol steps ([Definition 5.1](#)).

Since we obtain an indistinguishable relaxation to $\pi_{\mathcal{A}}\pi_{\mathcal{O}}\mathcal{R}$ that has the same leak as BOQC, this concludes the proof. $\square$

Analogous to the BOQC cases, the same security is also achieved for the BOQCo with classical input and output ($\tilde{I} \subset I$ and $\tilde{O} \subset O$), *e.g.*, for [Protocol 5.5](#) that has entirely classical input and output.

We have formalized how the BOQC and BOQCo protocols allow Alice to delegate her oracular quantum algorithm to Bob while Oscar provides her the oracles. The protocols are proven to be blind, *i.e.*, Bob can not learn about the computation apart from the allowed leaking information. We also tabulated the resource and the quantum power requirement for classical and quantum input-output, in which the requirement increases when the input and output are entirely quantum.

Protocol 5.6 A relaxation ($\mathcal{S}'\sigma_{\mathcal{B}}$) for BOQCo

Conventions:

- (I) Given public information $\{(\mathcal{G}, \tilde{I}, \tilde{O}), V_{\mathcal{A}}, V_{\mathcal{O}}, \succ, >, b\}$, where $\tilde{I} = I$, $\tilde{O} = O$, $\mathcal{G} = (V, E)$, $V = V_{\mathcal{A}} \cup V_{\mathcal{O}}$, and $\Omega = \{\frac{\pi k}{2^{b-1}}\}_{0 \leq k < 2^b}$.
- (II) $s_{\text{invf}(i)} = 0$, $\underline{s}_{\text{invf}(i)} = 0$ for all $i \in I$ and $t_i = 0$ for all $i \in I^c$.
- (III) $\text{invf}(i) \equiv f^{-1}(i)$, $z(i) := \bigoplus_{k < i, i \in N_{\mathcal{G}}(f(k))} s_k$, $\underline{z}(i) := \bigoplus_{k < i, i \in N_{\mathcal{G}}(f(k))} \underline{s}_k$, and $t(i) := \sum_{k \in I, i \in N_{\mathcal{G}}(k)} t_k$.
- (IV) Measurements are performed in the computational basis.

Part ①

- 1: $\mathcal{S}'$ receives $\{(\mathcal{G}, I, O), f, \rho_{\mathcal{A}}^{\text{in}}, \phi\}$ at Alice's interface and ψ at Oscar's interface.

Part ②

- 2: **for** $i \in V \setminus O$ with ordering $>$ **do**
 - 3: **for** $k \in A(i) \cup I$ **do** $\triangleright$ see [Equation \(2.48\)](#) from [Section 2.5](#)
-

The simulator σ_B

```

4:      if  $k \notin O$  then
5:          Prepares an EPR pair  $(|00\rangle + |11\rangle)/\sqrt{2}$  and outputs its half.
6:          Picks  $\delta_k \in \Omega$  at random, outputs  $\delta_k$ .
7:          Receives a responses  $\tilde{s}_k$  and  $\tilde{\underline{s}}_k$ , where  $\tilde{s}_k, \tilde{\underline{s}}_k \in \{0, 1\}$ .
8:          Sends resource  $S'$  the other half of EPR pair (half-EPR- $k$ ),  $\delta_k$ ,  $\tilde{s}_k$ , and
           $\tilde{\underline{s}}_k$ .
9:      else if  $k \in O$  then
10:          Receives output qubit  $\text{tr}_{O \setminus k}[\rho_B^{out}]$ , and sends it to the ideal resource.
11:      end if

```

The ideal BOQCo resource S'

```

12:      Receives  $\{\text{half-EPR-}k, s_k, \tilde{s}_k, \tilde{\underline{s}}_k, \delta_k\}$  if  $k \in O^c$ , otherwise receives
           $\text{tr}_{O \setminus k}[\rho_B^{out}]$ .
13:      if  $k \in I$  then
14:          Applies CNOT gate between  $\text{tr}_{I \setminus k}[\rho_A^{in}]$  and half-EPR- $k$ .
15:          Measures half-EPR- $k$ , stores the outcome as  $r_k$ , then updates:

```

$$\begin{aligned} \phi_k &= (-1)^{t_k} \phi_k, \\ \forall j \in N_G(k) \cap V_A, \phi_j &= \phi_j + t_k \pi \\ \forall j \in N_G(k) \cap V_O, \psi_j &= \psi_j + t_k \pi. \end{aligned}$$

```

16:      end if
17:      if  $k \in O^c$  then
18:          if  $k \in V_A$  then
19:              Computes  $\phi'_k = (-1)^{s_{inv}(k)} \phi_k + z(k)\pi$ .
20:              Computes  $\theta'_k = \delta_k - \phi'_k$ .
21:          else if  $k \in V_O$  then
22:              Computes  $\psi'_k = (-1)^{s_{inv}(k)} \psi_k + \underline{z}(k)\pi$ .
23:              Computes  $\theta'_k = \delta_k - \psi'_k$ 
24:          end if
25:          Applies  $HZ_k(\theta'_k)$  to half-EPR- $k$  followed by measurement.
26:          Stores the measurement outcomes as  $r_k$ .
27:          Sets  $s_k = \tilde{s}_k \oplus r_k$  and  $\tilde{\underline{s}}_k = \tilde{\underline{s}}_k \oplus r_k$  if  $k \in V_O$ .
28:      else  $\triangleright k \in O$ 
29:          Corrects  $\text{tr}_{O \setminus k}[\rho_B^{out}]$  with  $X_k^{s_{inv}(k)+t_k} Z_k^{z(k)+t(k)}$  and outputs it on Alice's interface.
30:      end if
31:      end for
32: end for

```

BOQC ALGORITHMS: A QUEST FOR THE MOST ECONOMICAL RESULTS

Up until this point, we have introduced the BOQC protocol in [Chapter 5](#), namely a protocol that helps Alice to privately delegate her oracular algorithm while Oscar is supplying her with the function evaluation embedded in his oracles. Also, we have discussed some oracle models and several oracular quantum algorithms in [Chapter 4](#). The question we are addressing in this chapter is: how does it look like in practice to delegate an “actual” oracular quantum computation in the BOQC scheme?

The underlying computation model of the BOQC is the 1WQC scheme (see [Section 2.1](#)); thus, Alice and Oscar need to represent their algorithm as a 1WQC computation as specified in [Definition 2.2](#), which comprises a graph with flow together with its measurement angles. Moreover, to maintain privacy, their computation must fulfill the criteria of BOQC-compatible graphs that are defined in [Section 5.2.1](#).

Our strategy in obtaining a BOQC-computation consists of two steps. Consider a set of unitary matrices implementing an algorithm. First, every unitary matrix is decomposed into a set of gates that are translatable into a graph state. Second, for the oracles, we generate a standard graph state that can implement all possible oracles in order to maintain the blindness. Notice that the second step is done only by Oscar.

To this date, many decomposition schemes work for an arbitrary size of unitary matrix; however, no algorithm is known yet that can efficiently perform any given combination of gates with optimal gate complexity. Instead, they perform well only for specific settings, constrained to the performance of the underlying physical device, such as the set of gates, the gates’ dimensions, the gates’ connectivity, and the circuit depth. Here, we develop a decomposition that aims to produce the most

economical circuit, where the minimal number of CPHASE gates determines the “economical” criterion. We approach this problem numerically, by extending the method of DiVincenzo and Smolin in [55]. Our decomposition constraints stem from the performance of NV-center qubits (Section 7.1), which is the physical system that we focus on.

In this chapter, we will walk through the aforementioned steps in details, and then produce several concrete “BOQC-algorithms” (the algorithms that can run within the BOQC scheme). Those algorithms include exact Grover algorithms (Section 4.3.2), a Simon algorithm (Section 4.5), and a Deutsch-Jozsa algorithm (Section 4.2).

6.1 DIVINCENZO AND SMOLIN’S SCHEME: EXTENDED

The DiVincenzo-Smolin scheme [55] is a systematic, exhaustive approach, that performs a decomposition of a unitary matrix in $SU(8)$ into a sequence of two-qubit gates ($SU(4)$ matrices) that is called a *two-bit gate network*. The scheme focuses on the decomposition of three-qubit gates, but it is extendable to unitary in arbitrary dimensions, *viz.*, $SU(2^n)$ for $n \in \mathbb{N}$. It had been believed that a three-qubit gate is necessary for universality, implied by the result of Deutsch [164], until some years later, DiVincenzo proved that two-qubit gates are also universal [16]. Moreover, in [164], the efficiency (in the gate-depth sense) of implementing a $SU(2^n)$ matrix is not addressed. The DiVincenzo-Smolin scheme addressed Deutsch’s issues by providing an explicit implementation to obtain a two-qubit network of a $SU(8)$ matrix. Here, we provide an extension of the DiVincenzo-Smolin scheme that works for an arbitrary dimensional unitary.

The gist of DiVincenzo-Smolin’s scheme lies in the non-linear minimization procedure of a gate network. A gate network is characterized by its two-bit *gate topology* that is defined in Definition 6.1. Now, let us discuss gate topologies in more detail.

Definition 6.1. [55] *A gate topology with size N is an N -tuple of positive integers $(i_1, i_2, \dots, i_N) := \mathcal{T}$, where each integer encodes a pair of qubits that correspond to a two-qubit gate, and i_j is an integer with Hamming weight 2.¹ In addition, two topologies $\mathcal{T}_1$ and $\mathcal{T}_2$ are called equivalent if the following conditions hold:*

(TR) Time reversal. *The time-reversed operation results in the inverse of the corresponding unitary. In the cases where the unitary is self-inverse, or*

¹ For example $3 = 00011$ and $24 = 11000$ are valid representation of two-qubit gates.

belongs to an ensemble which contains the inverse of every matrix, the inverse is equally powerful.

- (BR) Bit relabelling. *A unitary matrix is invariant with respect to relabeling the bits.*
- (CS) Conjugation by swapping. *Swapping two states of any pair of bits is a special two-bit logic gate [164], and this gate can be used to change the network topology. This operation is mostly useful in shortening the network.*

As there is much freedom in qubits indexing, we choose the standard integer-binary representation to associate elements of a gate topology with the corresponding two-qubit gates; we employ the bit endianness LSB (least significant bit), and the qubit indexing starts from 0. A two qubit-gate $\mathcal{G}_{ij}$ that is applied on qubits i and j is represented as the integer $2^i + 2^j$ in the gate topology, pictorially speaking we have

$$\equiv \cdots 0 \underbrace{1}_{\text{bit}-j} 0 \cdots 0 \underbrace{1}_{\text{bit}-i} 0 \cdots \underbrace{0}_{\text{bit}-0} \equiv 2^j + 2^i, \quad (6.1)$$

where $\mathcal{G}$ has a general form with 15 free-valued real parameters,² i.e., $\mathcal{G} \equiv \mathcal{G}(\phi_1, \dots, \phi_{15})$, where $\phi_i \in [0, 2\pi)$.

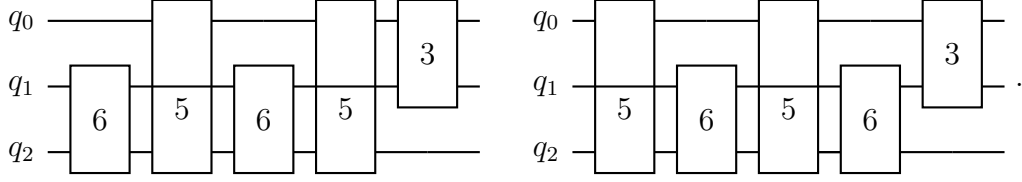
As the DiVincenzo-Smolín scheme numerically approaches the solution by brute-force, trying all possible topologies, the equivalence properties in Definition 6.1 become crucial. For instance, in the next section, we will find how such equivalence reduction becomes apparent, e.g., by comparing Table 6.1 and Table 6.2. Consider Example 6.1 to illustrate the equivalences in gate topologies.

Example 6.1. *The following equivalences are true: $(6, 5, 6, 5, 3) = (3, 5, 6, 5, 6)$, $(6, 5, 6, 5, 3) = (5, 6, 5, 6, 3)$, and $(6, 5, 6, 5, 3) = (6, 5, 3, 6, 3)$.*

Firstly, $(6, 5, 6, 5, 3) = (3, 5, 6, 5, 6)$ is true, clearly by time-reversal.

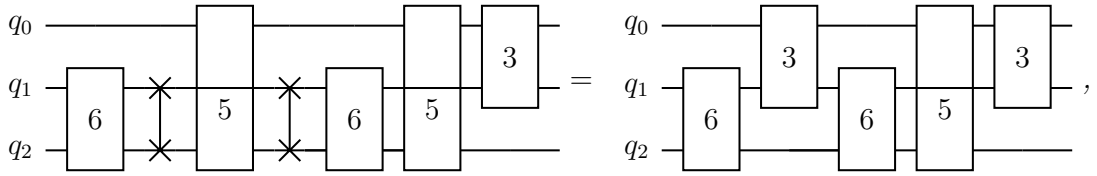
² A general unitary matrix $\text{SU}(m)$ has $m^2 - 1$ free parameters; thus, a general $\text{SU}(4)$ matrix has 15 real parameters.

Secondly, $(6, 5, 6, 5, 3) = (5, 6, 5, 6, 3)$ is true by bit relabelling. Consider the gate representation of both topologies:

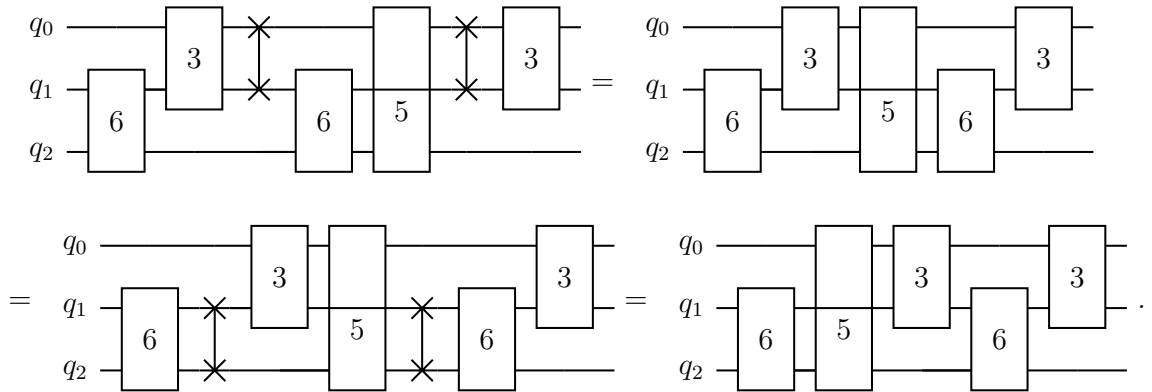


These circuits are equal by relabeling $q_0 \leftrightarrow q_1$.

Finally, $(6, 5, 6, 5, 3) = (6, 5, 3, 6, 3)$ is true by swap conjugation. First consider the gate representation of $(6, 5, 6, 5, 3)$, then perform swap gates before and after the third gate as follows



which are valid operations since the swap gates can be absorbed into gates 6. By adding more redundant swaps, we prove the equality as shown below.



The gate topology alone is usually not a sufficient representation for a gate decomposition. The compact yet general way is replacing each element in a gate topology with a general $SU(4)$ unitary, e.g., $\mathcal{G}(\phi_1, \dots, \phi_{15})$. Thus, a gate topology with size N has $15N$ free-valued parameters.

Algorithm 6.1 [55] DiVincenzo-Smolin decomposition (**DSdecomposition**)

Input: $M, \mathfrak{S}, N, f, \varepsilon$ $\triangleright$ target, gate set, size, objective function, precision
Output: $(\mathcal{T}, \vec{\Phi})$ or Fail

- 1: $\mathfrak{T} \leftarrow$ all unique n qubits topologies with size N $\triangleright$ see [Section 6.2](#) for details
- 2: **for** $\mathcal{T}$ in $\mathfrak{T}$ **do**
- 3: $\mathcal{N}(\vec{\Phi}_0) \leftarrow$ the built network of $\mathcal{T}$ composed of $\mathfrak{S}$, with initial angles $\vec{\Phi}_0$
- 4: $\vec{\Phi} \leftarrow$ a non-linear optimization result of $\mathcal{N}(\vec{\Phi}_0)$
- 5: **if** The optimization converges and $f \leq \varepsilon$ **then** $\triangleright$ See [Equation \(6.2\)](#)
- return** $\mathcal{T}, \vec{\Phi}$ $\triangleright$ The circuit is found
- 6: **end if**
- 7: **end for**
- return** Fail $\triangleright$ The circuit is not found with the given criteria

The DiVincenzo-Smolin scheme in [Algorithm 6.1](#) performs a decomposition of an $SU(2^n)$ unitary matrix M , sought within combinations of N two-qubit gates, and within precision ε .³ All of possible topologies are captured in variable $\mathfrak{T}$, where $|\mathfrak{T}| \leq \binom{n}{2}^N$, namely less than all possible topologies before eliminating the equivalent ones; this can be done by a particular search scheme, *e.g.*, [Algorithm 6.2](#). One may choose a suitable non-linear minimization scheme. For instance, DiVincenzo-Smolin scheme employs the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) [165] with the objective function defined as

$$f := \sum_i \sum_j |M_{ij} - S_{ij}|^2, \quad (6.2)$$

where S is the total resulting matrix from gate network $\mathcal{N}(\vec{\Phi})$, where $\mathcal{N}(\vec{\Phi})$ is constructed from $\mathcal{T}$ according to gates $\mathfrak{S}$. The successful optimization, *i.e.*, $|f| \leq \varepsilon$, returns the topology $\mathcal{T}$ and the optimized measurement angles $\vec{\Phi}$. Thus, the total operation from network gate $\mathcal{N}(\vec{\Phi})$ approximates M within error ε .

The DiVincenzo-Smolin scheme allows one to obtain arbitrary precision of decomposition. The scheme works particularly well for a small depth with a small number of qubits, as the algorithm grows exponentially in the depth. Moreover, a larger size topology exhausts the optimization, as the free parameter grows linearly in the depth, which quadratically increases the number of elements of the Hessian matrix.

³ The value of ε is determined by the outcome of the chosen objective function, *i.e.*, the Euclidean distance matrix like f . For this case, the best ε will be the machine precision, *e.g.*, the smallest floating point representable in `Python` is 2.225×10^{-308} with 53 digits precision.

6.2 ENUMERATING UNIQUE GATE TOPOLOGIES

Here, we provide an algorithm that enumerates all unique gate topologies, contributing to the first line of the DiVincenzo-Smolin scheme in [Algorithm 6.1](#), where the uniqueness is determined by [Definition 6.1](#). Before we jump into the algorithm, let us observe the number of all possible topologies without any elimination listed in [Table 6.1](#) below.

$\begin{array}{c} N \\ n \end{array}$	2	3	4	5	6	7	8	9
3	9	27	81	243	729	2187	6561	19683
4	36	216	1296	7776	46656	279936	1679616	10077696
5	100	1000	10^4	10^5	10^6	10^7	10^8	10^9

Table 6.1: The counting of all possible gate topologies with size N for n qubits, given by $\binom{n}{2}^N$.

The exponential growth of the entries in [Table 6.1](#) is apparent. Practically, it would take about 7 Gigabyte of memory, only to store the integers for the case of 5 qubits and 9 two-qubit gates. Nevertheless, let us imagine that we have a high capacity memory. The easy way to list the unique topologies is by listing all possible topologies, then eliminate every element that is equivalent with another element in the list by [Definition 6.1](#). This is certainly not a good strategy, as the complexity grows quadratically with the list size that grows exponentially. However, this is certainly not a problem for very small gate topologies.

Therefore, our attempt — in avoiding the quadratic, exponential algorithm — is iterating the unique topologies based on the criteria in [Definition 6.1](#). The scheme is provided in [Algorithm 6.2](#); the code implementation using `Python` is named `unique2net`, and is openly accessible in [\[166\]](#).

The structure of [Algorithm 6.2](#) is that it starts from the fewest number of edges, generating a list of non-isomorphic graphs right before the equivalence eliminations, then correspondingly add one edge to the previously resulting graphs; these steps are done for every iteration.

Algorithm 6.2 [166] Unique two-qubit gate topologies enumeration (**unique2net**)

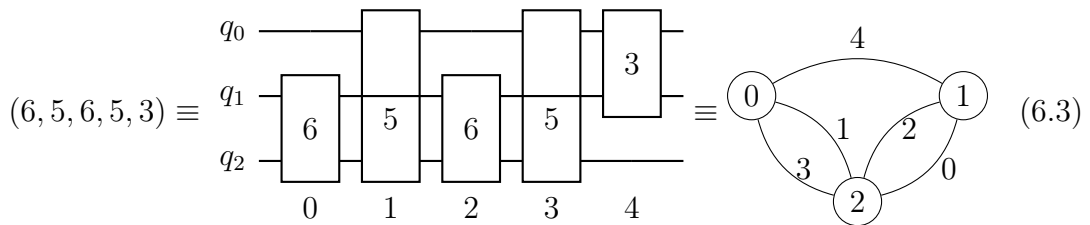
Input: n, N ▷ the number of qubits and final topology size
Output: $\mathcal{T}_N$

```

1:  $k \leftarrow 1$  ▷ initial topology size
2:  $\mathcal{T}_1 \leftarrow$  a list comprising an  $n$ -nodes graph with one edge
3: while  $k < N$  do
4:    $\mathcal{T}_{k+1} \leftarrow \emptyset$  ▷ a new empty list
5:   for  $\mathcal{T}$  in  $\mathcal{T}_k$  do ▷ loop through the old list
6:      $\tilde{\mathcal{T}} \leftarrow$  a list of topologies with one extra edge from  $\mathcal{T}$ 
7:     for  $\tilde{\mathcal{T}}$  in  $\tilde{\mathcal{T}}$  do
8:       if  $\exists$  4 identical edges consecutively then ▷ e.g., 566665
9:         break ▷ go to line 7
10:      end if
11:      for  $\mathcal{K}$  in  $\mathcal{T}_{k+1}$  do ▷ loop through the new list
12:        if  $\mathcal{K} \not\equiv \tilde{\mathcal{T}}$  and  $\mathcal{K} \not\stackrel{\text{CS}}{=} \tilde{\mathcal{T}}$  then ▷ by isomorphism (BR) and by CS
13:           $\mathcal{T}_{k+1} \leftarrow \mathcal{T}_{k+1} \cup \{\tilde{\mathcal{T}}\}$ 
14:        else
15:          break ▷ go to line 7
16:        end if
17:      end for
18:    end for
19:  end for
20:   $k \leftarrow k + 1$ 
21: end while
22: eliminate TR-criterion from  $\mathcal{T}_N$ , if necessary
    return  $\mathcal{T}_N$ 

```

Here, we introduce an alternative representation of two-bit gate networks as an undirected multi-edged graph, whose nodes correspond to qubits and edges correspond to two-qubit gates, as shown in the example below,



where each node i corresponds to qubit q_i , the edge numbers correspond to the time ordering of the two-qubit gates accordingly.

There are four equivalence eliminations in [Algorithm 6.2](#): the one with four identical edges that have consecutive ordering and three topology equivalence criteria in [Definition 6.1](#). The first elimination comes from the fact that four consecutive two-qubit gates (in a general form) can be realized with three gates [\[55\]](#); for example, topology $(5, 6, 6, 6, 6, 5) = (5, 6, 6, 6, 5)$. The second elimination is the following.

In the graph representation, it is straightforward to see that the graph isomorphism that one can think of as relabeling the nodes, corresponds to the **BR**-criterion. It is important to remark that the order of quantum gates matters; the graph isomorphism here must preserve the edge order. Then, the resulting non-isomorphic graphs are filtered according to **CS**-criterion. Finally, in every iteration k , the resulting $\mathcal{T}_k$ contains unique topologies with respect to **BR**- and **CS**-criteria. For an illustration, consider [Figure 6.1](#), which shows the resulting graphs in every iteration.

The last elimination is based on the **TR**-criterion, which is done to the final list generated from the iteration. For example, for $n = 3$ and $N = 4$, the unique topologies can be seen in [Figure 6.1](#), which comprises 9 topologies; this is much fewer than 81, *viz.*, the counting in [Table 6.1](#). More examples in counting unique topologies are presented in [Table 6.2](#). Note that, in some cases, the **TR**-criterion does not hold and should not be applied, for instance, in the case of Simon’s algorithm in [Section 6.5](#).

$\begin{matrix} N \\ n \end{matrix}$	2	3	4	5	6	7
3	2	4	9	17	29	43
4	3	9	35	151	663	2779
5	3	11	62	441	3519	28597

Table 6.2: The number of unique topologies with size N for n qubits. Calculated from the output of [Algorithm 6.2](#), implemented with `unique2net`[\[166\]](#).

The complexity of our algorithm is, unfortunately, not of polynomial-time, since it continually performs loops over the newly generated list ($\mathcal{T}_{k+1}$). However, there are advantages here, for instance it can be straightforwardly parallelized, it prevents adding up equivalent topologies, and one does not necessarily start from one edge ($k = 1$) but from a previously-stored list instead, *i.e.*, one can store $\mathcal{T}_{k+1}$ for every k , where the list contains unique topologies up to bit-relabelling and swap conjugation.⁴

⁴ This strategy is often referred to “dynamic programming.”

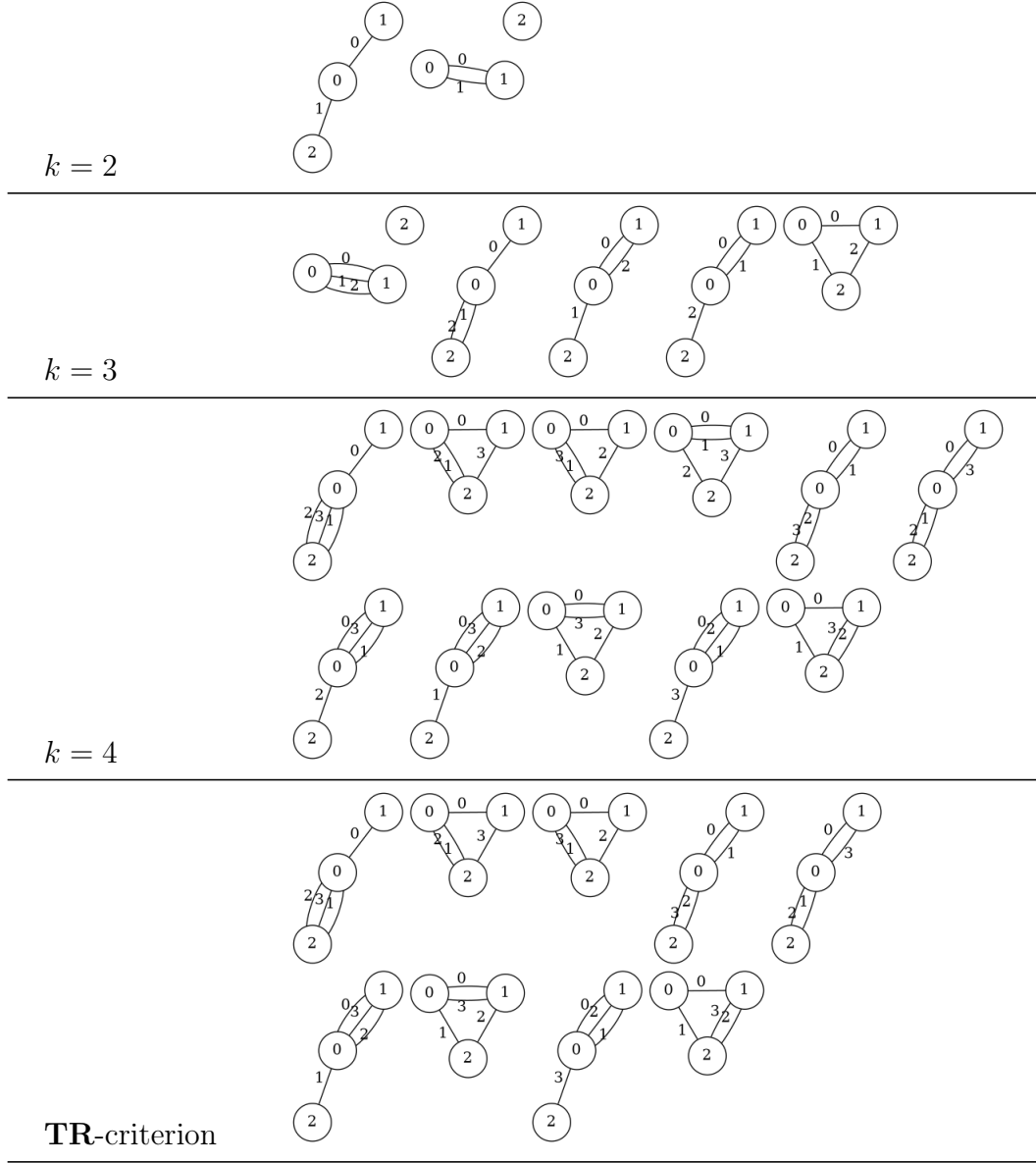


Figure 6.1: The intermediate (during iteration phase) and final gate topologies within graph representation, for $n = 3$ and $N = 4$. The first three rows show results in every iteration k , which do not consider **TR**-criterion. The last row shows the final results after **TR**-criterion. These figures are the output of `unique2net` program [166].

6.3 EXHAUSTIVE SEARCH FOR THE MOST ECONOMICAL (BLIND) GRAPH STATES

To realize an arbitrary unitary within the 1WQC scheme (see [Section 2.1](#)), we are faced with infinite choices of graph states and angles. An efficient algorithm has been proposed by Danos, Kashefi, and Panangaden in [\[167\]](#); the algorithm produces a graph state of the corresponding unitary, where 14 nodes and 2 CPHASE gates are required to realize a controlled- U , for an $U \in \text{SU}(2)$. Here, we consider more complex unitary matrices that involve blind oracles, while demanding even much fewer node numbers; in the end, we want to demonstrate BOQC computations on a physical device. Therefore, we approach this problem numerically instead.

We are optimistic that NV-centers ([Section 7.1](#)) have high potential to be the platform of the BOQC scheme, for their progress in single qubit control and quantum networking (further discussed in [Section 7.1](#)). To this date, NV-center systems are capable of producing a few qubits, which, unfortunately, decohere quickly with the number of operations. Therefore, we aim to produce the most economical graph state, with a minimal number of two-qubit gates and a minimal number of nodes.

We extend the DiVincenzo-Smolín algorithm ([Algorithm 6.1](#)) to construct a scheme to obtain the most economical graph states, that also fulfill the BOQC compatibility ([Section 5.2.1](#)). Our scheme is called BGS (*BOQC-compatible graphs search*), shown in [Algorithm 6.3](#); it is an explicit and generalized version of the BEQS (*blind exact quantum search*) algorithm introduced in [\[143\]](#). The BGS scheme heavily employs [Subroutine 6.1](#), so-called **DSoptimization**, which is a part of the DiVincenzo-Smolín algorithm. [Subroutine 6.1](#) optimizes the free-valued parameters to realize a unitary matrix M within ε , by a gate network that is built from a topology $\mathcal{T}$ and a gate set $\mathfrak{G}$, *e.g.*, [Equation \(6.4\)](#) and [Equation \(6.5\)](#). The objective function f can be defined accordingly, *e.g.*, [Equation \(6.2\)](#). However, before proceeding to the BGS algorithm, let us clarify several notations and conventions that we use here.

First, we have not discussed how one obtains the gate network of a gate topology explicitly, even though we have mentioned it several times beforehand; *e.g.*, it contributes to the first line of [Subroutine 6.1](#). There are two gate sets that we use here, namely

$$\mathfrak{G} := \{\text{CNOT}, U(\alpha, \beta, \gamma)\} \quad (6.4)$$

$$\mathfrak{J} := \{\text{CPHASE}, J(\phi)\}, \quad (6.5)$$

Subroutine 6.1 Optimization part of [Algorithm 6.1](#) (DSoptimization)**Input:** $\mathcal{T}, M, \mathfrak{S}, f, \varepsilon$ **Output:** Φ or Fail

- 1: $\mathcal{N}(\vec{\Phi}_0) \leftarrow$ the built network of $\mathcal{T}$ composed of $\mathfrak{S}$, with initial angles $\vec{\Phi}_0$
- 2: $\vec{\Phi} \leftarrow$ a non-linear optimization result of $\mathcal{N}(\vec{\Phi}_0)$
- 3: **if** the optimization converges and $f \leq \varepsilon$ **then**
 return $\vec{\Phi}$
- 4: **else**
 return Fail ▷ The circuit is not found with the given criteria
- 5: **end if**

with the free-valued parameters $\alpha, \beta, \gamma, \phi \in (0, 2\pi]$, the single unitary matrix

$$U(\alpha, \beta, \gamma) := \begin{pmatrix} e^{i\beta} \cos(\alpha) & e^{i\gamma} \sin(\alpha) \\ -e^{-i\gamma} \sin(\alpha) & e^{-i\beta} \cos(\alpha) \end{pmatrix}, \quad (6.6)$$

and

$$J^m(\phi) := \prod_{k=1}^m \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & e^{i\phi_k} \\ 1 & -e^{i\phi_k} \end{pmatrix}, \quad (6.7)$$

where $m \in \{0, 1, 2, 3\}$ is the multiplicity of occurrence and $J(\phi)$ is obtained at $m = 1$; equivalently, $J(\phi) = HR_z(\phi)$. Operator J covers the entire $SU(2)$ [167], *i.e.*, it can obtain arbitrary unitary since $J(0)J(\phi_3)J(\phi_2)J(\phi_1) = R_z(\phi_3)R_x(\phi_2)R_z(\phi_1)$,⁵ up to a global phase. A unitary $J(\phi)$ can be realized with pattern $X_1^{s_0}M_0^{-\phi}E_{01}N_1$, which is one measurement in $|\pm_{(-\phi)}\rangle$ basis together with the correction.⁶

Since we approach the decomposition numerically, we only specify three free-valued parameters ϕ_1, ϕ_2, ϕ_3 to approximate any 1-qubit unitary, which results in $J(\phi_3)J(\phi_2)J(\phi_1) = HR_z(\phi_3)R_x(\phi_2)R_z(\phi_1)$. Thus, our decomposition can be denoted using an extended gate topology notation as defined [Definition 6.2](#),⁷ to encode the gate network; for example $(\emptyset^2, 6, 5, 6^0, 5^3, 3^3)$, where unspecified multiplicity signifies $m = 1$, and $m = 0$ means replacing J with identity matrices. Here, the element $\emptyset$ signifies applying J to every qubit *i.e.*, operation $\bigotimes_{i=0}^{n-1} J(\phi_i)$. The elements of $\mathfrak{J}$ are to be applied in order, *i.e.*, first CPHASE, then the J operators. We can consider the presence of $\emptyset$ in a topology as an indicator of a gate decomposition construction using set $\mathfrak{J}$.

⁵ This corresponds to Euler rotation theorem [69].

⁶ This is explicitly shown in [Example 2.1](#)

⁷ Note that the encoding does not include angles, instead the optimization is performed over angles for every possible encoding.

Definition 6.2. *Extended gate topology notation.* An extended gate topology is the gate topology defined in [Definition 6.1](#) with the additional information of multiplicity on each topology element. An extended gate topology with size N has the form

$$(\emptyset^{m_0}, z_1^{m_1}, \dots, z_N^{m_N}) =: \mathcal{T}, \quad (6.8)$$

where $z_i \in \mathbb{N}$, multiplicity $m_i \in \{0, 1, 2, 3\}$, and the integer z_i has the same encoding as [Definition 6.1](#), thus the equivalence criteria hold.

As the two-qubit gates in $\mathfrak{G}$ and $\mathfrak{J}$ do not contain any free-valued parameters, we append two one-qubit unitary matrices after each application of CNOT or CPHASE gate. Note that, for the construction with $\mathfrak{G}$, we always start with applying U to each qubit, *i.e.*, $\bigotimes_{i=0}^{n-1} U(\vec{\phi}_i)$, whereas for another scheme with $\mathfrak{J}$, it is explicitly specified with element $\emptyset$.

Such a construction results in the following growth of free-valued parameters. Given the gate network $\mathcal{N}(\vec{\Phi})$ of n qubits constructed from a topology $\mathcal{T}$ with gate set $\mathfrak{G}$, the free valued-parameters' size $|\vec{\Phi}|$ is

$$3n + 2|\mathcal{T}|. \quad (6.9)$$

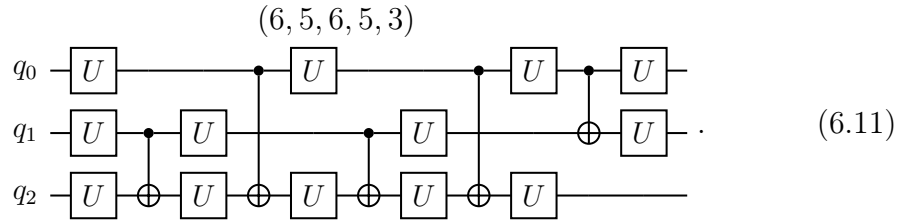
Now, given that $\mathcal{T}$ is constructed with $\mathfrak{J}$, $\mathcal{T}$ also contains multiplicity information, *i.e.*, $\mathcal{T} = (\emptyset^{m_0}, z_1^{m_1}, \dots, z_N^{m_N})$, the value of $|\vec{\Phi}|$ becomes

$$nm_0 + 2 \sum_{i=1}^N m_i, \quad (6.10)$$

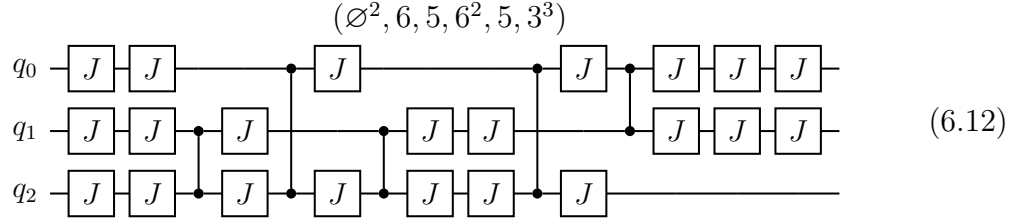
which is also the upper bound of the number of nodes in the resulting graph state. Consider [Example 6.2](#) for some illustrations.

Example 6.2. *In the following, we obtain the gate networks from topology $\mathcal{T}_1 := (6, 5, 6, 5, 3)$ with gate set $\mathfrak{G}$, and topology $\mathcal{T}_2 := (\emptyset^2, 6, 5, 6^2, 5, 3^3)$ with gate set $\mathfrak{J}$. We also make note of the number of free parameters for each topology.*

Obtaining the gate network with set $\mathfrak{G}$ from topology $(6, 5, 6, 5, 3)$ means the following maps, where $U \equiv U(\alpha, \beta, \gamma)$.



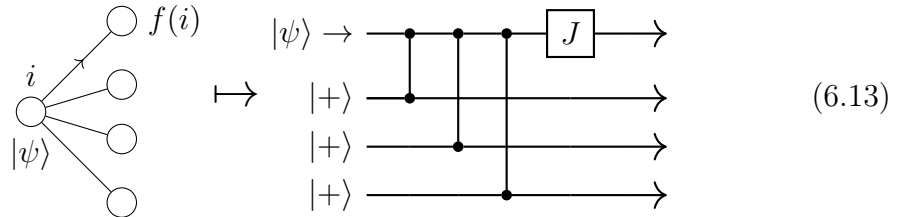
Obtaining the gate network with gate set $\mathfrak{G}$ from topology $(\emptyset^2, 6, 5, 6^2, 5, 3^3)$ means the following maps, where $J \equiv J(\phi)$.



Topology $\mathcal{T}_1$ has 39 free-valued parameters, and topology $\mathcal{T}_2$ has 22 free-valued parameters.

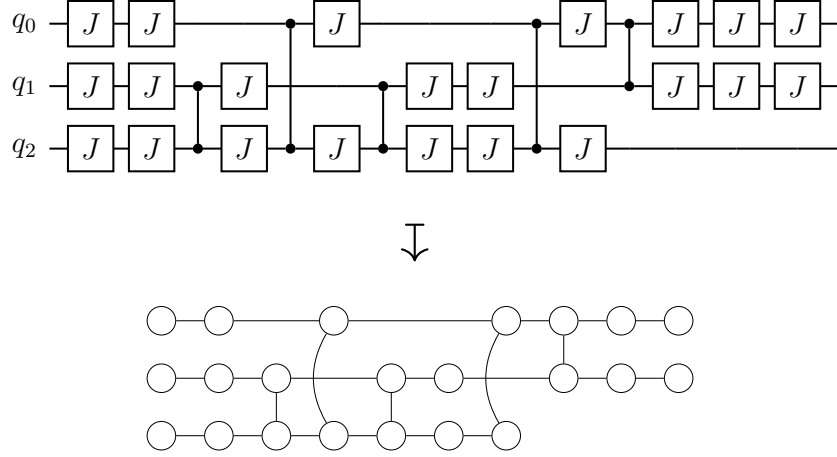
Here, we implement with two types of gate decompositions: composition with gate set $\mathfrak{G}$ and $\mathfrak{J}$. The first decomposition aims to find the lowest number of CNOT gates, in order to obtain minimal depth of the graph. The second decomposition aims to find the minimal number of graph nodes, where the results can be directly converted into a graph.

Recall from [Section 2.4](#) that we consider only graphs with a flow; therefore, a direct translation between a circuit that comprises gates $\mathfrak{J}$ and a graph state can be performed [\[71\]](#). This translation can be arrived at by observing the resulting quantum circuit for the following star graph, where the center node i is measured, and $f(i)$ indicates the flow.



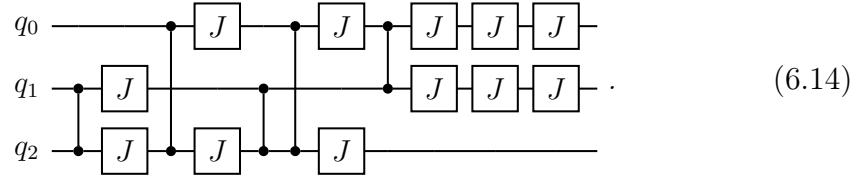
Assuming that the measurement outcome of i is 0 — no correction is needed, operator J is indirectly applied to qubit $f(i)$ while $|\psi\rangle$ is indirectly inputted to $f(i)$. Since node i is measured, it does not appear anymore on the circuit, but the state is propagated into $f(i)$. Consider [Example 6.3](#) to illustrate such a translation applied to some gate topologies.

Example 6.3. The graph state of a topology $(\emptyset^2, 6, 5, 6^2, 5, 3^3)$ can be straightforwardly obtained from the corresponding circuit, namely



Notice that in our construction, to increase a multiplicity in [Definition 6.2](#) means adding extra n nodes.⁸ The following trick can be used to reduce the number of nodes. Recall that each J has one parameter ϕ , which is the measurement angle $-\phi$; if two consecutive J operators have zero angles, we can eliminate both nodes (both J). Notice that a zero angle produces Hadamard operation, $J(0) = H$, thus, $J(0)J(0) = HH = I$. Consider [Example 6.4](#) for an illustration.

Example 6.4. *Given that some operators J in [Example 6.3](#) ([Equation \(6.12\)](#)) have parameter zero; we remove several of them accordingly which results in the following circuit*



Using the same principle as [Equation \(6.13\)](#), we obtain the following graph state,



where the arrows indicate the flow.⁹ The topology of the circuit is $(\emptyset^0, 6, 5, 6^0, 5, 3^3)$.

⁸ It provides simpler notation but inefficient graph state.

⁹ The flow induces a partial ordering $\succ$; the measurement order must follow $\succ$.

Note that, the resulting graphs from our construction always have a flow, because we start from a circuit, which guarantees causality and determinism. The non-existence of the flow is the result of graph acausality by the construction defined in [71] — see [Section 2.4](#).

Now, we are ready to present the BGS algorithm, shown in [Algorithm 6.3](#). BGS algorithm is an exhaustive method that comprises several search layers. Referring

Algorithm 6.3 BOQC-compatible graphs search (BGS)

Input: $\mathcal{U}, \mathcal{O}$ ▷ a set of unitary matrices, a set of oracles

Output: $\{\mathcal{G}_M\}, \{\mathcal{G}_{O_x}\}$, for all $M \in \mathcal{U}, O_x \in \mathcal{O}$

(S1) [Alice, Oscar] Estimate the lower bound of CNOT gates (equivalently CPHASE gates) for all given unitary matrices ($\mathcal{U} \cup \mathcal{O}$), given an objective function f — e.g., [Equation \(6.2\)](#), precision ε , and gate set $\mathfrak{G}$ ([Equation \(6.4\)](#)).

```

1: for  $M$  in  $\mathcal{U} \cup \mathcal{O}$  do
2:    $N_M \leftarrow 0$  ▷ topology size; size 0 means 1-qubit unitaries only
3:    $\mathcal{T}_M, \vec{\Phi}_M \leftarrow \text{DSdecomposition}(M, \mathfrak{G}, N_M, f, \varepsilon)$  ▷ Algorithm 6.1
4:   while  $(\mathcal{T}_M, \vec{\Phi}_M) = \text{Fail}$  do ▷ keep searching if the first guess fails
5:      $N_M \leftarrow N_M + 1$  ▷ increase topology size one by one
6:      $\mathcal{T}_M, \vec{\Phi}_M \leftarrow \text{DSdecomposition}(M, \mathfrak{G}, N, f, \varepsilon)$  ▷ Algorithm 6.1
7:   end while
8: end for

```

♣ At this point, we know $\mathcal{T}_M$ for all $M \in \mathcal{U} \cup \mathcal{O}$.

(S2) [Alice] Obtain graph states for all unitary matrices $M \in \mathcal{U}$ (non-oracle) by decomposing M into gates $\mathfrak{J}$ ([Equation \(6.5\)](#)). From S1, we know topology $\mathcal{T}_M$ can realize M . Here, we are given $\mathcal{T}_M = (z_1, \dots, z_N)$.

```

9: for  $M$  in  $\mathcal{U}$  do
10:   $\vec{m} \leftarrow \vec{1}$ , where  $|\vec{m}| = N + 1$  ▷ multiplicities
11:   $\tilde{\mathcal{T}}_M \leftarrow (\emptyset^{m_0}, z_1^{m_1}, \dots, z_N^{m_N})$  ▷ Definition 6.2
12:   $\vec{\Phi}_M \leftarrow \text{DSoptimization}(\tilde{\mathcal{T}}_M, M, \mathfrak{J}, f, \varepsilon)$  ▷ Subroutine 6.1
13:  while  $\vec{\Phi}_M = \text{Fail}$  do ▷ keep searching if the first guess fails
14:     $m_i \leftarrow m_i + 1$  ▷ increase one element of  $\vec{m}$  accordingly, up to  $m_i = 3 \forall i$ 
15:     $\tilde{\mathcal{T}}_M \leftarrow (\emptyset^{m_0}, z_1^{m_1}, \dots, z_N^{m_N})$ 
16:     $\vec{\Phi}_M \leftarrow \text{DSoptimization}(\tilde{\mathcal{T}}_M, M, \mathfrak{J}, f, \varepsilon)$ 
17:  end while
18: end for

```

♦ At this point, we can conclude the graph states from $\tilde{\mathcal{T}}_M$, together with measurement angles $\vec{\Phi}_M$, for all $M \in \mathcal{U}$.

(S3) [Oscar] Obtain a common graph state for all oracles in $\mathcal{O}$. The set $\mathcal{O} := \{O_x\}$ comprises oracles for all possible inputs, *i.e.*, unitary $O_x \in \mathcal{O}$ is the matrix representation of oracle evaluation with input x . The minimizations take into account all cases of x , by specifying an objective function accordingly. For example, one may have an objective function as

$$F := \sum_x \sum_{i,j} |(O_x)_{ij} - (S_x)_{ij}|^2, \quad (6.16)$$

which is the objective sum (Equation (6.2)) for all cases x . This function is more expensive than separated optimizations, since the size of free-valued parameters is $|\mathcal{O}|$ -times of a single O_x . This step comprises two parts resembling **S1** and **S2**, but with a more extensive process. First, we find the smallest topology for the blind circuit (a common topology for all cases x such that only the angles that differ). And then, we perform the decomposition to obtain a blind graph state.

(S3.1) Create a set of ansatz $\mathfrak{B} := \{\mathcal{B}_i\}$, concluded from $\{\mathcal{T}_x\}$ accordingly, where the elements of $\mathfrak{B}$ are ordered from the smallest size.

19: **for** $\mathcal{B}$ **in** $\mathfrak{B}$ **do**

20: $\{\mathcal{N}(\vec{\Phi}_x^0)\} \leftarrow$ network gates of $\mathcal{B}$ composed of $\mathfrak{G}$, with initial angles $\vec{\Phi}_x^0$, for each x

21: $\{\vec{\Phi}_x\}, F \leftarrow$ a non-linear optimization result of $\{\mathcal{N}(\vec{\Phi}_x^0)\}$

22: **if** the optimization converges **and** $F \leq \varepsilon$ **then**

return $\mathcal{B}$

$\triangleright$ quit once it is found

23: **end if**

24: **end for**

♥ At this point, we found the blind topology $\mathcal{B}$. In an unlucky event, **S3.1** fails, which means the ansatz $\mathfrak{B}$ is too small; thus, we need to extend the ansatz.

(S3.2) Obtain the blind graph for oracles $\mathcal{O}$ from blind topology $\mathcal{B}$. Here, we are given $\mathcal{B} = (b_1, \dots, b_N)$.

25: $\vec{m} \leftarrow 1$, where $|\vec{m}| = N + 1$

26: $\tilde{\mathcal{B}} \leftarrow (\mathcal{O}^{m_0}, z_1^{m_1}, \dots, z_N^{m_N})$

27: $\{\mathcal{N}(\vec{\Phi}_x^0)\} \leftarrow$ network gates of $\mathcal{B}$ composed of $\mathfrak{J}$, with initial angles $\vec{\Phi}_x^0$, for each x

28: $\{\vec{\Phi}_x\}, F \leftarrow$ a non-linear optimization result of $\{\mathcal{N}(\vec{\Phi}_x^0)\}$

29: **while** the optimization does not converge **or** $F > \varepsilon$ **do**

30: $m_i \leftarrow m_i + 1$ $\triangleright$ increase one element of $\vec{m}$ accordingly, up to $m_i = 3 \forall i$

31: $\tilde{\mathcal{B}} \leftarrow (\mathcal{O}^{m_0}, z_1^{m_1}, \dots, z_N^{m_N})$

32: $\{\mathcal{N}_x(\vec{\Phi}_0)\} \leftarrow$ network gates of $\mathcal{B}$ composed of $\mathfrak{J}$, with initial angles $\vec{\Phi}_0$, for all x

33: $\{\vec{\Phi}_x\}, F \leftarrow$ a non-linear optimization result of $\{\mathcal{N}_x(\vec{\Phi}_0)\}$

34: **end while**

♠ Finally, we found the blind oracles with a common graph state constructed from $\tilde{\mathcal{B}}$, with measurement angles $\{\vec{\Phi}_x\}$.

to [Algorithm 6.3](#), Alice performs parts **S1** and **S2**, whereas Oscar performs **S1** and **S3** (**S3.1** and **S3.2**). Oscar's processing is more extensive than Alice's part since his optimizations at part **S3** simultaneously assess all oracles $\{O_x\}$ for all inputs x . We recall that the BOQC scheme assumes Oscar to possess high classical power (see [Figure 5.1](#) for an illustration).

The BGS scheme requires us to express our algorithm as a sequential application of unitary matrices. There, Alice's unitary matrices are captured in $\mathcal{U}$, and she will obtain graphs from $\{\mathcal{T}_M\}$ with measurement angles $\{\vec{\Phi}_M\}$, for all $M \in \mathcal{U}$. Meanwhile, Oscar's oracles are a collection of unitaries with all possible outcomes $\{O_x\}$, and he will obtain one graph that is constructed from $\tilde{\mathcal{B}}$, together with the measurement angles for each request $\{\vec{\Phi}_x\}$. From these graphs, Alice and Oscar can perform further optimization to reduce the node number by removing the two consecutive nodes that have measurement angles close to zero — as done in [Example 6.4](#) — and then re-optimize it.

The BGS algorithm is a scheme that works for oracular quantum algorithms in general. One can adjust it, *e.g.*, by using a particular objective function and optimization scheme. We will directly see how the BGS scheme is implemented on explicit algorithms in the next sections.

6.4 BLIND GROVER-HØYER ALGORITHM

In this section, we construct two BOQC algorithms that solve an unordered search problem ([Problem 4.4](#)) exactly. Given a database $\mathbf{d}$ represented by n qubits, we aim to find an item x with certainty. We solve this problem for $n = 2$ and $n = 3$, where the first holds for $|\mathbf{d}| = 4$, and the latter corresponds to $|\mathbf{d}| \in \{5, 6, 7, 8\}$. The first case is solvable by Grover's algorithm ([Equation \(4.11\)](#)), whereas the second case is solved with the Grover-Høyer algorithm ([Algorithm 4.1](#)). Since we know the unitary matrices of both algorithms — [Equation \(4.11\)](#) and [Algorithm 4.1](#), we obtain the BOQC-compatible graph states using BGS algorithm defined in [Algorithm 6.3](#).

We encode each item of a database as a binary string with the least significant bit convention, *i.e.*,

$$\begin{array}{l} q_0 \text{ ---} \\ q_1 \text{ ---} \\ \vdots \text{ ---} \\ q_{n-1} \text{ ---} \end{array} \equiv q_{n-1} \dots q_1 q_0 \equiv \sum_{i=0}^{n-1} 2^{q_i}, \text{ where } q_i \in \{0, 1\}. \quad (6.17)$$

We will interchangeably use binary and integer representation. Thus, a database with size N comprises N items that are represented with integers, *i.e.*, $\mathbf{d} = \{d_1, \dots, d_N\}$, where $d > 1$. An item $d \in \mathbf{d}$ has a quantum state representation as $|q_{n-1} \dots q_0\rangle$ or $|x\rangle$, where x is the integer representation of the index of d . Notice that we separate the notion of items and integer representation of a database element. A database comprises items where an item can be anything, where each item is accompanied by an index. However, here, we always refer to an index while referring to an item.

Given $|\mathbf{d}| =: N$ with $N \leq 2^n$, there are $\binom{2^n}{N}$ degrees of freedom in choosing the indices; here we set $2^{n-1} < N \leq 2^n$. Our focus lies in the cases of $n = 2$ and $n = 3$; there is one choice database for $n = 2$,¹⁰ *i.e.*, $\{0,1,2,3\}$, whereas there are more choices for the three qubits database: 56 choices for $N = 5$, 28 choices for $N = 6$, 8 choices for $N = 7$, and 1 choice for $N = 8$. Therefore, to simplify the notation, we write the database as follows

$$\mathbf{d} := \{d_1, d_2, \dots, d_N\} \equiv d_1 d_2 \dots d_N, \text{ where } \mathbf{d} \subset \Sigma^n. \quad (6.18)$$

Theoretically speaking, choosing a subset of indices to encode a database is arbitrary; however, it matters in practice, *i.e.*, different item encoding leads to different gates composition. Thus, before proceeding to the algorithm, we search for the most economical set of indices to encode the database based on the operation of oracles. The most convincing strategy is to try all possible combinations, and then compare the gate complexity for each case — which is what we do here.

Intuitively, two databases $\mathbf{d}, \mathbf{e}$, where $|\mathbf{d}| = |\mathbf{e}|$, are equivalent if a set of oracles that can “mark” for all $x \in \mathbf{d}$ can also mark for all $y \in \mathbf{e}$ up to some global phases. The notion of marking here means an operation that adds a phase $e^{i\phi}$ to the solution, *i.e.*, $\mathcal{O}_f |x\rangle = e^{i\phi} |x\rangle$ if and only if x is a solution (see [Section 4.3.2](#)). For an explicit example, consider [Example 6.5](#). Thus, by considering the binary representation, $\mathbf{d}$ is equivalent to $\mathbf{e}$ if they are identical up to permutation and bit complementation. For example, consider two equivalent databases within the binary representation: $d_{01234} = \{000, 001, 010, 011, 100\}$, $d_{10543} = \{001, 000, 101, 100, 011\}$. One can be obtained from another by complementing the third bit and

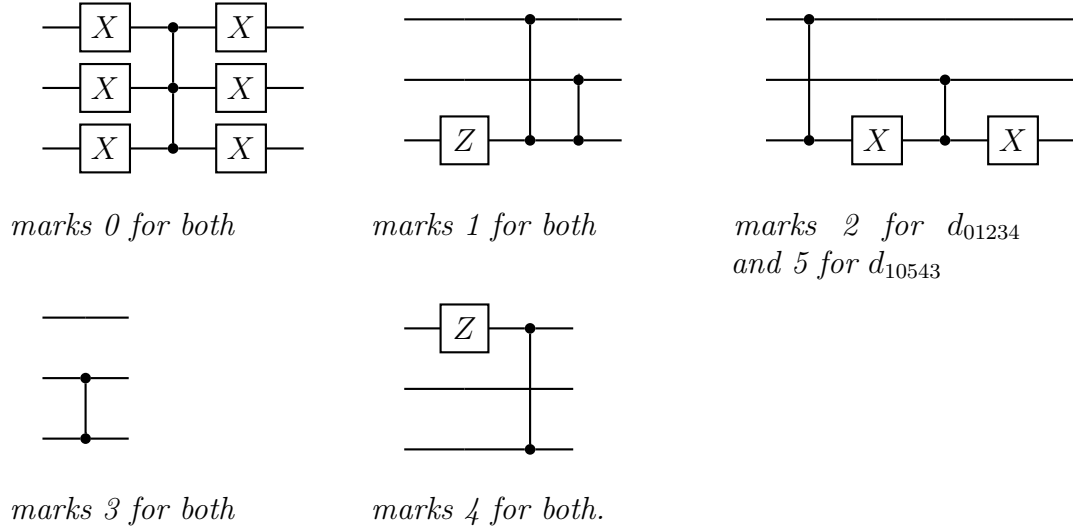
¹⁰ We can also have $N = 3$, *e.g.*, the database $\{0,2,3\}$. We just do not bother analyzing it.

permuting the first and the second bits. In the circuit representation, it means their oracles are equivalent up the following operations:

$$\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{|c|} \hline \mathcal{O}_{01234} \\ \hline \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \times \\ \text{---} \times \\ \text{---} \end{array} \begin{array}{|c|} \hline \mathcal{O}_{10543} \\ \hline \end{array} \begin{array}{c} \text{---} \times \\ \text{---} \times \\ \text{---} \end{array}, \quad (6.19)$$

where $\mathcal{O}_{01234}$ and $\mathcal{O}_{10543}$ represent the oracle operators for d_{01234} and d_{10543} , respectively.

Example 6.5. *Given database combinations d_{01234} and d_{10543} are equivalent; we can confirm this by showing that there exists a set of oracles that marks every element in d_{01234} also distinctly marks every element in d_{10543} . Such a set of oracles exists as follows (*Mathematica* code is available in [Section B.4](#)).*



The equivalence of database combinations can be generated by combining all possible permutations and bit complementations. [Table 6.3](#) shows the equivalences of databases in a row. Thus, by removing those equivalences, we can obtain a set of unique database combinations, which is the implemented strategy in [Algorithm 6.4](#).

From [Table 6.3](#), we conclude that there are eight unique databases for $n = 3$, namely

$$\mathcal{D} := \{01234, 01247, 01256, 012345, 012347, 012567, 0123456, 01234567\}. \quad (6.20)$$

N	Equivalent combinations
5	01234, 01235, 01236, 01237, 01245, 01246, 01345, 01357, 01456, 01457, 02346, 02367, 02456, 02467, 04567, 12357, 12367, 13457, 13567, 14567, 23467, 23567, 24567, 34567
	01247, 01356, 02356, 03456, 03567, 12347, 12457, 12467
	01256, 01257, 01267, 01346, 01347, 01367, 01467, 01567, 02345, 02347, 02357, 02457, 02567, 03457, 03467, 12345, 12346, 12356, 12456, 12567, 13456, 13467, 23456, 23457
6	012345, 012346, 012357, 012367, 012456, 013457, 014567, 023467, 024567, 123567, 134567, 234567
	012567, 013467, 023457, 123456
	012347, 012356, 012457, 012467, 013456, 013567, 023456, 023567, 034567, 123457, 123467, 124567
7	0123456, 0123457, 0123467, 0123567, 0124567, 0134567, 0234567, 1234567
8	01234567

Table 6.3: Equivalent databases for $N \in \{5, 6, 7, 8\}$. The equivalent combinations are listed within the same row. This table has already appeared in reference [143]

Algorithm 6.4 Unique databases with size N (**uniquedb**)

Input: N

Output: $\mathcal{D}$

```

1:  $n \leftarrow \lceil \log N \rceil$ 
2:  $\mathcal{D} \leftarrow \{\mathbf{d} \in \text{Combinations}(\{0, \dots, 2^n - 1\}) \mid |\mathbf{d}| = N\}$ ,  $|\mathcal{D}| = \binom{2^n}{N} =: M$ 
3:  $\mathcal{C} \leftarrow \mathcal{P}(\{0, \dots, n - 1\})$  ▷ powerset, for bit complement
4:  $\mathcal{P} \leftarrow \text{Permutations}(\{0, \dots, n - 1\})$  ▷ all bit permutations
5:  $\mathcal{CP} \leftarrow \mathcal{C} \times \mathcal{P}$  ▷ all combination of bit complements and permutations
6: for  $\mathbf{d}$  in  $\mathcal{D}$  do
7:   if  $\mathbf{d} \neq \text{False}$  then
8:      $\mathfrak{d} \leftarrow \{\text{Permute}(\text{Complement}(\mathbf{d}, C), P) \mid (C, P) \in \mathcal{CP}\}$  ▷  $d$ -equivalences
9:      $i \leftarrow \text{indexof}(\mathbf{d})$ 
10:    for  $e$  in  $\{d_{i+1}, d_{i+2}, \dots, d_M\}$  do ▷ given  $\mathcal{D} := \{d_1, \dots, d_M\}$ 
11:      if  $e \in \mathfrak{d}$  then
12:         $\mathcal{D}[\text{indexof}(e)] \leftarrow \text{False}$ 
13:      end if
14:    end for
15:  end if
16: end for
17:  $\mathcal{D} \leftarrow \{\mathbf{d} \in \mathcal{D} \mid \mathbf{d} \neq \text{False}\}$  ▷ filter the equivalences
   return  $\mathcal{D}$ 

```

Note that, for $n = 2$, there are two choices, *i.e.*, $\{012, 0123\}$; here we consider only database 0123.

Recall from Equation (4.42) in Section 4.4, Grover-Høyer algorithm for $n = 3$ can be described by a sequence of unitary applications as follows

$$|0\rangle^{\otimes 3} \Rightarrow \boxed{\mathcal{A}} \Rightarrow \boxed{\mathcal{O}} \Rightarrow \boxed{D} \Rightarrow \boxed{\mathcal{O}(\varphi + \zeta)} \Rightarrow \boxed{D(\psi)} \Rightarrow \boxed{\text{Measurement}}, \quad (6.21)$$

$\underbrace{\hspace{10em}}_G \qquad \underbrace{\hspace{10em}}_{Q(\varphi, \psi)}$

where all variables and operators are discussed in Section 4.3.2. To summarize, $\mathcal{A}$ is the preparation operator (Equation (4.21)) that transforms state $|0\rangle^{\otimes 3}$ into an equal superposition of quantum states, $Q(\varphi, \psi)$ is Høyer's amplitude amplification, $G = Q(\pi, \pi)$ is Grover's iterator, $\mathcal{O}$ is the parameterized oracle (Equation (4.24)), and D is the parameterized diffusion operator (Equation (4.25)). The Høyer parameters φ, ψ , and ζ are defined in Equation (4.36), Equation (4.35), and Equation (4.38), respectively.

For every combination of database $\mathbf{d} \in \mathcal{D}$ in Equation (6.20), we construct a set of unitary matrices, that is input to the BGS algorithm in Algorithm 6.3, namely

$$\{(\mathcal{U}_{\mathbf{d}}, \mathcal{O}_{\mathbf{d}}) \mid \mathbf{d} \in \mathcal{D}\} := \mathcal{I}, \quad (6.22)$$

where

$$\mathcal{U}_{\mathbf{d}} = \{\mathcal{A}, D, D(\psi)\}_{\mathbf{d}} \quad \text{and} \quad \mathcal{O}_{\mathbf{d}} = \{\mathcal{O}_x \mid x \in \mathbf{d}\} \cup \{\mathcal{O}_x(\varphi + \zeta) \mid x \in \mathbf{d}\}. \quad (6.23)$$

For each pair of sets in $\mathcal{I}$, we run **S1** of the BGS algorithm. We employ a **BFGS** solver of the **Python SciPy** library [168] in our program — the same solver that was used earlier.

We use the following objective functions in our optimization, namely

$$\mathbf{f} = \sum_{i,j \in \mathbf{d}, M_{ij} \neq 0} |M_{ij} - S_{ij}|^2 \quad \text{and} \quad f_{\mathcal{A}} = \sum_{i \in \mathbf{d}, M_{i0} \neq 0} |M_{i0} - S_{i0}|^2, \quad (6.24)$$

which are more relaxed than Equation (6.2), in that not all elements are subjected to the optimization;¹¹ we could have used this trick always to speed up the optimization. The matrix M is a unitary target, $M \in \mathcal{U}_{\mathbf{d}} \cup \mathcal{O}_{\mathbf{d}}$ and S is the total matrix resulting from the corresponding gate network. The indexing of M_{ij} , where $i, j \in \mathbf{d}$, indicates that we care only about the block diagonal part spanned by $\mathbf{d} \subseteq \Sigma^n$. Objective function $f_{\mathcal{A}}$ is used only for the optimization of $\mathcal{A}$, because $\mathcal{A}$

¹¹ The objective function f defined in Equation (6.2) subjects the optimizations to all elements in the matrix, whereas $\mathbf{f}$ and $f_{\mathcal{A}}$ care only non-zero matrix elements. Moreover, $f_{\mathcal{A}}$ care only about the first column of matrix $\mathcal{A}$.

(a) $N=5, \varphi + \zeta=0.1707, \psi=0.4510$						(b) $N=6, \varphi + \zeta=1.861, \psi=0.841$					
w	$\mathcal{A}$	$O(\pi)$	$D(\pi)$	$O(\varphi + \zeta)$	$D(\psi)$	w	$\mathcal{A}$	$O(\pi)$	$D(\pi)$	$O(\varphi + \zeta)$	$D(\psi)$
01234		0 1 2 3 4		0 1 2 3 4		012345		0 1 2 3 4 5		0 1 2 3 4 5	
CNOT	2	1 1 1 1 0	7	2 2 2 2 0	8	CNOT	1	2 2 1 1 1 1	4	4 4 2 2 2 2	6
01247		0 1 2 4 7		0 1 2 4 7		012347		0 1 2 3 4 7		0 1 2 3 4 7	
CNOT	2	0 1 1 1 1	7	0 2 2 2 2	8	CNOT	2	2 1 1 2 1 1	6	4 2 2 4 2 2	7
01256		0 1 2 5 6		0 1 2 5 6		012567		0 1 2 5 6 7		0 1 2 5 6 7	
CNOT	3	0 1 1 1 1	8	0 2 2 2 2	9	CNOT	3	1 1 1 2 1 1	8	2 2 2 2 2 2	8

(c) $N=7, \varphi + \zeta=2.0277, \psi=1.2056$					
w	$\mathcal{A}$	$O(\pi)$	$D(\pi)$	$O(\varphi + \zeta)$	$D(\psi)$
0123456		0 1 2 3 4 5 6		0 1 2 3 4 5 6	
CNOT	3	3 2 2 1 2 1 1	8	4 4 6 2 4 2 2	9

(d) $N=8, \varphi + \zeta=2.2143, \psi=1.5708$					
w	$\mathcal{A}$	$O(\pi)$	$D(\pi)$	$O(\varphi + \zeta)$	$D(\psi)$
01234567		0 1 2 3 4 5 6 7		0 1 2 3 4 5 6 7	
CNOT	0	6 6 6 6 6 6 6 6	6	6 6 6 6 6 6 6 6	6

Table 6.4: Minimal number of CNOT gates for all distinct combinations of $(\mathbf{d}, M, \tau)$ for all $\mathbf{d} \in \mathcal{D}$, $\tau \in \mathbf{d}$, $M \in \mathcal{W}_{\mathbf{d}} \cup \mathcal{O}_{\mathbf{d}}$, where $n = 3$ and $N \in \{5, 6, 7, 8\}$. The angles φ, ψ , and ζ refer to the Høyer parameters, defined in [Section 4.3.2](#).

acts on the zero state $|0\rangle^{\otimes 3}$, thus, only the first column matters; whereas $\mathbf{f}$ is used for the optimization of other unitary matrices. Both objective functions consider only the subspace of the Hilbert space spanned by the choice of $\mathbf{d}$. We tabulate the results of the step **S1** in [Table 6.4](#), which counts the minimal CNOT gates.

We define the optimization precision ε such that the success probability is approximately one: given δ , there exists an ε such that $p_s \geq 1 - \delta$, where p_s is the success probability of running [Algorithm 4.1](#) while replacing block M with the tested network. For the non-oracles, $M \notin \mathcal{O}_{\mathbf{d}}$, we take the worst p_s among all obtained p_s from all possible marked items $d \in \mathbf{d}$. Here, we set $\delta = 10^{-4}$, and obtain $\varepsilon_{\mathcal{A}} \leq 4.8 \times 10^{-11}$ for preparation blocks and $\varepsilon \leq 2.5 \times 10^{-8}$ for other blocks (this parameter also applies to step **S3.1**). The complete tables that show values of success probabilities p_s are shown in the Appendix of reference [\[143\]](#). While this does not complete our analysis of the three-qubit exact quantum search algorithms, these preliminary calculations indicate that the most efficient gate network will be achieved for $N = 6$ and $w = 012345$ (and not the smaller $N = 5$) — see the highlighted row in [Table 6.4](#). Thus, from this point, we seek a BOQC-compatible graph that solves the exact unstructured search with database $\mathbf{d} = 012345$.

Before going to step **S2**, let us go through **S3.1** — Oscar’s job — as this is the most interesting subtask. In this step, we redefine the objective function F (Equation (6.16)), into **ObjPOVM** that is defined in Subroutine 6.2, such that the optimization permits probability distribution of measurement outcomes instead of a definite outcome (the generic objective function). The **ObjPOVM** subroutine contributes to line 21 in the BGS algorithm. The subroutine allows the optimization to consider a more general measurement scheme, which implies providing a general means in storing the information in the database. Such an optimization is achieved by permitting several marked items x to stand for a single database entry; we do this by making the final measurement of the Grover-Høyer algorithm an incomplete or POVM measurement.¹² We refer to this scheme as “POVM measurement strategy”.

Subroutine 6.2 [143]**ObjPOVM**

Input: $\{\mathcal{N}_x(\vec{\Phi})\}, \mathfrak{M}, \mathfrak{d}, k, \mathcal{A}, D(\pi), D(\psi)$

Output: v

```

1:  $v \leftarrow 0$  ▷ initialize the objective value
2: for  $x$  in  $\mathfrak{d}$  do
3:    $S \leftarrow$  total matrix of  $\mathcal{N}_x(\vec{\Phi})$ 
4:    $|\Psi\rangle \leftarrow \mathcal{A}|0\rangle^{\otimes n}$  ▷ preparation, Equation (4.21)
5:   for  $i = 1$  to  $k$  do
6:      $|\Psi\rangle \leftarrow D(\pi)S|\Psi\rangle$  ▷ Grover iteration,  $D$  is defined in Equation (4.25)
7:   end for
8:    $|\Psi\rangle \leftarrow D(\psi)S|\Psi\rangle$  ▷ one Høyer’s iteration
9:    $v \leftarrow v + 1 - |\langle x | \mathfrak{M}(x) | \Psi \rangle|$  ▷ condition C1
10:   $v \leftarrow v + |f_{\square}|$  ▷ condition C2,  $f_{\square}$  is defined in Equation (6.25)
11: end for
    return  $v$ 

```

The objective function **ObjPOVM** evaluates the gate network candidate $\mathcal{N}$, given measurement angle guesses $\{\vec{\Phi}_x\}$ for each $x \in \mathfrak{d}$ — see **S3.1** in Algorithm 6.3 for details. The function is subject to two constraints: **C1** in line 9 and **C2** in line 10 of Subroutine 6.2. **C1** imposes a successful computation within the defined POVM measurement for all permitted marked items — notice that the loop goes for all $x \in \mathfrak{d}$. Constraint **C2** assures a block diagonal matrix; this is critical for the cases that occupy only a subset of the 2^n -dimensional Hilbert space, which is obtained by imposing the sum of the absolute values of the elements outside diagonal block to be zero:

$$f_{\square} = \sum_{i \in X, j \in X \setminus \mathfrak{d}, i \neq j} |S_{ij}|^2 + \sum_{i \in X \setminus \mathfrak{d}, j \in \mathfrak{d}} |S_{ij}|^2, \quad (6.25)$$

¹² The definition of POVM measurement can be found in Nielsen and Chuang’s book [133], page 90.

where $X := \{0, \dots, 2^n - 1\}$, and S is the matrix from line 3 in [Subroutine 6.2](#). Notice that, **ObjPOVM** sets an identical oracle in every query that is shown by applying the same S on each iteration (lines 6 and 8). The identical queries is made to reduce the complexity of the optimization, *i.e.*, to reduce the number of free parameters.

We use the POVM measurement strategy here, attempting to reduce the number of CNOT gates. We have found that this is effective to set the POVM measurement as

$$\mathfrak{M} := \{|0\rangle\langle 0| + |1\rangle\langle 1|, |2\rangle\langle 2|, |3\rangle\langle 3|, |4\rangle\langle 4|, |5\rangle\langle 5|\}. \quad (6.26)$$

The number of operators in $\mathfrak{M}$ indicates that there are only five items stored in $\mathfrak{d}$. As mentioned before, the entry of $\mathfrak{d}$ can be anything; *e.g.*, let $\mathfrak{d}$'s entries be A, B, C, D , and E , measurement $\mathfrak{M}$ can conclude the entries according to measurement outcomes as follows

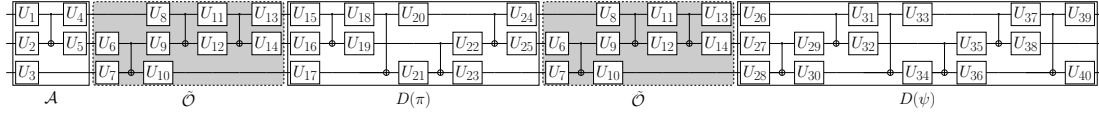
$$0 \mapsto A, 1 \mapsto A, 2 \mapsto B, 3 \mapsto C, 4 \mapsto D, 5 \mapsto E. \quad (6.27)$$

Thus, line 2 of **ObjPOVM** can be replaced with a loop over the items, *e.g.*, $\{1, 2, 3, 4, 5\}$, since measurement outcome 0 or 1 indicate the same item A and one can arbitrarily choose one of them.

Implementing all this, after steps **S1** and **S3.1**, we obtain [Figure 6.2](#).

From [Figure 6.2](#), we know that 17 CNOT gates are needed to run a three-qubit Grover-Høyer algorithm, where all oracles are identical, for every input and every query, denoted as $\tilde{\mathcal{O}}$. It is worth mentioning that the resulting matrix of $\tilde{\mathcal{O}}$ neither resembles the Høyer- nor Grover-oracle from our results.

We proceed to steps **S2** and **S3.2**, where the graph state for each matrix in $\mathcal{U}_{\mathfrak{d}} \cup \mathcal{O}_{\mathfrak{d}}$ is obtained. These steps have straightforward procedures when we know the topology (*e.g.*, [Figure 6.2](#)). For **S2**, the optimization implements the same objective functions as **S1** accordingly, *i.e.*, f for diffusion operators and $f_{\mathcal{A}}$ for the preparation operator. Also, **S3.2** employs the same objective function as **S3.1**, namely **ObjPOVM**. We set the exactness accuracy as $\delta = 10^{-4}$, namely probability of success $p_s \geq 1 - \delta$, which results in the numerical accuracy $\varepsilon \leq 2.1 \times 10^{-9}$ for each optimization here. From these steps, we obtain decompositions for each operator with gate set $\mathfrak{J}$ ([Equation \(6.5\)](#)) with the most economical results. Also, we perform a further optimization from each result after removing two consecutive nodes with measurement angles close zero. Finally, we obtain the final result from the BGS algorithm, shown in [Figure 6.3](#); the measurement angles are approximated to fractions of π , such that each angle can be stored in 10 bits. These angles — after encryption — are sent as 10-bit messages to Bob.



Gate	α	β	γ
U_1	2.1863	3.4700	3.4700
U_2	2.5159	3.3937	2.1618
U_3	5.4978	4.7124	1.5708
U_4	1.5708	1.8160	5.0408
U_5	5.6584	2.8913	5.3000
U_{15}	0.3929	1.9270	1.9237
U_{16}	0.7011	2.8271	1.0836

U_{17}	4.7124	2.7614	1.5708
U_{18}	5.0204	1.3905	5.2465
U_{19}	0.8541	2.7177	2.1142
U_{20}	2.8341	0.3039	3.7996
U_{21}	1.5708	4.2914	1.5708
U_{22}	5.5189	5.6770	4.3625

U_{23}	0.0000	3.1416	4.0531
U_{24}	1.9641	2.6566	0.4818
U_{25}	3.9010	1.9380	5.8626
U_{26}	4.3149	5.8000	2.8293
U_{27}	4.1749	1.0410	4.8345
U_{28}	3.3519	4.7124	0.0000

U_{29}	5.3865	0.4235	2.8814
U_{30}	2.5762	0.2296	4.8464
U_{31}	2.3073	5.0124	4.7439
U_{32}	1.5708	3.3484	3.1416
U_{33}	1.9640	4.5092	3.5709
U_{34}	0.9027	4.5151	4.5872

U_{35}	2.2493	2.3631	0.7364
U_{36}	3.1416	3.1416	5.0640
U_{37}	5.5892	4.5517	5.0515
U_{38}	1.0992	1.0522	2.5764
U_{39}	4.2518	5.6790	6.1262
U_{40}	1.4576	1.4573	4.4534

$\tau = 0$			
U_6	0.0000	5.2267	2.4690
U_7	0.0000	4.7124	2.3946
U_8	0.8805	5.0403	1.9146
U_9	2.9266	0.4472	2.0889
U_{10}	4.7124	4.6003	1.5708
U_{11}	5.2427	0.2804	1.5793
U_{12}	4.9284	3.4651	4.2091
U_{13}	2.5244	4.1733	4.6219
U_{14}	3.8309	4.1770	2.3886

$\tau = 2$			
U_6	1.5708	5.5013	2.2216
U_7	0.1287	3.2740	5.7062
U_8	4.1648	3.5317	5.6437
U_9	2.7877	5.7409	5.7010
U_{10}	6.1545	2.4033	5.1003
U_{11}	1.9797	5.6191	1.5769
U_{12}	4.9201	2.2995	2.3634
U_{13}	5.7356	4.4452	2.8675
U_{14}	1.7632	0.0188	3.8868

$\tau = 3$			
U_6	1.5708	0.7393	1.9242
U_7	5.3911	3.3728	2.8631
U_8	2.4257	0.0141	1.7221
U_9	0.3262	3.4474	4.6704
U_{10}	5.3911	1.5255	4.6197
U_{11}	5.6475	2.6774	1.5045
U_{12}	4.1456	4.2234	4.1602
U_{13}	0.6915	1.0830	2.5152
U_{14}	6.2190	0.9305	3.3521

$\tau = 4$			
U_6	0.0000	1.8189	4.4016
U_7	1.7606	5.7840	5.8093
U_8	2.3289	0.1070	3.0468
U_9	2.9211	1.0980	3.6533
U_{10}	6.0934	5.6116	3.4430
U_{11}	1.9619	1.6645	2.0202
U_{12}	3.6540	3.1061	0.6672
U_{13}	5.5116	5.2363	5.8918
U_{14}	4.7451	4.3359	0.4896

$\tau = 5$			
U_6	0.0000	2.7690	1.7284
U_7	5.2442	3.6695	5.6213
U_8	3.8175	6.0993	4.1414
U_9	0.3776	5.1356	5.8965
U_{10}	3.6734	2.9525	6.2280
U_{11}	2.6959	3.5218	0.4414
U_{12}	1.2797	5.6970	2.5852
U_{13}	5.5732	0.2160	0.9320
U_{14}	1.6530	0.2316	0.0389

Figure 6.2: The output of **S1** and **S3.1** in Algorithm 6.3, i.e., gates decomposition of Equation (6.21) with gate set $\mathfrak{G}$ (Equation (6.4)). The decomposition in **S1** employs objective functions $\mathfrak{f}$ and f_A (Equation (6.24)), whereas decomposition in **S3.1** employs **ObjPOVM** (Subroutine 6.2). Step **S3.1** results in an identical blind oracle $\tilde{O}$ as shown above. The Euler angles (in radians) are tabulated underneath, where the ones correspond to $A, D, D(\psi)$ are held by Alice and $\tilde{O}$ are held by Oscar. Depending on the database state, Oscar chooses the Euler angles accordingly. This result is identical to the one in [143].

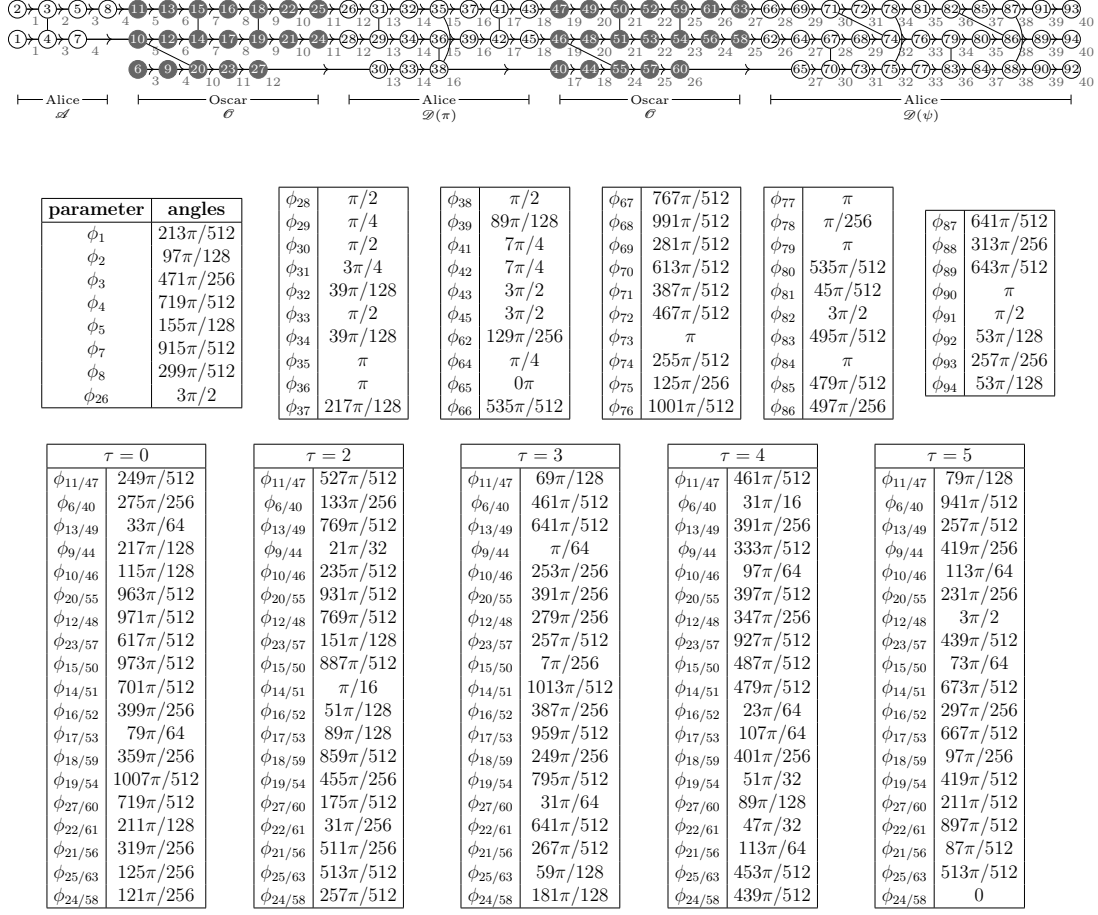


Figure 6.3: Three-qubit Grover-Høyer algorithm, for $\mathbf{d} = 012345$, and POVM $\mathfrak{M}$ (Equation (6.26)). It is a BOQC-compatible computation, comprising graph state $(\mathcal{G})$ and measurement angles (ϕ) obtained from the BGS algorithm (Algorithm 6.3). Oscar's measurement angles follow the chosen τ . We denote each operator with curly font accordingly, indicating a device-dependent representation, i.e., a one-way quantum computer — as shown in Equation (4.42). The gray nodes indicate blind oracles that are controlled by Oscar, and Alice controls the rest. Every integer in the measurement angles is expressible with 10 bits. Using the notation from Definition 5.1, $I = \{1, 2, 6\}$, $O = \{93, 94, 92\}$, $\tilde{I} = \tilde{O} = \emptyset$, flow $(f, \succ)$ is indicated with arrows (f) and the grey number $(\succ)$, total order $>$ is denoted by the node number, and integer $b \geq 10$. The (classical) input state is implicit, i.e., 000 , thus $\rho_A^{\text{in}} = |000\rangle\langle 000|_{216}$. This algorithm has success probability $p_s > 1 - 10^{-4}$.

Using the same principle as when solving the problem in [Example 6.4](#), our final graph state, in [Figure 6.3](#), has a flow that goes from left to right, for each node — unless the node is at the last layer.¹³ As the flow induces a partial ordering $\succ$, we construct a total ordering that is consistent with $\succ$. The total ordering, which corresponds to the measurement order, is represented as node number. There, all nodes are numbered, which means measuring all of them. This algorithm has classical input-output.

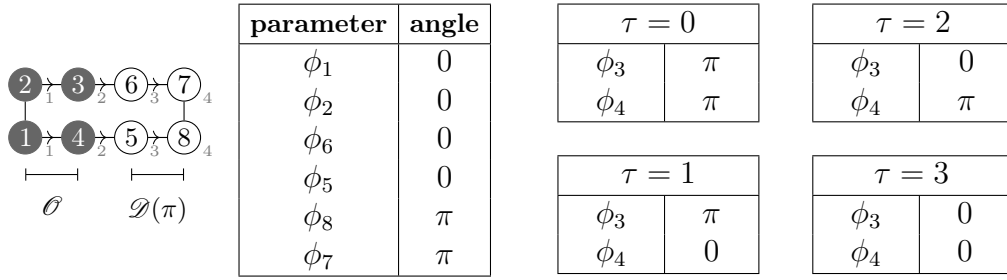


Figure 6.4: Two-qubit Grover algorithm comprising graph $\mathcal{G}$ and measurement angles ϕ with database $\mathbf{d} = 0123$ and POVM $\mathfrak{M} = \{|0\rangle\langle 0|, |1\rangle\langle 1|, |2\rangle\langle 2|, |3\rangle\langle 3|\}$; there are four entries in $\mathbf{d}$ and Oscar’s measurement angles follow the chosen τ . Curly font of the operators indicate device-specific implementation, i.e., a 1WQC algorithm. This algorithm is BOQC-compatible, where white nodes signify Alice’s computation and grey nodes denote Oscar’s oracle. Using the notation from [Definition 5.1](#), $I = \{1, 2\}, O = \{7, 8\}, \tilde{I} = \tilde{O} = \emptyset$, flow $(f, \succ)$ is indicated by arrows (f) and grey numbers ($\succ$), total order $>$ is denoted by node numbers, and integer $b > 2$. The (implicit) classical input state is 00 , thus $\rho_{\mathcal{A}}^{\text{in}} = |00\rangle\langle 00|_{12}$.

Now, for the case $n = 2$, we can run Grover’s algorithm ([Equation \(4.11\)](#)), which has one oracle call. Again, we use the BGS algorithm (only in principle, as this problem is solvable by hand) to produce a BOQC-compatible version, and we obtain the two-qubit Grover algorithm that is BOQC-compatible. Our result is shown in [Figure 6.4](#). In this case, the POVM measurement turns into the projective measurement.

Until this point, we have provided two BOQC-compatible algorithms in [Figure 6.3](#) and [Figure 6.4](#), which solve an unstructured search problem exactly (formulated in [Problem 4.4](#)). Also, we have gone through each step of finding a BOQC-compatible graph using the BGS algorithm. Moreover, we have successfully reduced the size of computation for a five-entry database from using 19 CNOT gates (see [Table 6.4](#)) to 17 CNOT gates (see [Figure 6.2](#)). From our results, we establish the unfortunate fact that the implementation complexity grows very rapidly for the Grover algorithm in the BOQC model. Going from a four-element database to a five-element database

¹³ Discussions about flow and ordering can be found in [Section 2.4](#).

for an exact quantum search algorithm within the BOQC scheme, means going from a 8-node to a 94-node cluster state.

6.5 BLIND SIMON'S ALGORITHM

Here, we provide a BOQC algorithm that solves Simon's problem defined in [Problem 4.5](#) for the two qubit case. Firstly, we denote $\Sigma := \{0, 1\}$. Simon's problem can be formalized as follows. Given access to an oracle that encodes a function $f : \Sigma^n \rightarrow \Sigma^n$, where f is promised to act as $f(x \oplus a) = f(x)$ for all query $x \in \Sigma^n$ and for a period $a \in \Sigma^n$, the task is to determine a . We construct the Simon oracle according to [Definition 6.3](#) (*useless Simon oracle*); the definition is adapted from [\[169\]](#). This construction admits the generalization of the oracle model of Harrow and Rosenbaum [\[43\]](#), namely oracle with internal randomness.¹⁴

Definition 6.3. *Useless Simon oracle. Given a period $a \in \Sigma^n$ and a function $f : \Sigma^n \rightarrow \Sigma^n$, where $f(x \oplus a) = f(x)$, $\forall x \in \Sigma^n$. The useless Simon oracle $\mathcal{O}_f$ that encodes f is obtained by the following construction.*

Let the binary representation of a be $a_{n-1} \dots a_1 a_0$ (least significant bit convention), and $B = \{i \in \{0, \dots, n-1\} \mid a_i = 1\}$ comprises locations of non-zero bits of a . Let $k = \min(B)$ be the first location of a non-zero a_i , and $\tilde{B} = B \setminus \{k\}$. The following operators construct a useless Simon oracle

$$\mathcal{O}_f = \bigotimes_{l=0}^{n-1} X_{n+l}^{r_l} H_{k+n} \prod_{i \in \tilde{B}} CNOT_{k,i} \bigotimes_{j=0, j \neq k}^{n-1} CNOT_{j,j+n} \prod_{i \in \tilde{B}} CNOT_{k,i}, \quad (6.28)$$

where $r_l \in \{0, 1\}$ is chosen at random in every query, $CNOT_{c,q}$ indicates a controlled- X_q gate with the control qubit c , and $2n$ qubits are employed. If a is trivial, i.e., $a = 0^n$, k is undefined, and we take $CNOT_{k,i} = I$ for all i .

Note that as first introduced in [Section 2.5](#), we distinguish notations $\bigotimes$ and $\prod$; the first signifies concurrent operations — every index appears once, whereas the latter indicates serial operations — there are some overlapping indices between operators. The internal randomness comes from random bit-flip operations $\bigotimes_{l=0}^{n-1} X_{n+l}^{r_l}$, where $r_l \in \{0, 1\}$ is chosen at random for every query. The bit-flip operations results in a different map f applied on each query; without the randomness, $f(x \oplus a) = f(x) = x$. The notion of *uselessness* comes from [\[53\]](#); queries that provide no information are called useless. The uselessness in [Definition 6.3](#) refer to the

¹⁴ We mentioned the oracle model with internal randomness in [Section 4.1](#).

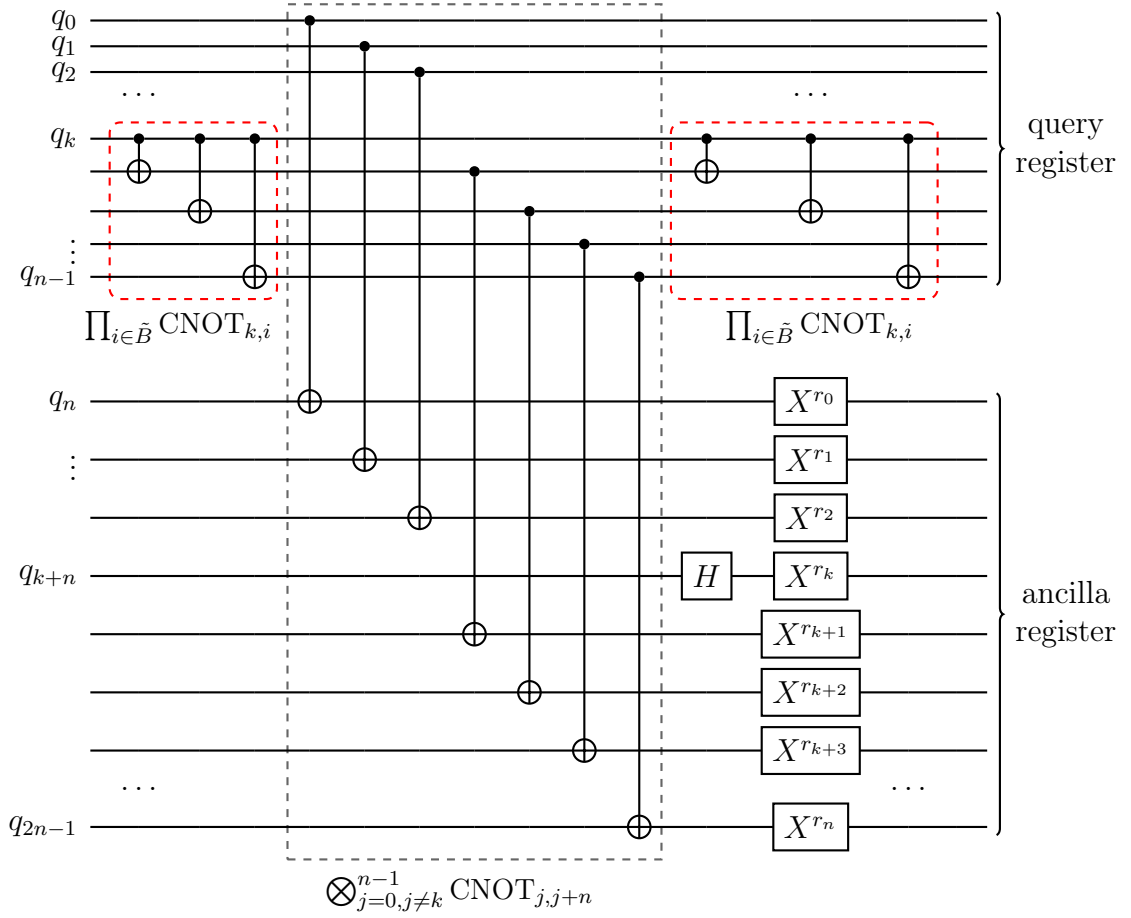


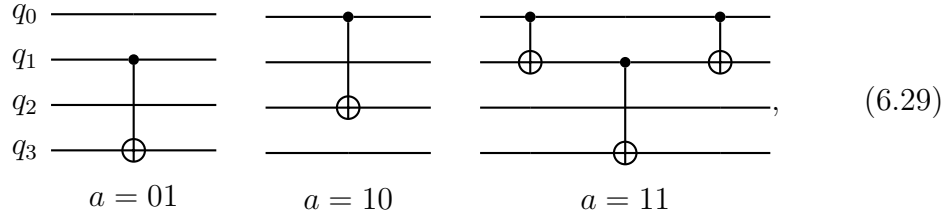
Figure 6.5: The circuit representation of a useless Simon oracle $\mathcal{O}_f$ defined in Definition 6.3. The first n qubits are called the query register and the last n qubits are called the ancilla register. Every notation corresponds to the one in Definition 6.3. If $a = 0^n$, the red boxes disappear, and there is no k (undefined), thus, the grey box contains n CNOT gates. The internal randomness is obtained from the random bit-flip X^{r_i} applied to the ancilla register, which captures choosing a function f at random among 2^n choices.

uselessness of classical queries. Simon's algorithm with useless Simon oracle provides strong separation, *i.e.*, *infinity-vs- $\Theta(n)$* , which means infinite classical queries are useless, while $\Theta(n)$ quantum queries solve the problem (not useless) [43]. Consider Figure 6.5 that shows a circuit construction of Definition 6.3.

We provide **Mathematica** code in Section B.5 that simulates Simon's algorithm with the oracle defined in Definition 6.3. There, we run some examples for $n = 2$ and $n = 3$ (only oracles), for all possible periods a . The probability distribution and the final states are shown.

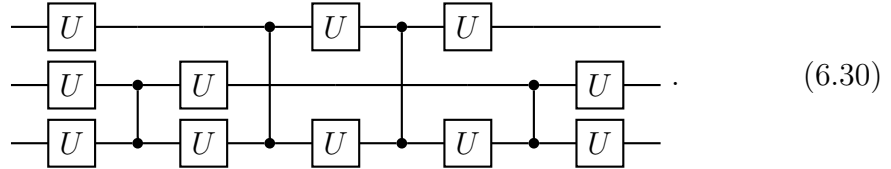
Note that, the Hadamard gate applied to the ancilla q_{k+n} is used to maintain the number of qubit for all $a \in \Sigma^n$. For $a = 0$, $2n$ qubits are needed, while for $a \neq 0$, $2n - 1$ qubits are sufficient. The constant number of qubits is useful to incorporate blindness into the oracle, when we consider all possible periods including the trivial case (one-to-one function). If we consider only non-trivial cases, the Hadamard operator H_{k+n} is omitted. For instance, Sachdeva [169] has demonstrated Simon's algorithm for $n = 2$ on the BOQC scheme, which provides blindness for non-trivial a (two-to-one functions).

Using oracle construction from Definition 6.3, for $n = 2$, with two-to-one f , there are three possible oracles

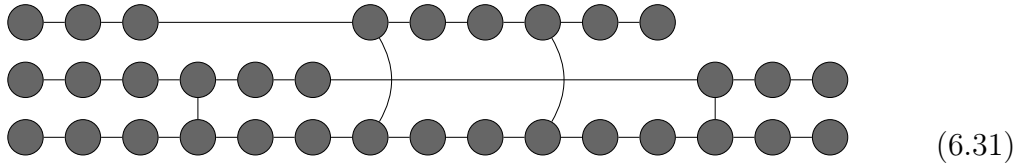


$a = 01$
 $a = 10$
 $a = 11$

where internal randomness is omitted and it requires three qubits in each case — one can remove the unused ancilla qubit. Sachdeva runs the BGS algorithm (Algorithm 6.3) to seek a BOQC-compatible graph that runs the Simon's algorithm with the oracles above (Equation (6.29)). From the results of steps **S1** and **S3.1** in Algorithm 6.3, the following gate network can realize all oracles in Equation (6.29).



Proceeding to **S3.2** in the BGS algorithm, the following graph can implement all oracles in Equation (6.29).

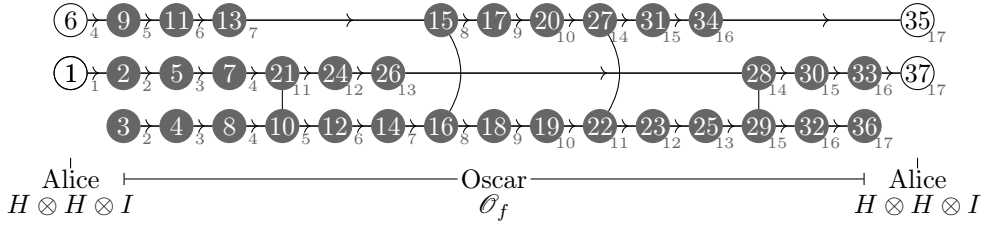


Notice that the matrices of circuits in Equation (6.28) are sparse; we consider only the non-zero elements. For all optimizations performed for this problem, we use objective function $\mathcal{F}$, which is the total of all objective values of $\mathbf{f}$ (Equation (6.24))

with full Hilbert space, *i.e.*, $\mathbf{d} = \{0, \dots, 2^n - 1\}$) for all non-trivial cases a , namely

$$\mathcal{F} := \sum_{a=1}^{2^n-1} \sum_{i,j=0, M_{ij}^a \neq 0}^{2^n-1} |M_{ij}^a - S_{ij}|^2, \quad (6.32)$$

where M^a is the target matrix, *e.g.*, the ones from Equation (6.29), and S is the resulting matrix of a network gate, *e.g.*, Equation (6.30).



node	$a = 01$	$a = 10$	$a = 11$	20	$1717\pi/1000$	$3\pi/2$	$3\pi/2$
2	$371\pi/250$	$6\pi/125$	$371\pi/250$	21	$1773\pi/1000$	$1153\pi/1000$	$1941\pi/1000$
3	$7\pi/8$	0	$7\pi/8$	21	$1851\pi/1000$	$883\pi/1000$	$1851\pi/1000$
4	0	$223\pi/500$	0	22	$437\pi/200$	$149\pi/200$	0
5	$1279\pi/1000$	π	$1279\pi/1000$	23	$\pi/2$	$303\pi/250$	$11\pi/8$
7	$51\pi/50$	$137\pi/250$	$51\pi/50$	24	$3\pi/2$	π	$3\pi/2$
8	$\pi/8$	π	$\pi/8$	25	$3\pi/2$	$223\pi/250$	$1071\pi/1000$
9	$\pi/2$	$669\pi/1000$	$\pi/2$	26	$\pi/2$	$383\pi/1000$	$\pi/2$
10	$727\pi/1000$	$31\pi/200$	$727\pi/1000$	27	$9\pi/8$	$126\pi/125$	$437\pi/200$
11	$139\pi/250$	$119\pi/200$	$139\pi/250$	28	$1071\pi/1000$	$379\pi/200$	$28\pi/125$
12	$3\pi/2$	$27\pi/50$	$3\pi/2$	29	$28\pi/125$	$1919\pi/1000$	$529\pi/1000$
13	π	$221\pi/200$	π	30	$529\pi/1000$	π	$28\pi/125$
14	$\pi/2$	$1933\pi/1000$	$1773\pi/1000$	31	0	$137\pi/100$	$\pi/2$
16	$1941\pi/1000$	$79\pi/100$	π	32	π	0	$529\pi/1000$
17	π	$\pi/2$	$3\pi/2$	33	$397\pi/100$	$79\pi/200$	$397\pi/200$
18	$3\pi/2$	$3\pi/2$	$1717\pi/1000$	34	$11\pi/8$	$1891\pi/1000$	$3\pi/2$
19	$3\pi/2$	$\pi/2$	$9\pi/8$	36	$913\pi/500$	$503\pi/1000$	$913\pi/500$

Figure 6.6: Two-qubit Simon's algorithm [169] with a different total ordering. The grey (Oscar) nodes indicate oracle. The curly font operators indicate a device-specific implementation, *i.e.*, a 1WQC algorithm. It is a BOQC-compatible algorithm, comprising graph $\mathcal{G}$ and measurement angles ϕ specified in the tables. The white (Alice) nodes have measurement angles $\phi_1 = \phi_{35} = \phi_{37} = 0$, and the grey nodes are Oscar's nodes. Oscar's measurement angles are chosen according to the period a . Using the notation from Definition 5.1, $I = \{1, 6, 3\}$, $O = \{35, 36, 37\}$, $\tilde{I} = I$, $\tilde{O} = \emptyset$, flow $(f, \succ)$ is indicated by arrows (f) and grey numbers ($\succ$), total order $>$ is denoted by node numbers, and integer $b \geq 11$. The (quantum) input state is explicitly initialized as $\rho_A^{in} = |000\rangle\langle 000|_{136}$.

Notice that the graph state in Equation (6.31) is implemented with the maximal number of nodes given the network circuit in Equation (6.30). At this point, obtaining a total graph state that runs Simon's algorithm is straightforward, namely by adding four Hadamard gates at the query register: two gates before and after the oracle. Recall that one Hadamard gate can be implemented with one measurement with measurement angle zero. Thus, we obtain the final graph state, as shown in Figure 6.6, together with measurement angles for all non-trivial a .

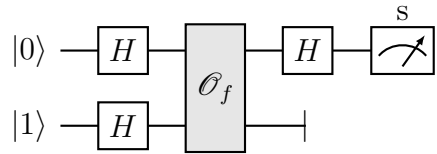
Figure 6.6 is the final result of the BGS algorithm, after running steps S2.2 and S3.2, where objective function $\mathcal{F}$ is also employed in the optimization of these steps. There, every measurement angle is representable with 11 bit strings.

If we also consider the trivial case $a = 00$, the following changes must be made. First, we need two ancillae instead of one; the graph states will have four layers instead of three. Second, the optimization must include this case; thus, the sum in the objective function F (Equation (6.32)) starts from $a = 0$ instead of $a = 1$.

6.6 BLIND DEUTSCH-JOZSA ALGORITHM

For some algorithms with a few qubits, sometimes we can construct a blind oracle by hand, *e.g.*, the 2-qubit Grover algorithm in Figure 6.4. Here, we construct a BOQC-compatible graph (by hand) that implements the two-qubit Deutsch-Jozsa algorithm (Section 4.2), *viz.*, Deutsch's algorithm [8].

Deutsch's algorithm is has the following circuit



where $f : \{0, 1\} \rightarrow \{0, 1\}$. The algorithm has an outcome $s = 0$ for a constant f and an outcome $s = 1$ for a balanced f . Thus, there are four possible functions, *e.g.*,

$$\begin{array}{cccc}
 \text{---} & \text{---} & \text{---} & \text{---} \\
 \text{---} & \boxed{X} & \text{---} & \boxed{X} \text{---} \\
 & & \text{---} & \text{---} \\
 (c_1) & (c_2) & (b_1) & (b_2)
 \end{array} \quad (6.33)$$

where c_1, c_2 result in constant functions and b_1, b_2 result in balanced functions; in particular, $f_{c_1}(0) = f_{c_1}(1) = 1$, $f_{c_2}(0) = f_{c_2}(1) = 0$, $f_{b_1}(0) = 1$, $f_{b_1}(1) = 0$, and $f_{b_2}(0) = 0$, $f_{b_2}(1) = 1$. We decompose every circuit in Equation (6.33) into the ones comprising gate set $\mathfrak{J}$ (Equation (6.5)) as follows

$$(c_1) \quad (c_2) \quad (b_1) \quad (b_2) \quad (6.34)$$

These circuits above are realizable with one brick of a brickwork (see Definition 3.1), namely

$$(6.35)$$

where the gray nodes indicate the blind oracle, and the square nodes signify quantum outputs. We embed the blind oracle graph into the algorithm in Section 6.6, and obtain a BOQC-compatible graph that realizes Deutsch's algorithm in Figure 6.7 below.

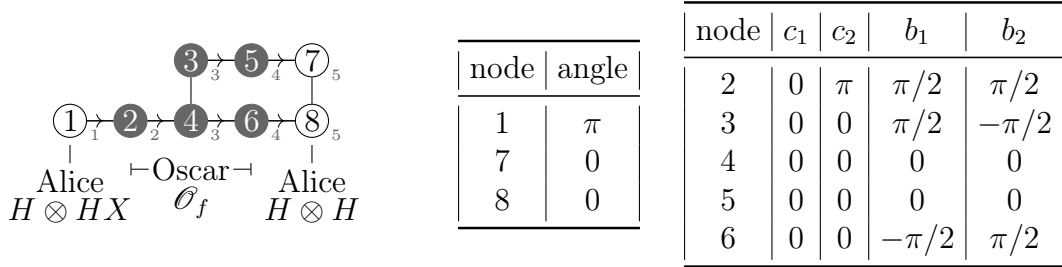


Figure 6.7: Deutsch's algorithm (Section 6.6) within the BOQC scheme; the graph ($\mathcal{G}$) is BOQC-compatible, and the tables present measurement angles (ϕ) for all nodes. The grey nodes indicate Oscar's oracle, whose measurement angles are selected accordingly by Oscar, depending on the desired function f , i.e., $f_{c_1}, f_{c_2}, f_{b_1}$, or f_{b_2} , defined in Equation (6.33). Whereas Alice's nodes, the white nodes, have fixed measurement angles. The measurement outcome of node 8 tells the result, i.e., outcome 0 for f_{c_1}, f_{c_2} and outcome 1 for f_{b_1}, f_{b_2} . Using the notation from Definition 5.1, $I = \{1, 3\}, O = \{7, 8\}, \tilde{I} = \tilde{O} = \emptyset$, flow ($f, \succ$) is indicated by arrows (f) and grey numbers ($\succ$), total order $>$ is denoted by the node number, and integer $b \geq 2$. The (classical) input state is zeros and implicit, thus $\rho_A^{in} = |00\rangle\langle 00|_{13}$.

NITROGEN-VACANCY CENTERS AS BOQC PLATFORM

The nitrogen-vacancy defect (NV) is a color-center¹ in diamond, whose spins are a promising platform for processing quantum information. The defect is consisting of a substitutional nitrogen atom and a neighbouring lattice vacancy, as shown in [Figure 7.1](#). The recent progress of NV-center experiments during our work has inspired us to study the system. For example, several successful conducted experiments related to our interest are: long coherence time spin [[170–173](#)], qubits entanglement [[49, 50, 174, 175](#)], quantum teleportation [[51](#)], and implementing Deutsch-Jozsa algorithm at room-temperature [[44](#)]. In our observation, NV-center systems (shortened as NV-centers) seem to be a potential platform for the BOQC scheme.² Thus, we come up with a concrete scheme on how to physically implement a BOQC algorithm using NV-center systems.³

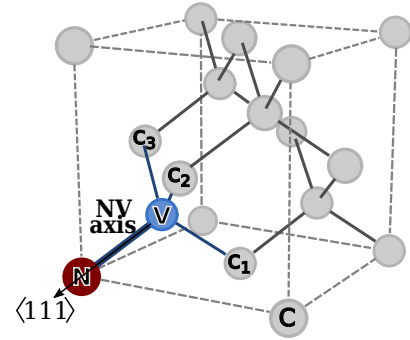


Figure 7.1: Nitrogen-vacancy center (NV-center) in diamond. The grey, red, and blue indicate carbon, nitrogen, and vacancy respectively.

In this section, we briefly discuss some critical features of NV-centers, followed by our proposal for implementation of the BOQC scheme on an NV-center platform, then demonstrate several BOQC algorithms (from [Chapter 6](#)) that could be run on this NV-center platform.

- ¹ Color center is a type of crystallographic defect, where an anionic vacancy in a crystal lattice is occupied by unpaired electrons. Such electrons tend to absorb light in the visible spectrum that results in colored materials that are normally transparent; therefore, it is called color center. For example, the NV-center that is used for quantum computing is pink.
- ² See [Chapter 5](#) for the discussion on the BOQC scheme.
- ³ The implementation of the BOQC scheme with NV-centers is introduced in [[143](#)].

7.1 THE NITROGEN-VACANCY CENTER IN DIAMOND

A point defect can introduce new energy levels in the energy band of an insulator or semiconductor host, which might radically change the magnetic or optical properties of the host material; this is the situation happening in the NV-center diamond. A pure diamond is an insulator with an ultra-wide band gap (~ 5.5 eV); the point defect introduced by nitrogen impurity introduces molecular orbitals within the host's band gap, which can locally create a two-level energy system. The electronic, optical, and dynamical properties of NV-centers has been studied over decades [48, 173, 176–180], for its interesting optical interactions, and its potential in quantum information processing and quantum sensing. In the following, we will touch on several essential properties of NV-center as a qubit and as a quantum network, which include electronic structure, optical interface as spin readout, qubits characterization, electron and nuclear spin control, and the *heralded entanglement* mechanism.

7.1.1 Characterization, initialization and control of electron and nuclear spins

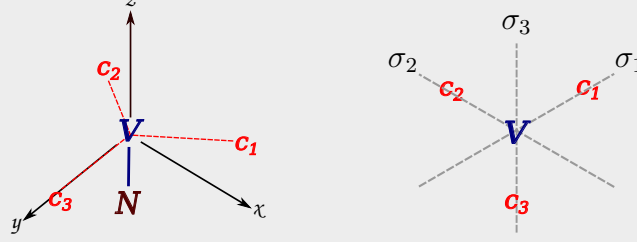
An NV-center host contains carbon and nitrogen atoms, in particular, two carbon isotopes: the majority spin-free ^{12}C and ^{13}C (1.1%) isotopes with spin $I_C = 1/2$, and each NV-center has one nitrogen ^{14}N isotope with nuclear spin $I_N = 1$, neglecting ^{15}N (0.4%) isotope with nuclear spin $I_N = 1/2$ that occurs by chance or engineering [181, 182]. The defect lies along an axis with Miller index $\langle 111 \rangle$ as shown in Figure 7.1, and it has point symmetry C_{3v} — see the extra box below. We use the negative charged defect (NV^- , spin $S = 1$) for quantum computing purposes, although it can have neutral charge (NV^0 , spin $S = 1/2$). The defect state is identified with Photoluminescence Spectroscopy via its optical zero phonon line (ZPL), *i.e.*, 1.945 eV (637 nm) for NV^- and 2.156 eV (575 nm) for NV^0 ; if the charge is neutral, a green laser (~ 532 nm) pumps an electron from the environment [183].



The C_{3v} point group, singlet, and triplet

The name of C_{3v} comes from Schönflies' notation that specifies point groups in three dimensions. The point group C_{3v} means that the configuration is

invariant under C_3 operations ($\frac{2\pi}{3}$ -rotations around z -axis) and has 3 vertical mirror planes $\sigma_1, \sigma_2, \sigma_3$.



Since there are six symmetry operations, C_{3v} is a group with order six. One may display all symmetry operations by its matrices, but it is inefficient. The compact (and standard) way is using the *character table*, *i.e.*, a multiplication table whose elements are *characters*. A character $\chi(M)$ signifies trace of operation M , where M is the matrix representation of the corresponding element; the character table of C_{3v} point group is the following.

C_{3v}	E	$2C_3$	$3\sigma_v$	III	IV
A_1	1	1	1	z	$x^2 + y^2, z^2$
A_2	1	1	-1	R_z	-
E	2	-1	0	$(x, y)(R_x, R_y)$	$(x^2 - y^2, xy)(xz, yz)$

The first row indicates point group operations, *e.g.*, $\{E, 2C_3, 3\sigma_v\} =: \mathcal{R}$; the first column gives Mulliken symbols for each irreducible representation, *e.g.*, $\{A_1, A_2, E\} =: \mathcal{C}$, where $\mathcal{R} \times \mathcal{C}$ forms the character table; columns III and IV indicate irreducible representation for which the functions can serve as a basis. Symbols A and E indicate 1- and 2- dimensional space, respectively. The square sum of characters of a class gives the group order, *e.g.*, representation E has order $2^2 + 2 \cdot (-1)^2 + 3 \cdot 0 = 6$. For example, an orbital having A_1 symmetry in C_{3v} means that it satisfies symmetry transformations of table C_{3v} at row A_1 .

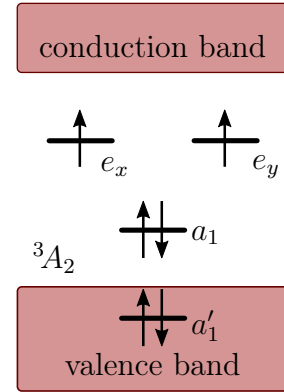
The subscripts 1 and 3 before a representation, *e.g.*, ${}^3A_2$, ${}^1A$, ${}^1E$, and ${}^3E$ signify degeneracy, particularly singlet (1) and triplet (3) states, which comes from combining two spin- $\frac{1}{2}$ systems accordingly. Each degenerated element is

indicated with magnetic number $m_s \in \{-S, -S + 1, \dots, S - 1, S\}$ for a spin S system. For instance, the singlet and triplet states are described below.

$$\text{singlet } \frac{|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle}{2}, m_s = 0, \text{ and triplet } \begin{cases} |\uparrow\uparrow\rangle, & m_s = 1 \\ (|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle)/\sqrt{2}, & m_s = 0 \\ |\downarrow\downarrow\rangle, & m_s = -1 \end{cases}.$$

Many properties of NV-centers, such as spin-orbital eigenstates, optical transitions, and vibronic transitions are predictable from the characteristics of C_{3v} [180, 184].

Six electrons sit at the point defect: three from dangling bonds of carbon atoms occupying the vacancy's neighborhood, two from the nitrogen, and one from the environment [179]; as they are well localized in the vacancy, its behavior is well explained in terms of molecular orbitals. Moreover, the energy levels a and e , which we use to define electron qubit later, lie deep inside the band gap, providing isolation from spin bath fluctuation from nuclear spins of carbon (isotope ^{13}C) and the nitrogen; this configuration results in a long coherence time up to seconds [172, 173].



The six electrons occupying the point defect have triplet ground state ${}^3A_2$ with occupation $a_1^2 e^2$ (as shown in Figure 7.2). Triplet excited state ${}^3E$ with occupation $a_1^1 e^3$, is occupied upon optical excitation from the a_1 orbital to the e_x or e_y orbital, where the superscript indicates the occupation number; we omit orbital a'_1 since it is sitting inside the valence band. We will see later that this optical excitation provides a means to achieve spin initialization and readout.

Figure 7.2: Electronic configuration $a_1^2 e^2$.

The triplet ground state energy ${}^3A_2$ and triplet excited state ${}^3E$ are separated by 1.945 eV (~ 637 nm), which is the ZPL, whereas the singlet ground state ${}^1E$ and singlet excited state ${}^1A$ are separated by 1.19 eV (~ 1.04 μm). The triplet ground state ${}^3A_2$ has one energy level for $m_s = 0$ and doubly degenerate states for $m_s = \pm 1$, whereas ${}^3E$ has 4 energy levels: $(E_{1,2}, m_s = \pm 1)$, $(E_{x,y}, m_s = 0)$, $(A_1, m_s = \pm 1)$, and $(A_2, m_s = \pm 1)$, where the notations are following irreducible representation of C_{3v} — see [184] for further details.

The point defect symmetry also allows prediction on the photo-dynamics of an NV-center, as shown in Figure 7.3; these dynamics are responsible for determining

the parameters for spin initialization, readout, and estimating photon emission rates.

The energy splitting from Figure 7.3 can be adjusted by applying external magnetic and electric fields. An external magnetic field applied along the NV-axis induces Zeeman splitting for $m_s = -1$ and $m_s = 1$ states at ${}^3A_2$. In addition to the magnetic field, energy levels at ${}^3E$ are also adjusted by manipulating lateral strain via an electric field perpendicular to NV-axis. A qubit is defined at triplet ground state ${}^3A_2$, where $m_s = 0$ encodes $|0\rangle$ and $m_s = -1$ encodes $|1\rangle$.

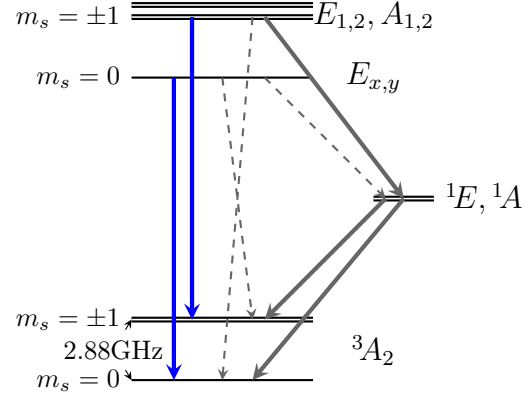


Figure 7.3: Spin dynamics of an NV-center, adapted from [185]; the scaling of energy levels does not represent real splittings. There are two types of transitions: spin-preserving (blue lines), non-spin preserving, and spin-dependent inter-system crossing (ISC) via singlet excitation. Excitations represented with blue lines results in immediate emission with a longer wavelength than the incoming wave — known as fluorescence, whereas the decay via ISC does not result in emission. The dashed transitions violate selection rule, which happens in the experiment due to some mixing of $E_{1,2}$ and $E_{x,y}$ [178]; thus, they are weak and happen with low probability at cryogenic temperature. These differences help to investigate the energy structure through experiments.

The electron spin (electrons in NV-center) initialization and readout is done optically by lasers; two laser frequencies for readout (f_{RO}) and initialization (f_{INIT}) are determined experimentally to obtain high fidelity outcomes, *i.e.*, by examining Photoluminescence spectrum of the NV-center — see Figure 7.4 for an illustration. Once the appropriate frequencies are found, the optical excitation with frequency f_{RO} is performed to measure the spin, while laser with frequency f_{INIT} is used to initialize the spin to $m_s = 0$ ($|0\rangle$). As shown in Figure 7.4, frequency f_{RO} is set such that it is resonant with one of $m_s = 0$ excitations, whereas frequency f_{INIT} is resonant with excitation of $m_s = \pm 1$ to one of four levels; for example, Bernien’s experiment in [185] sets $m_s = 0 \leftrightarrow E_x$ resonant readout and $m_s = \pm 1 \leftrightarrow A_1$ resonant initialization.

To initialize an electron qubit into state $|0\rangle$, *viz.*, $m_s = 0$, an INIT-laser is turned on. If the system is initially at $m_s = \pm 1$, it will repeatedly be excited until eventually the state transitions to $m_s = 0$ with high probability (see Figure 7.3 for the transition paths), if the system starts with $m_s = 0$, the system remains in this state; this method also works for superposition states. In principle, since f_{INIT}

does not resonate with $m_s = 0$, the INIT-laser is simply applied until the system goes dark [186]. This initialization provides a high fidelity state, *e.g.*, with lower bound $(99.7 \pm 0.1)\%$ [186].

The electron spin readout is done by collecting photons emitted into the phonon sideband, while a laser with frequency f_{RO} periodically transitions $m_s = 0 \leftrightarrow E_{x,y}$ — see Figure 7.4. If some photon(s) are detected, the spin was in the bright $m_s = 0$ state since f_{RO} is in resonance with $m_s = 0$; if no photon is detected (dark), the state was in $m_s = \pm 1$ [186]. This readout fidelity for state $m_s = 0$ is constrained by the finite collection efficiency and the spin-flip probability per optical cycle $< 1\%$ [186].

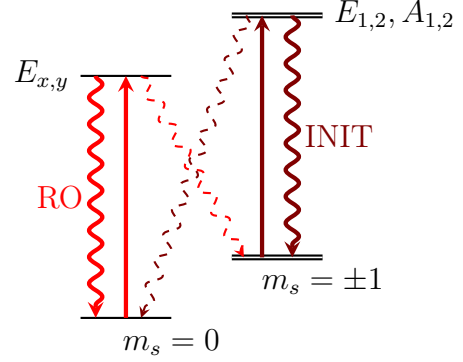


Figure 7.4: This figure is adapted from [186], which corresponds to readout (RO) excitation and initialization (INIT); The solid lines represent resonant excitations, whereas the dashed lines are weak transitions or ISC (see Figure 7.3). Frequency f_{RO} is seek to parameterize readout through process RO, while frequency f_{INIT} is seek to parameterize spin initialization through process INIT.

Having the qubits defined at ground state $^3A_2$: $m_s = 0$ encodes $|0\rangle$ and $m_s = -1$ encodes $|1\rangle$, now we want to coherently control⁴ the spin to obtain an arbitrary superposition; this is done using electron spin resonance techniques. An AC current at a frequency resonant with the energy splitting between $m_s = 0$ and $m_s = -1$ that results in coherent oscillations between the two energy levels [183]. For instance [183, 185, 186] achieve Rabi frequencies of (7.67 ± 0.02) MHz. Thus, an arbitrary single-qubit rotation can be driven by a particular width and amplitude of the applied AC pulse.

Now, since the nitrogen host also has a nuclear spin ($I_N = 1$), it introduces hyperfine splitting. Transitions $m_s = 0 \leftrightarrow m_s = -1$ depend on the nuclear spin state of the nitrogen — see Figure 7.5. Thus, electron spin operations conditional to the nitrogen spin state can be done by driving the electron spin sufficiently weakly such that only one of the three lines (see Figure 7.5 at $|1\rangle_e$) is addressed, or alternatively by driving strongly enough such that it rotates all the three lines.

The weak coupling between the NV-center and nuclear spins ^{13}C forms a quasi-static fluctuating spin bath, which is the primary source of electron spin decoherence. Such fluctuations can be significantly reduced by dynamical decoupling techniques,

⁴ Coherent spin control here means that the state of spin oscillates between two energy levels in the presence of a driving field. One can imagine it as a qubit continuously precessing in a rotating frame; this is known as Rabi oscillation. The precession frequency is called Larmor frequency.

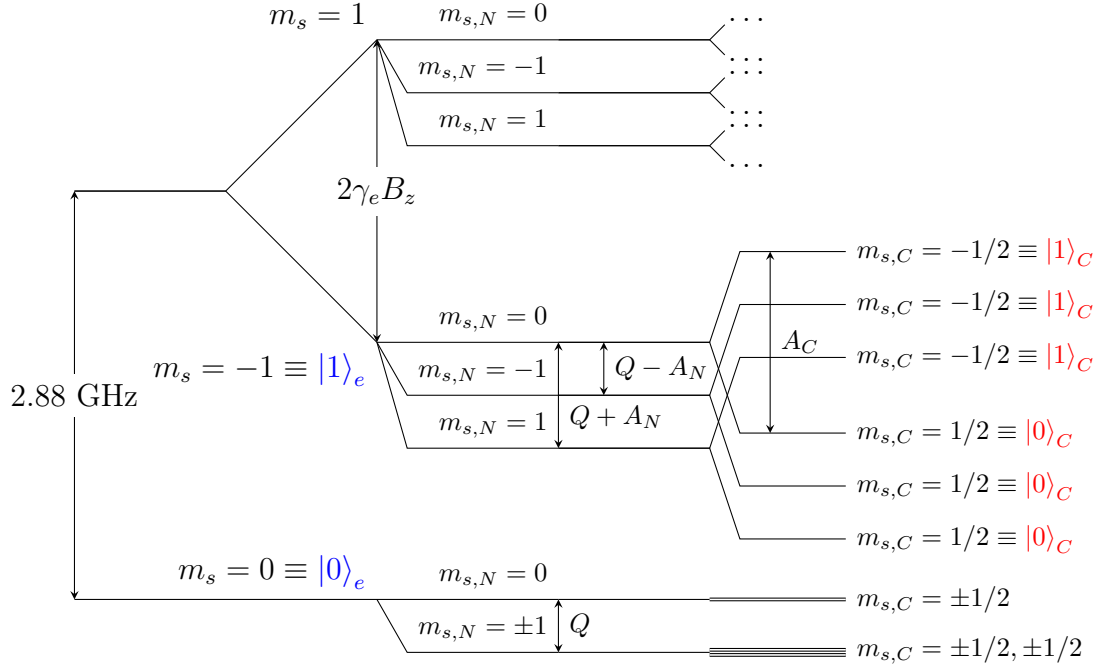


Figure 7.5: Energy levels of the ground state ${}^3A_2$, adapted from [185], with hyperfine structure from the nuclear spin of nitrogen N with spin $I_N = 1$ and a nuclear spin of carbon ${}^{13}C$ with spin $I_C = 1/2$. The energy levels are sketched according to the eigenvalues of Hamiltonians with secular approximation (see [185] for further details). Magnetic spin numbers for electron, nitrogen, and carbon ${}^{13}C$ are indicated with m_s , $m_{s,N}$, and $m_{s,C}$, respectively. Notice that, since $I_N = 1$, the energy is lowered due to quadrupole splitting at $Q = 2\pi \times 4.946$ MHz. The nitrogen hyperfine coupling constant is $A_N = 2\pi \times 2.186$ MHz, whereas the hyperfine coupling constant for carbon A_C depends on its distance to the electric spin. A magnetic field B_z is applied along the z -axis, where $\gamma_e = 2.802$ MHz/G is the gyromagnetic ratio. Here, the electron qubit is indicated by $|0\rangle_e$ and $|1\rangle_e$, and the nuclear qubit is indicated by $|0\rangle_C$ and $|1\rangle_C$. The constants in this figure are taken from [185].

as suggested in [187]. The technique allows a very long coherence time, *e.g.*, an experiment in [173] has successfully reached a coherence time of one second. It turned out that the weakly coupled nuclear spins ${}^{13}C$ can be used as qubits, where the decoupling technique becomes the main principle of characterization, initialization, and control of them. Taminiau et al. [45] proposed techniques to individually control and initialize weakly coupled nuclear spins ${}^{13}C$ based on the dynamical decoupling technique. Moreover, Cramer [183] has successfully implemented their techniques to demonstrate a three-qubit repetition code error correcting scheme using five qubits: one electron spin and four nuclear spins.

Note that here we only consider weakly coupled nuclear spins ^{13}C as nuclear qubits. Additionally, in the process of characterizing an NV-center, the best NV-center is chosen for its fluorescence brightness and the absence of strong coupling to any nuclear spin ^{13}C in the electron spin resonance (ESR) spectrum and the Ramsey measurements [45]. Thus, the term “nuclear spin” here always means a weakly coupled nuclear spin ^{13}C . The following is a brief discussion on Taminiau et al.’s techniques [45], which is adapted from [45, 183, 186].

The key to isolating a particular nuclear spin from other nuclear spins is its distinct conditional precession due to hyperfine interaction with the electron spin.⁵ When the electron spin state is $m_s = 0$, all nuclear spins precess around an axis parallel to the applied magnetic field B_z , with a Larmor frequency $\vec{\omega}_L$. For $m_s = -1$, each nuclear spin precesses uniquely with Larmor frequency $\vec{\omega}_L + \vec{A}_C$, where $\vec{A}_C$ results from the hyperfine interaction between nuclear and NV-center spin whose strength depends on the position of the nuclear spin relative to the NV-center.⁶

The state-dependent electron-nuclear spin interaction is probed by preparing the electron spin in $|+\rangle_e := (|0\rangle_e + |1\rangle_e)/\sqrt{2}$,⁷ and applying a dynamical decoupling sequence of π -pulses

$$\tau - \pi - 2\tau - \pi - \tau =: s_\tau \quad (7.1)$$

for N times, where τ is the spin relaxation time.⁸ The unit sequence s_τ results in one of the rotational operations V_i :

$$V_0 = \exp[-i\phi(S \cdot \vec{n}_0)] \text{ or } V_1 = \exp[-i\phi(S \cdot \vec{n}_1)], \quad (7.2)$$

which rotates the nuclear spin around an axis $\vec{n}_i$ with angles ϕ ; the index i corresponds to the initial electron state $|i\rangle_e$, where $i \in \{0, 1\}$; therefore, the dot product is done over the electron spin operator S . The rotation angle ϕ and rotation axes $\vec{n}_i$ are also conditioned on the electron spin, *i.e.*, $\phi = \text{tr}(V_i)$, where the axes are dependent on the decoupling time τ , and τ is dependent on electron spin and the individual hyperfine coupling constant A_C . The individual coupling A_C can be inferred by sweeping on all possible τ for a fixed number of pulses N ⁹ applied to the electron in state $|+\rangle_e$. The individual nuclear spin is identified from the sharp dips of the resonance imaging output that is also matched to a spin model — we refer to [45] for the model.

⁵ One can quickly observe this from Figure 7.5.

⁶ Consider Figure 7.5 to illustrate a hyperfine splitting from a nuclear spin ^{13}C . To see details in deriving spin Hamiltonian for the nuclear spin, we refer to [183].

⁷ See Figure 7.5 for the spin encoding

⁸ τ is also known as decoupling time or free evolution time in literatures. The unit sequence s_τ is commonly practiced in a magnetic resonance, taking an advantage of *spin echo*; we refer to [188] for further details.

⁹ In [45, 183], $N = 32$ or $N = 96$ for higher sensitivity.

Applying a pulse s_{τ_k} , with a particular τ_k that corresponds to nuclear spin k will perform the conditional rotation $V_{0,k}$ or $V_{1,k}$, where at the same time the electron spin decouples from other nuclear spins. The desired quantum operator on nuclear spin k is determined by choosing the appropriate τ_k and N ; we refer to [45, 183] for the derivation and further details.

In practice [183], the initialization and control of nuclear spins are done via the electron spin, and swapped to the corresponding nuclear spin. Note that the evolution of nuclear spins is insensitive to optical excitation of the electron spin, as long as the electron spin state is known throughout the full experiment; this preserves the nuclear spin state under the readout process of the electron spin [183].

7.1.2 Quantum network with NV-center nodes

High fidelity entanglement of two quantum nodes is the main resource of a quantum network. Here, we discuss Barrett and Kok's scheme in [189], providing an efficient and robust heralded entanglement scheme with arbitrary distance between the nodes.¹⁰ Their scheme has been successfully implemented within various settings [50, 52, 175, 190] with node separation up to 1.3 kilometers [52].

The physical requirements of Barrett and Kok scheme is the following. Ideally, there are two low-lying states $|\uparrow\rangle, |\downarrow\rangle$, and an excited state $|e\rangle$, such that an optical π -pulse performs a selective transition, *e.g.*, $|\uparrow\rangle \mapsto |e\rangle$ and $|\downarrow\rangle \mapsto |e\rangle$; transition $|\downarrow\rangle \mapsto |e\rangle$ is forbidden. The states $|\uparrow\rangle$ and $|\downarrow\rangle$ represent qubits with state $|0\rangle$ and $|1\rangle$, respectively, and some quantum operations can be carried out on those qubits. While we are concentrating on the NV system here, several other physical systems fulfill the requirements: quantum dots with an extra electron and trapped ions [189]. The setup is sketched in Figure 7.6, where the scheme is formulated in Protocol 7.1.

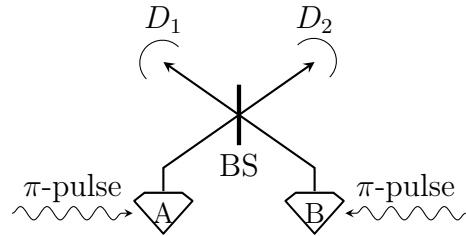


Figure 7.6: The setup for Barrett and Kok's heralded scheme, entangling node A and node B. A π -pulse selectively excites the electron spin in the node, D_1 and D_2 are photon detectors, and BS denotes a 50:50 beam-splitter.

To understand the mechanism behind Protocol 7.1, let us follow the state evolution for every step therein. First, we adopt this scheme into the NV-center setting.

¹⁰ Heralded means that there is a clear sign whether the entanglement was successful.

Protocol 7.1 Double heralded entanglement scheme [189]

The setup follows Figure 7.6.

- 1: Prepare the electron spin of each node in the equal superposition state $|+\rangle$.

First round

- 2: Apply the spin-selective optical excitation π -pulse to each node.
- 3: Wait for up to a time t_{wait} for a photon detection event in either D_1 or D_2 .
- 4: Wait for t_{relax} , for any remaining excitation.
- 5: Apply X gate to the electron spin of each node.

Second round

- 6: Repeat steps 2-3-4-5.

To check:

▷ A “click” means detecting one photon.

If no click is observed in a round or if a detector clicks twice, then the scheme has failed, go to step 1. If in every round only one detector clicks, then the entanglement is established on both nodes with unit fidelity — assuming an ideal process.

Here, $|\uparrow\rangle$ and $|\downarrow\rangle$ are ${}^3A_2$ state with $m_s = 0$ and $m_s = -1$, respectively. The excited state $|e\rangle$ is encoded by the triplet excited state $E_{x,y}$, where the optical excitation can be performed using the π -pulse with frequency f_{RO} that is resonant on the transition $m_s = 0 \leftrightarrow E_{x,y}$ — see Figure 7.4. Notice that the same frequency f_{RO} is also used for readout. The spin transition is coherently performed with a π -pulse at MW frequency; thus, an equal superposition of quantum states is obtained at $\frac{\pi}{2}$ -pulse, namely the half pulse width of the resonance MW. As the resonant excitation emits photons, we denote $|0\rangle$ and $|1\rangle$ to indicate vacuum and single-photon Fock states, respectively.

By referring to Protocol 7.1, the first and second steps are described as

$$\frac{(|\uparrow\rangle_A + |\downarrow\rangle_A)}{\sqrt{2}} \frac{(|\uparrow\rangle_B + |\downarrow\rangle_B)}{\sqrt{2}} \xrightarrow{\text{line 2}} \frac{(|\uparrow 1\rangle_A + |\downarrow 0\rangle_A)}{\sqrt{2}} \frac{(|\uparrow 1\rangle_B + |\downarrow 0\rangle_B)}{\sqrt{2}}. \quad (7.3)$$

At this point, the electron spin on each node is entangled to the emitted photon. Let the emitted photon from both NV-centers be indistinguishable, after the emitted photons have passed the beam-splitter, a click in one detector (with probability 1/2) corresponds to photon state $|1_A 0_B\rangle \pm e^{-i\phi} |0_A 1_B\rangle$, where ϕ is a phase shift depending on optical path length, while the spin state becomes

$$\frac{|\uparrow\downarrow\rangle_{AB} \pm e^{-i\phi} |\downarrow\uparrow\rangle_{AB}}{\sqrt{2}}. \quad (7.4)$$

As some error is likely to happen, such as photon loss or imperfect photon detection, the second round is done to detect an error.¹¹ If one detector clicks in the first round, it is either the good case (Equation (7.4)), or the bad case $|\uparrow\uparrow\rangle_{AB}$ with an error in one of photon states. Thus, an application X gate in line 5 results in either

$$\frac{|\uparrow\downarrow\rangle_{AB} \pm e^{i\varphi} |\downarrow\uparrow\rangle_{AB}}{\sqrt{2}} \text{ or } |\downarrow\downarrow\rangle. \quad (7.5)$$

For the good case, the second round does not change the detection, one detector will click if there is no detection error, otherwise, no detector clicks. For the error case, the optical excitation will not happen; thus, no photon is detected. Therefore, two rounds (or double) heralded entanglement adds a fault-tolerant entanglement scheme. Moreover, the second round eliminates the phase φ , which makes this protocol distance independent.

In the successful event, *i.e.*, when a detector clicks once in each round, the final state is either

$$\frac{|\uparrow\downarrow\rangle_{AB} + |\downarrow\uparrow\rangle_{AB}}{\sqrt{2}} \text{ or } \frac{|\uparrow\downarrow\rangle_{AB} - |\downarrow\uparrow\rangle_{AB}}{\sqrt{2}}, \quad (7.6)$$

where the first happens if the same detector clicks in both rounds, and the latter happens when a different detector clicks in each round.

7.2 BOQC ALGORITHMS ON NV-CENTER PLATFORM

Here, we discuss our physical implementation scheme [143] using NV-centers as platform for BOQC. We explicitly show how several BOQC-compatible computations can be implemented on the platform: 3-qubit and 2-qubit exact search algorithms, Simon's algorithm, and Deutsch's algorithm. As most of quantum algorithms require classical inputs and outputs, we only consider classical inputs and outputs cases. Also, we employ the lazy 1WQC scheme in Section 2.5 to reduce the number of physical qubits. Therefore, our proposal on the NV-center platform is designed for the BOQCo scheme with classical inputs and outputs, which is defined in Protocol 5.5.

To build a platform that runs BOQC means achieving the required quantum power (discussed in Section 5.3, tabulated in Table 5.1) for the clients and the server. When inputs and outputs are classical, this implies the following requirements:

¹¹ Here we consider only measurement error that comes from photon loss.

- (client) create a class of states $\{|+\theta\rangle\}, \theta \in [0, 2\pi)$, and transmit it to Bob.
- (server) receive the states from clients, perform CPHASE gates, and measure qubits in the $|\pm\theta\rangle$ bases, where $\theta \in [0, 2\pi)$.

Previous NV-center experiments [47, 50, 52] have demonstrated single- and two-qubit operations on electron and nuclear spins. To accomplish reliable transmission, we propose using remote state preparation (RSP) [163] as a quantum channel. The RSP is understood to be very efficient for the family of states $\{|+\theta\rangle\}$ for arbitrary θ [191]. Moreover, transmitting qubits via RSP does not require any classical feedback. For BOQC computations with classical inputs and outputs, the setting of our NV-center platform is described in Figure 7.7.

The computation is realized by N rounds of partial computations, where each round i corresponds to the corresponding computation subgraph at time step i in the BOQCo scheme, where N is the number of measured nodes; for classical inputs and outputs, N is the total nodes number. The notion of partial computation here corresponds to “BOQC by parts” from Protocol 5.5, *i.e.*, evaluation of an index i inside the loop starting from line 3.

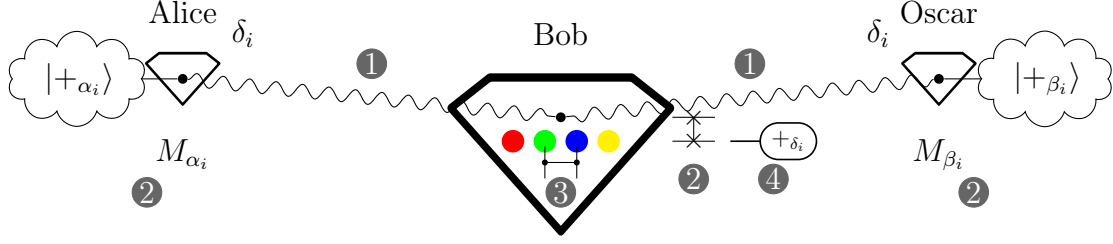
Transmitting the encrypted qubit via RSP introduces additional randomness that comes from a client’s measurement outcome, whose correction is incorporated into the measurement angle. For instance, given that Alice wants to transmit $|+\alpha_i\rangle$ to Bob, and they know that entanglement is established — with different clicks on each round in the process of step ①; using Equation (7.6), the electron spin state of Alice (A) and Bob (B) is given by

$$\frac{|01\rangle_{AB} - |10\rangle_{AB}}{\sqrt{2}} = \frac{|+\alpha_i - \alpha_i\rangle_{AB} - |-\alpha_i + \alpha_i\rangle_{AB}}{\sqrt{2}}, \quad (7.7)$$

where $|\uparrow\rangle \equiv |0\rangle$ and $|\downarrow\rangle \equiv |1\rangle$. At step ②, Alice measures with measurement projectors M_{α_i} ; thus, Bob’s quantum state is $|+\alpha_i + \pi s_i\rangle$, where $s_i = 0$ and $s_i = 1$ correspond to outcome $|+\alpha_i\rangle$ and $|-\alpha_i\rangle$, respectively. The state is unknown to Bob. Since Alice is the only person knowing s_i , the protocol remains secure, and the computation correctness is maintained by adding the correction, *i.e.*, redefining $\delta_i := \phi'_i + \pi r_i + (\alpha_i + \pi s_i)$ at line 17 in Protocol 5.5.

To estimate the execution time of a BOQC computation on our NV-center platform, based on state-of-the-art technologies [47, 175, 190], we assume execution times of each step in Figure 7.7 as

$$t_{\textcircled{1}} \approx 25 \text{ ms}, \quad t_{\textcircled{2}} \approx 1.5 \text{ } \mu\text{s}, \quad t_{\textcircled{3}} \approx 3.5 \text{ ms}, \quad t_{\textcircled{4}} \approx 0.5 \text{ ms}, \quad (7.8)$$



Step	Party	Process
①	client, server	Double heralded entanglements (Protocol 7.1) with different detector clicks on each round.
②	client	The electron spin measurement with projector M_{α_i} if j is Alice's node or M_{β_i} if j is Oscar's node.
	server	Operator $\text{SWAP}(e, n_k)$, accomplishing an RSP on n_k , where e is the electron qubit and n_k is a free nuclear spin.
③	server	CPHASE operations on nuclear spins accordingly, where $\text{CPHASE}(n_1, n_2)$ is implemented with a sequence of operations: $\text{SWAP}(n_1, e)$ - $\text{CPHASE}(e, n_2)$ - $\text{SWAP}(n_1, e)$.
④	server	Measure the corresponding nuclear spin k with projectors M_{δ_i} , where δ_i is instructed by Alice or Oscar.

Figure 7.7: The setting of an NV-center platform for the BOQC scheme. Alice, Oscar, and Bob are spatially separated, where each of them has an NV-center. The black dot in the diamond indicates electron spin qubit, defined as two triplet ground states $m_s = 0$ and $m_s = 1$, whereas the colorful nodes in Bob's diamond represent nuclear spins ^{13}C , see Figure 7.5 for the spin encoding. Variables α_i and β_i are randomly generated by Alice and Oscar, respectively, and kept only for themselves, and measurement angle δ_i is calculated either by Alice or Oscar, depending on i . Projectors M_{α_i} and M_{β_i} indicate projective measurement operators, where $M_{\theta} := \{|+\theta\rangle\langle+\theta|, |-\theta\rangle\langle-\theta|\}$. All variables here correspond to the ones in Protocol 5.4. A BOQC computation is executed by running several rounds of steps ①–④ (shown in the table), where the number indicates the time step. Note that multiple transmissions may happen in one round, but Bob's measurement happens only once.

where we assume every two-bit operation requires 0.5 ms and all the operations on the nuclear spins are performed via the electron spin, *i.e.*, CPHASE gates and measurements. We use the data to estimate the execution time of the algorithms presented here.

From the results shown in Figures 7.8 to 7.11, the longest required coherence time of a nuclear spin is around 0.1 s, which is far less than the achieved coherence in [49, 171] (1 second). The coherence time requirement is depending on the total ordering. To reduce the time, one may find a more efficient ordering, or add redundant nodes

to shorten the idling time. Moreover, high-fidelity one- and two-bit gate operations on the nuclear spins have been demonstrated [46]. Note that, our entangling rate estimation comes from an experiment with the nodes separated by 2 meters [175], with a rather low fidelity ~ 0.6 for this rate. Obtaining high fidelity entangled states with a high generation rate and a long distance remains a challenge to this date. We conclude that the current technology is still not yet ready to implement BOQC algorithms as in Figure 7.7, but foreseeable improvements would make it possible.

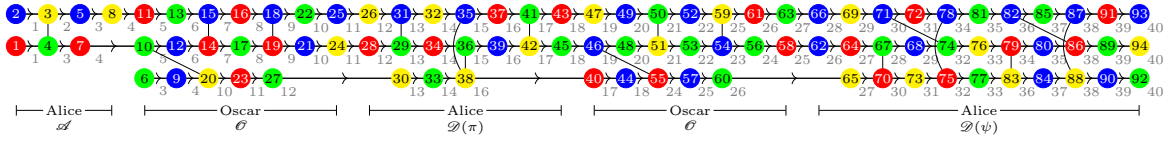


Figure 7.8: Three-qubit Grover-Høyer algorithm (see Figure 6.3 for more details) on the NV-center platform in Figure 7.7. Each node color corresponds to the same color of nuclear spin shown in Figure 7.7. The arrows represent the graph's flow (Section 2.4); the flow induces a partial ordering $\succ$ covering the nodes, where incomparable nodes belong to the same class represented by the grey label. The measurement order is shown by the node number, which is consistent with partial ordering $\succ$. The term idling time here indicates the duration from initialization to measurement for a given qubit. Using the gate times from Equation (7.8), the entire computation is runnable within ~ 3 s; on average, each qubit needs to maintain its coherence for ~ 121 ms while 1 RSP are performed on the electron qubit. The longest idling time happens on nodes 20 and 55 that need to maintain their coherence for ~ 377 ms, while 10 RSPs are performed on the electron spin.

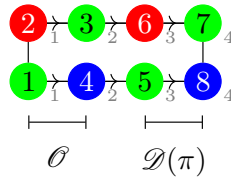


Figure 7.9: Two-qubit Grover algorithm on the NV-center platform in Figure 7.7; see Figure 6.4 for more details on the algorithm. The notation that we use here is identical to Figure 7.8. The execution time is estimated to be ~ 244 ms. On average, each qubit needs to maintain its coherence for ~ 91 ms while 1 RSP is performed on the electron qubit. The worst idling time happens on node 4 that needs to maintain its coherence for ~ 178 ms, while 4 RSPs are performed on the electron spin.

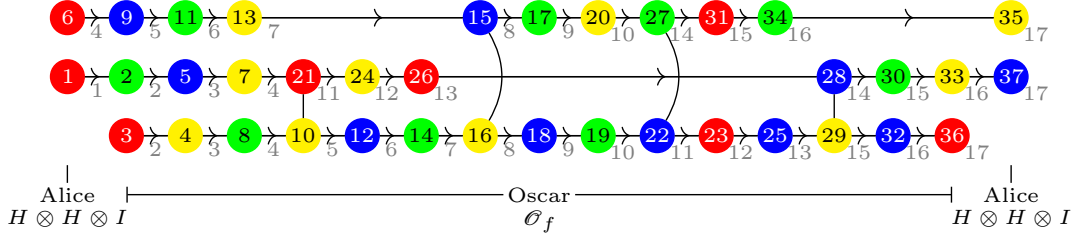


Figure 7.10: Two-qubit Simon algorithm with a useless oracle (from the construction in Definition 6.3) on the NV-center platform; further details of the algorithm are shown in Figure 6.6. The notation that we use here is identical to Figure 7.8. The execution time is estimated to be ~ 1.1 s. On average, each qubit needs to maintain its coherence for ~ 116 ms, while one RSP is performed on the electron spin. The longest idling time for a qubit to maintain its coherence is ~ 468 ms that happens on node 21, while 14 RSPs are performed on the electron spin.

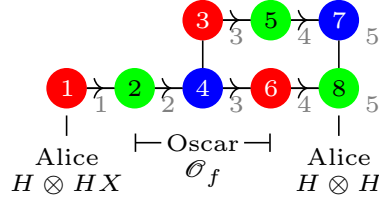


Figure 7.11: Deutsch's algorithm on the NV-center platform; further details of the algorithm are shown in Figure 6.7. The notation that we use here is identical to Figure 7.8. The whole computation takes ~ 125.5 ms. On average, each qubit needs to maintain its coherence for ~ 80 ms, while one RSP is performed on the electron spin. The longest coherence requirement is ~ 121 ms that happens on node 4, while 3 RSPs are performed on the electron spin.

SUMMARY AND OUTLOOK

SUMMARY

In this thesis, we introduce BOQC, a three-party quantum cloud computing scheme, where privacy is taken into account. The BOQC provides a means for a client (Alice) to delegate her oracular quantum computations on a 1WQC server (Bob), where the oracle is outsourced to a third party (Oscar). Alice and Oscar have only small quantum power, Oscar has access to the oracle, and Bob has a powerful quantum server. However, Alice and Oscar do not risk revealing information about their computation while running it on Bob’s quantum computer; thus, they run their computation within the BOQC scheme. Our scheme promises blindness, which means that Bob cannot learn anything about the computation beyond public information provided by his clients.

We provide several variations of BOQC protocols depending on the underlying physical availability: flying or solid-state qubits and one- or two-way quantum networks. We have proven that our BOQC protocols are secure within a composable definition; in particular, we use the abstract cryptography framework to quantify each protocol’s security. Our protocols offer composable blindness as security, which implies a BOQC protocol is also blind as a subsystem of a more extensive cryptographic system.

If the cloud quantum computer is solid-state based, we employ the BOQCo protocol instead. The BOQCo is an optimized version of BOQC, where the computation is run within the lazy 1WQC paradigm ([Section 2.5](#)). The lazy 1WQC scheme is optimized to carry out a 1WQC computation using a minimal number of physical qubits, given that the qubits possess permanence and can be rapidly re-initialized after being measured. We also propose a concrete scheme to implement a BOQCo protocol on an NV-center diamond platform in [Section 7.2](#), where Alice and Oscar

each require a diamond with one electron spin, whereas Bob has a diamond with an electron spin and several nuclear carbon spins (^{13}C). While the platform admits only the BOQCo protocol with classical input and output, this is sufficient to run most oracular quantum algorithms in general.

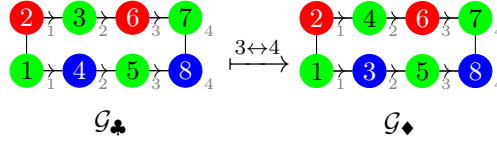
We construct several explicit oracular quantum algorithms that run on the BOQC scheme in [Chapter 6](#). The algorithms include 3- and 2- qubit exact Grover algorithms, Deutsch’s algorithm, and 2-qubit Simon’s algorithm with a useless oracle construction. Based on the security analysis of the BOQC scheme, the computation must be BOQC-compatible ([Section 5.2.1](#)), namely it fulfills some criteria imposed upon the graph state in order to maintain perfect blindness. We propose a numerical construction (BGS algorithm in [Algorithm 6.3](#)) to obtain such BOQC-compatible graphs. We conclude that we need 94 nodes to run 3-qubit exact Grover’s algorithm, 8 nodes to run 2-qubit Grover’s algorithm, 37 nodes to run 2-qubit Simon’s algorithm, and 8 nodes to run Deutsch’s algorithm. Every algorithm provides a set of functions $\{f_i\}$ that can be evaluated by the oracle, where Oscar chooses the function f_i accordingly. If we ran these algorithms within the BOQCo scheme, the requirement of physical qubits are considerably reduced: 4 qubits to run 3-qubit exact Grover algorithm, 3 qubits to run 2-qubit Grover’s algorithm, 4 qubits to run 2-qubit Simon’s algorithm, and 3 qubits to run Deutsch’s algorithm. Finally, in [Section 7.2](#), we illustrate how each algorithm is executed within the BOQCo protocol on an NV-center platform. Using some experimental results, we estimate the coherence time needed on each qubit and the run time of the entire algorithm. Although each nuclear qubit needs to remain coherent for much less than one second — hitherto the best known coherence time of the NV-center nuclear spin, we conclude that the current technologies are not yet ready to run the BOQC algorithms presented in this thesis, but foreseeable improvements would make it possible.

OUTLOOK

There are several open problems and some ideas that have potential as a follow-up study, which we categorize them into four directions (topics): **(D1)** graph state optimization, **(D2)** security analysis, **(D3)** potential BOQC algorithms, and **(D4)** physical implementation.

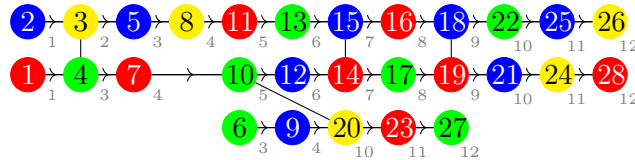
First, in the category **(D1)**, the following problem is related to graph state ordering and local complementary modification. In the lazy 1WQC scheme, discussed in [Section 2.5](#), the total ordering $>$ is determined at random, as long as it follows the partial ordering $\succ$ that is induced by the flow. Determining an ordering $>$

at random can lead to poor performance of computation; in particular, a bad ordering increases idling time for some nodes, while too many RSPs are performed on the electron spin. Heralded entanglement attempts significantly contribute to decoherence of nuclear spins [190]. Thus, reducing RSPs improves nuclear spin coherence. A proper total ordering can reduce the RSPs, which improves the coherence of nuclear spins. For an illustration, consider the following re-ordering of the graph from Figure 7.9.



Node 4 of graph $\mathcal{G}_{\clubsuit}$ needs to stay coherent while nodes 2, 3, 5, and 6 are being transmitted via RSP — this excludes other activities such as operating CPHASE gates, SWAP gates, and measurements. On the other hand, node 4 of graph $\mathcal{G}_{\diamondsuit}$ needs to maintain its coherence while nodes 3, 5, and 6 are transmitted. Hence, swapping the measurement order $3 \leftrightarrow 4$ reduces the number of RSPs by one. This reduction will be more apparent for graphs that have many rows. Remark that measurement order swaps are only valid for the incomparable nodes within the partial ordering $\succ$. Thus, we need to find a systematic way to find the best total ordering.

However, for graphs with more complex connectivity, finding the best total ordering might not show much of the improvement. Consider the following case.



Node 20 has the worst idling time; this is because node 20 is connected to node 9. After initialization of node 20, nodes 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, and 23 are transmitted via RSP. To reduce the idling time, we can add redundant nodes accordingly, propagating the information to fresher qubits; however, this strategy increases the graph order. Another possibility is to find an equivalent graph by applying a series of local complementations (Definition 2.12) accordingly, such that the idling times are improved. While local complementary operations are performed, we keep track of each measurement angle which changes according to the transformation in Definition 2.13. Finally, we recalculate the flow.

Second, in the category **(D1)**, the following problem is related to the unitary decomposition in the BGS algorithm (Algorithm 6.3). The optimization in step **(S1)** of the BGS algorithm is exhaustive, where the topology is found by trying all unique topologies with a certain size. However, the BGS algorithm yields compact circuits with a minimal number of CNOT gates. For instance, we obtain 3-qubits exact Grover algorithm with 17 CNOT gates (Figure 6.2), which are composed of 5 $SU(2^3)$ unitary matrices. Any $SU(2^n)$ matrix can be realized with $\frac{1}{4}(4^n - 3n - 1)$ CNOT gates [192]; our result is very much below the upper bound (70 CNOT gates). Several efficient systematic decompositions of arbitrary unitary matrices have been studied [193, 194]; although the number of CNOT gates are exponential, incorporating their techniques into the BGS algorithm followed by several layers of optimizations might yield interesting results.

Third, in category **(D2)**, is adding verifiability to the BOQC scheme. Verifiability is one step forward after achieving blindness, namely a mechanism that predicts with a high probability whether Bob was cheating; thus, Alice may reject or accept the final outcome accordingly. A verifiability scheme for UBQC has been investigated [88] and experimentally demonstrated [39], where the scheme uses a *trap-based* verification. For instance, adapting their verifiability scheme into the BOQC scheme results in Figure 8.1, using the resource state defined in [88]. Besides, their verifiability scheme has been proven to be composable secure [113].

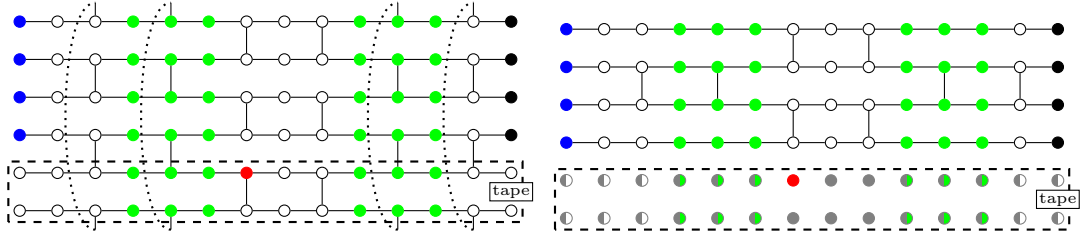


Figure 8.1: A verifiable BOQC protocol inspired by the verifiable UBQC scheme [88]. The blue, white, green, black, and red nodes denote Alice’s input nodes, Alice’s auxiliary nodes, Oscar’s auxiliary nodes, Alice’s output nodes, and trap nodes, respectively. The resource state forms a cylindrical brickwork state. Alice and Oscar randomly choose one trap node, and “tape” the rows connected to the trap. The nodes inside the tape are dummy nodes and are not contributing to the computation nor verifiability. Nodes outside the tape are used for the computation. The dummy qubits are randomly prepared in the z -basis, thus they are not entangled — as shown in the right figure, and they do not affect the computation. The trap node t is prepared in state $|+\theta_t\rangle$. Given that Alice sets the measurement angle as $\delta_t = \theta_t + r_t\pi$, she accepts the final outcome (on the black nodes) if Bob’s measurement outcome of the trap is r_t , otherwise she rejects the final result.

The verifiability scheme in Figure 8.1 only works for brickwork states, which requires many nodes. Therefore, how to come up with a trap-based verifiable

scheme for the BOQCo protocol is worth investigating. For instance, by placing one trap for each round of measurements without disturbing the computation, or placing several traps on an arbitrary (total) graph.

Fourth, in the category **(D2)**, is to the studying of a composable security analysis of the BOQC in a more realistic setting, *i.e.*, in the presence of noise. To this date, composable security proofs for noisy computations have not been done even for general delegated quantum computation schemes. Thus, one can start from a two-party delegated quantum computation with player Alice (A) and Bob (B). In the AC framework, we define a noisy ideal resource $\tilde{S}$ and construct the real-world resource $\tilde{R}$ that might also be noisy. We construct the correctness statement $\tilde{R} \xrightarrow{\pi, \varepsilon} \tilde{S}$ for a protocol $\pi = \{\pi_A, \pi_B\}$ and failure ε . The security statement is $\tilde{R}' \xrightarrow{\pi, \varepsilon} \tilde{S}'$ for a malicious Bob, which is achieved by showing that there exists a simulator σ_B , such that $d(\tilde{R}'\pi_A, \tilde{S}'\sigma_B) \leq \varepsilon$, where $\tilde{R}'$ and $\tilde{S}'$ are the real-world resource and the ideal-world resource in the presence of an adversary, respectively. The failure ε is determined such that the failure comes from the noise with high probability — not from Bob deviating from his protocol.

Fifth, in the category **(D3)**, is constructing useless oracle problems where the oracle has internal randomness [43]. Here, we refer to the useless oracle introduced in [43], which provides strong classical-quantum separations, and potentially results in exciting algorithms. A Simon algorithm with useless oracle has been investigated in [169] and reviewed in Section 6.5 with the separation infinity-vs- $\Theta(n)$. Several oracles with stronger separation, *i.e.*, infinity-vs-one are provided in [43]. Constructing an oracle that provides infinity-vs-one separation in the BOQC scheme is quite interesting to investigate.

Sixth, in the category **(D3)**, is incorporating a compression scheme into a BOQC algorithm, in particular running a hashed-Simon algorithm in the BOQC scheme. We refer to a hashing technique by May and Schlieper [195], which reduces the size of the ancilla register (Oscar's computation space) from n to 1; it means a hashed-Simon algorithm requirement is reduced from $2n$ to $n + 1$ qubits. Incorporating compression into a BOQC algorithm could be interesting, *e.g.*, one could aim at factoring an integer experimentally in the BOQC setting.

Lastly, in the category **(D4)**, is improving the NV-center platform in Figure 7.7 by allowing quantum input-output while maintaining blindness. For instance, an RSP scheme introduced by Xiang et al. [196] can potentially be incorporated into the BOQC scheme as quantum input-output transmission.

As the final remark, BOQC algorithms have the potential to provide exciting experiments in a setting in which the oracle paradigm is physically meaningful.

The next (experiment) step is to experimentally demonstrate BOQC algorithms on a physical platform, which can be done on any platform that admits the setup in [Figure 7.7](#). The potential physical systems that may implement BOQC algorithms are not limited to NV-center systems, but also quantum dots with an extra electron and trapped ions — based on [\[189\]](#). Moreover, when the quantum technology is mature and widely implemented, BOQC protocols can accommodate client-server situation in more realistic settings, *e.g.*, as Deutsch initially suggested [\[8\]](#), when the oracle evaluates an extensive classical computation to decide on time whether to buy or to sell some stock in a stock market.

LIST OF FIGURES

Figure 2.1	An illustration of 1WQC by Raussendorf	8
Figure 2.2	Graph $\mathcal{G}$	28
Figure 2.3	Measure node 3 of graph $\mathcal{G}$ with the timeline shown.	28
Figure 2.4	Eliminating acausality by applying the stabilizer of the graph state	28
Figure 2.5	The correction of measuring node 3 with gflow.	29
Figure 3.1	An illustration of blind quantum computation by Fitzsimons	50
Figure 3.2	An example of how Alice's computation look like in a delegated quantum computing scheme	54
Figure 3.3	Task separation between Alice and Bob	56
Figure 3.4	A brickwork state	57
Figure 3.5	Arbitrary 1-qubit rotations and CNOT gate on the brickwork state	57
Figure 4.1	A computing machine $\mathfrak{M}$	66
Figure 4.2	Turing Machine	66
Figure 4.3	A machine $\mathfrak{M}$ that solves a promise problem.	67
Figure 4.4	Illustration of classical and quantum oracles in the standard model	71
Figure 4.5	Illustration of process in Grover's algorithm	78
Figure 5.1	A cartoon illustrating the BOQC scheme	92
Figure 5.2	Communication diagram in the BOQC scheme	97
Figure 5.3	The ideal BOQC resource $\mathcal{S}$	102
Figure 5.4	Ideal resource $\mathcal{S}'$	104
Figure 5.5	Resource $\pi_{\mathcal{A}}\pi_{\mathcal{O}}\mathcal{R}$	105
Figure 5.6	Resource $\pi_{\mathcal{A}\mathcal{O}}\mathcal{R}$	106
Figure 5.7	Resource $\mathcal{S}'\sigma_{\mathcal{B}}$	106
Figure 6.1	Example of unique topologies for $n = 3$ and $N = 4$	129
Figure 6.2	The output of steps S1 and S3.1 from the BGS algorithm .	145
Figure 6.3	Three-qubit Grover-Høyer algorithm with $\mathbf{a} = 012345$. . .	146
Figure 6.4	Two-qubit Grover algorithm in BOQC scheme	147
Figure 6.5	Circuit representation of a useless Simon oracle	149
Figure 6.6	Two-qubit Simon's algorithm in the BOQC scheme	151
Figure 6.7	Deutsch's algorithm within the BOQC scheme	153
Figure 7.1	NV-center in diamond	155

Figure 7.2	Electronic configuration $a_1^2 e^2$	158
Figure 7.3	Spin dynamics of an NV-center	159
Figure 7.4	Readout (RO) and initialization (INIT) resonance excitation	160
Figure 7.5	Energy levels of the ground state $^3A_2$	161
Figure 7.6	The setup for Barrett and Kok's heralded scheme	163
Figure 7.7	The NV-center platform for the BOQC scheme	167
Figure 7.8	Three-qubit Grover-Høyer (BOQC) algorithm on NV-center platform	168
Figure 7.9	Two-qubit Grover (BOQC) algorithm on NV-center platform	168
Figure 7.10	Two-qubit Simon (BOQC) algorithm on NV-center platform	169
Figure 7.11	Deutsch's (BOQC) algorithm on NV-center platform	169
Figure A.1	Plot from <code>result.json</code>	191

LIST OF TABLES

Table 2.1	Descriptions of node shapes used in this thesis.	12
Table 2.2	Basic commands in patterns.	23
Table 5.1	Minimal quantum power requirements in the BOQC	111
Table 6.1	The counting of all possible gate topologies	126
Table 6.2	The number of unique topologies with size N for n qubits .	128
Table 6.3	Equivalent databases for $N \in \{5, 6, 7, 8\}$	140
Table 6.4	Counting for minimal CNOT gates to run Grover-Høyer algorithm for all unique combination of database	142

NOMENCLATURE

Acronyms

Symbol	Description
1WQC	One-Way Quantum Computer
	Extra box, additional discussion that is not closely related to the thesis
AC	Abstract Cryptography or Alternating Current
BGS	BOQC-compatible Graph Search (algorithm)
BOQC	Blind Oracular Quantum Computation
BOQCo	Optimized Blind Oracular Quantum Computation
BQC	Blind Quantum Computing
BR	Bit Relabelling (criterion)
BS	Beam splitter
CPTP	Completely positive trace-preserving map
CS	Conjugation by Swapping (criterion)
ESR	Electron Spin Resonance
INIT	Initialization
ISC	Inter-system Crossing
MW	Microwave
NV	Nitrogen-Vacancy
OTP	One-time Pad
QHE	Quantum Homomorphic Encryption
QKD	Quantum Key Distribution
QOTP	Quantum One-time Pad
QTM	Quantum Turing Machine

RO	Readout
RSP	Remote State Preparation
TM	Turing Machine
TR	Time Reversal (criterion)
UBQC	Universal Blind Quantum Computation
ZPL	Zero Phonon Line

Greek Symbols

Symbol	Description
α_i	Random angles generated by Alice to encrypt her qubits
β_i	Random angles generated by Oscar to encrypt his qubits
δ_i	Measurement angle broadcast to Bob
ω_L	Larmor frequency
ϕ'_i	Measurement angle of node i , after correction, where i belongs to Alice.
ϕ	An angle $\phi \in [0, 2\pi)$ or a set of angles $\phi = \{\phi_i\}$, $\phi \in [0, 2\pi)$
Π, Π_s	A promise problem and projection in measurements of a pattern with outcomes s , respectively
π_i	A converter of party i
ψ'_i	Measurement angle of node i , after correction, where i belongs to Oscar.
ψ	An angle $\psi \in [0, 2\pi)$ or a set of angles $\psi = \{\psi_i\}$, $\psi \in [0, 2\pi)$
ρ	A density of state
σ_B	Bob's simulator
Σ	$\{0,1\}$
τ	Delay time in a pulse sequence or an oracle query
$\vec{\Phi}, \Phi$	Free-valued parameters of an optimization

Operators

Symbol	Description
--------	-------------

$\oplus$	Modulo 2 evaluation
$\otimes$	Tensor product
$\uplus$	Union of multisets
$invf(i)$	$f^{-1}(i)$ preimage of flow $f(i)$, viz., i
LC_j	Local complementary with respect to node i
$Odd(i)$	Odd neighborhood of node i
$\Pr[A], p(A)$	Probability of event A
$N_{\mathcal{G}}(j)$	Open neighborhood of node j — j is not included
$N_{\mathcal{G}}[j]$	Closed neighborhood of node j — j is included
$\rhd\!\!\!\rhd$	Ordered product, goes from the left to the right, according to a total ordering $>$
$\rhd\!\!\!\rhd$	Ordered product, goes from the left to the right, according to a partial ordering $\succ$
$\text{tr}_A[\rho]$	A partial trace, tracing subsystem A
$A^c, \mathcal{F}^c$	Complement of set A and complement of graph $\mathcal{F}$, respectively
$E_{\mathcal{G}}$	Entangling operations $\prod_{(i,j) \in E(\mathcal{G})} E_{ij}$
$E_{i,j}$	Entanglement command, i.e., apply CPHASE_{ij}
$E_{iK}^>$	Entangling operations given some ordering, i.e., $\prod_{k \in K, k > i} E_{ik}$
E_{iK}	Entangling operations $\prod_{k \in K} E_{ik}$
M_i^θ	Measurement command, i.e., measure i in θ
N_j^θ	Preparation command, i.e., prepare qubit(s) j in state $ +\rangle_\theta$

Quantum Gates, Operators, and States

Symbol	Definition	Description
CNOT	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	Controlled- X
CPHASE	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$	Controlled- Z

$ 0\rangle_C, 1\rangle_C$	$ m_{s,C} = -1/2\rangle, m_{s,C} = 1/2\rangle$	Spin electron state encoding
$ 1\rangle_e, \downarrow\rangle$	$ m_s = -1\rangle$	Spin electron state encoding of $ 1\rangle$
$ 1\rangle_e, \downarrow\rangle$	$ m_s = 0\rangle$	Spin electron state encoding of $ 0\rangle$
$ \psi\rangle$	$\alpha 0\rangle + \beta 1\rangle, \alpha ^2 + \beta ^2 = 1$	An arbitrary quantum state
$ \pm_\theta\rangle$	$\frac{ 0\rangle \pm e^{i\theta} 1\rangle}{\sqrt{2}}$	A quantum state on the xy -plane
$ \pm\rangle$	$\frac{ 0\rangle \pm 1\rangle}{\sqrt{2}}$	Eigenbasis of Pauli X
σ_x, X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	Pauli X
σ_y, Y	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	Pauli Y
σ_z, Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	Pauli Z
H	$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	Hadamard
$R_x(\theta)$	$HR_z(\theta)H$	Rotation around x -axis with angle θ
$R_z(\theta), Z(\theta)$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$	Rotation around z -axis with angle θ

Roman Symbols

Symbol	Description
d	A database
$\ell_{\mathcal{AO}}$	Information leak from Alice and Oscar
$\mathcal{R}, \mathcal{S}, \mathcal{S}'$	Resources in Abstract Cryptography
$\mathcal{E}$	A CPTP map
$\mathcal{F}, \mathcal{G}$	Graphs
$\mathcal{H}$	A Hilbert space
$\mathcal{I}$	A set of interfaces
$\mathcal{O}$	An oracle, Oscar, or a big-O notation
$\mathcal{R}$	A relation

S_g	A stabilizer generator
$\mathcal{S}$	A stabilizer group with elements $\{S_j\}$
$\mathcal{T}, \mathcal{K}, \mathcal{B}$	Topology of a two-bit gate network
$\mathcal{A}, \mathcal{B}, \mathcal{O}$	Alice, Bob, and Oscar, respectively
$\mathcal{A}, \mathcal{A}$	Preparation operator in Grover's iteration; the latter is adapted to
$\mathcal{O}_f, \mathcal{O}_f$	Oracles adapted to a particular quantum computer, with function evaluation f
$\mathfrak{G}$	Quantum gates composition with gate set $\{\text{CNOT}, U(\alpha, \beta, \gamma)\}$
$\mathfrak{J}$	Quantum gates composition with gate set $\{\text{CPHASE}, R_z(\phi)\}$
$\mathfrak{M}(x), \mathfrak{M}$	A Turing machine that evaluates x . Also used to indicate measurement projectors.
$\mathfrak{P}$	A measurement pattern
$\mathcal{P}(A)$	The powerset of set A
$\mathcal{T}, \mathcal{K}$	A list of topologies of two-bit gate networks
$\mathcal{U}, \mathcal{O}$	A set of unitary matrices or oracles, respectively
$\mathcal{D}, \mathcal{D}(x)$	Diffusion operator adapted to a particular quantum computer, without and with parameter, respectively
EPR	Entangled pair of qubits with state $(00\rangle + 11\rangle)/\sqrt{2}$
Ω	A set of allowed measurement angles $\Omega = \{\frac{n\pi}{2^{b-1}}\}_{0 \leq n \leq 2^b}$, for an integer b
V_A, V_O	A set of nodes belonged to Alice and Oscar, respectively
$A(i)$	The set of fresh nodes must be allocated at time step i
A_1, A_2, E	Symmetry operations of the C_{3v} point group
A_C	Hyperfine coupling constant of a carbon nuclear spin
b	The number of bit that can represent measurement angles
C_{3v}	Point group symmetry of the NV-center
C	Carbon or connectivity of graphs
$D, D(x)$	Diffusion operator, without and with parameter, respectively
$E, E(\mathcal{G})$	Edges of graph $\mathcal{G}$
$f(i), f(x)$	A measurement flow of node (i). The $f(x)$ is also used to indicate an evaluation done by an oracle with query x

$f, \mathfrak{f}, F$	Objective functions
$g(i)$	A measurement gflow
G	Grover's iterator, or a graph
I, O	A set of input and output nodes, respectively
I_C, I_N, S	Spin number operator for Carbon, Nitrogen, and electron, respectively
N, n	Nitrogen or integer
$Q(\varphi, \psi)$	Høyer iterator with parameters φ, ψ
r, t	A set of random binaries to hide measurement outcome, <i>i.e.</i> , $r \equiv \{r_i\}$ and $t \equiv \{t_i\}$, $r_i, t_i \in \{0, 1\}$
$s, \tilde{s}$	A signal or measurement outcome $s \in \{0, 1\}$ or a set of signals $s = \{s_i\}$, $s \in \{0, 1\}$
$U(\alpha, \beta, \gamma)$	A general form of SU(2) matrix with three free-valued parameters
$V, V(\mathcal{G})$	Vertices of graph $\mathcal{G}$
X_{yes}, X_{no}	Yes-instance and no-instance, respectively
$z(i)$	The power of Z-correction applied to node i

Appendices

A

SOFTWARE OVERVIEW

In the following, we overview several software used in this thesis that are accessible — some are simply not published.

A.1 MBQC

We develop the **MBQC** package to simulate MBQC (Measurement-based quantum computation) algorithms. We plan to provide simulations for MBQC, 1WQC, BOQC, and BOQCo computation. However, the development is still on-going, to follow the update, see [197]. Some functions in **MBQC** are ready, *e.g.*, the one in [Section A.1.1](#).

A.1.1 *Listing open graphs with flow*

Here, we discuss our numerical experiment that supports [Conjecture 2.1](#) in [Section 2.1](#). We provide a brute-force method that enumerates all possible open graph given some properties; this functionality is available in module `mbqc.qres`. Our enumeration produces only graphs with flow. Consider [Listing A.1](#) that demonstrates the generation of open graphs with flow.

***Listing A.1:** Generate all open graphs with 5 nodes, 2 inputs, and 2 outputs.*

```
1 import mbqc.qres as mres
2
3 #set the parameters
4 N = 5 #the number of total nodes |V|
5 n_I = 2 #the number of input nodes |I|
6 n_O = 2 #the number of output nodes |O|
```

```

7 | n_IO = 1 #the number of overlap between input and output nodes
8 |
9 | # (1) Return open graphs with flow, then record them in "out/result
   | .json"
10 | mres.bf_list_opengraphs(N, n_I, n_O, n_IO, outdir="out", outfname="
    | result.json")
11 |
12 | # (2) Plot all graphs in "out/result.json", and store them in "out"
13 | mres.plotfrom_jsonfile("out/result.json", outdir="out")

```

Listing A.2: content of out/result.json from Listing A.1

```

1 | { "nnodes": 4,
2 |   "I": [0,1],
3 |   "O": [1,3],
4 |   "IO": [1],
5 |   "ngraphs": 7,
6 |   "time": 0.029191970825195312,
7 |   "intedges": [[3,12,5],[12,5,6],[3,12,5,6],[10,12,5],
8 |               [10,3,12,5],[10,12,5,6],[3,5,6,10,12]]
9 | }

```

There are two main commands in [Listing A.1](#): command (1) generates all open graphs with a flow, where the graphs have five vertices, two input nodes, two output nodes, and one node overlaps between input and output nodes, and command (2) stores the result in a file.

Using edges convention from [Equation \(6.17\)](#) and input-output convention in [Definition 5.1](#), all nodes, input nodes, and output nodes are initially numbered with $V = \{0 \dots N - 1\}$, $I = \{0 \dots n_I - 1\}$, and $O = \{N - n_O - 1 \dots N - 1\}$ correspondingly. If $n_{IO} \neq 0$, we replace the n_{IO} smallest number of elements in O with the n_{IO} largest elements in I . For instance, for the case in [Listing A.1](#), $V = \{0, 1, 2, 3, 4\}$, $I = \{0, 1\}$, and $O = \{1, 4\}$. Note that here we do not include yet the notion of classical or quantum.

The output from the first command is stored in `out/result.json` file, which comprises the information of the graphs represented by their *integer edges* ([Equation \(6.17\)](#)). The file `result.json` has a content listed in [Listing A.2](#), where the “time” corresponds to the execution time — in seconds — to list all opengraphs with flow, the “intedges” comprises integer edges correspond to the opengraphs, and the rest have self-explanatory names.

In the code implementation, we mostly work with integer edges for practical reasons.

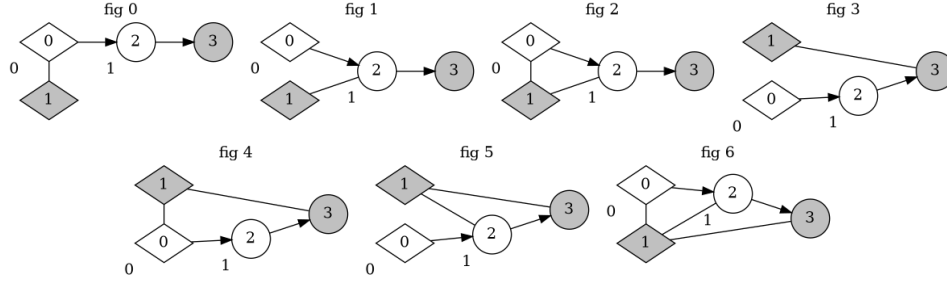


Figure A.1: The plot from `result.json` in [Listing A.1](#), with ordering from left to the right, e.g., the first figure (fig 1) corresponds to a graph with integer edges [3, 12, 15] and the last figure (fig 6) corresponds to a graph with integer edges [3, 5, 6, 10, 12]. Every node is indicated with an integer inside the node, the flow is indicated with arrows, and the numbers outside the nodes indicate a measurement ordering that follows the flow. The shape and color of nodes indicate the type of nodes, where the convention is taken from [Table 2.1](#), [Section 2.1](#).

The second command from [Listing A.1](#) reads file `result.json` and plot all the graphs from edges written there, where the results are shown in [Section A.1.1](#). There are seven graphs present; also notice that the CNOT graph can be seen there as fig 1.

[Algorithm A.1](#) is the algorithm behind open graph generation, which is rather simple, this is due to the fact that we are only dealing with a simple connected graph (see [Definition 2.5](#)).

Algorithm A.1 Brute-force listing of open graphs with flow

Input: (V, I, O)

Output: a list of graphs with flow

- 1: $E \leftarrow$ 2-combinations of V ▷ edges of a complete graph
 - 2: $P^E \leftarrow$ Power set of E
 - 3: **for** $\tilde{E} \in P^E$ **do**
 - 4: **if** $\mathcal{G}[\tilde{E}]$ is connected **then** ▷ elimination of non-connected graphs
 - 5: **if** flow of $\mathcal{G}[\tilde{E}]$ exists **then**
 - 6: **yield** $\tilde{E}$
 - 7: **end if**
 - 8: **end if**
 - 9: **end for**
-

We first list the all possible edges E — edges of the corresponding complete graph — given a set of nodes V . Secondly, we create a power set of E , indicated with P^E . Finally, for every combination of edges $\tilde{E} \in P^E$, we create a graph with edges $\tilde{E}$,

indicated with $\mathcal{G}[\tilde{E}]$, then yield $\tilde{E}$ if $\mathcal{G}[\tilde{E}]$ is connected (see [Definition 2.4](#)) and has a flow.

In practice, we work with integer edges instead for simplicity. For instance, the power set P^E can be straightforwardly generated by enumerating integers $\{0 \dots 2^{|E|} - 1\} =: S$ using [Equation \(6.17\)](#).

The simplicity of a brute-force algorithm comes with a big cost; the running time of [Algorithm A.1](#) grows exponentially. The size of sets are $|E| = \frac{n(n-1)}{2} \approx n^2$, and thus $|P^E| \approx 2^{n^2}$. Therefore, the algorithm has complexity $O(2^{\text{poly}(n)})$, which makes it belong the class EXPTIME.

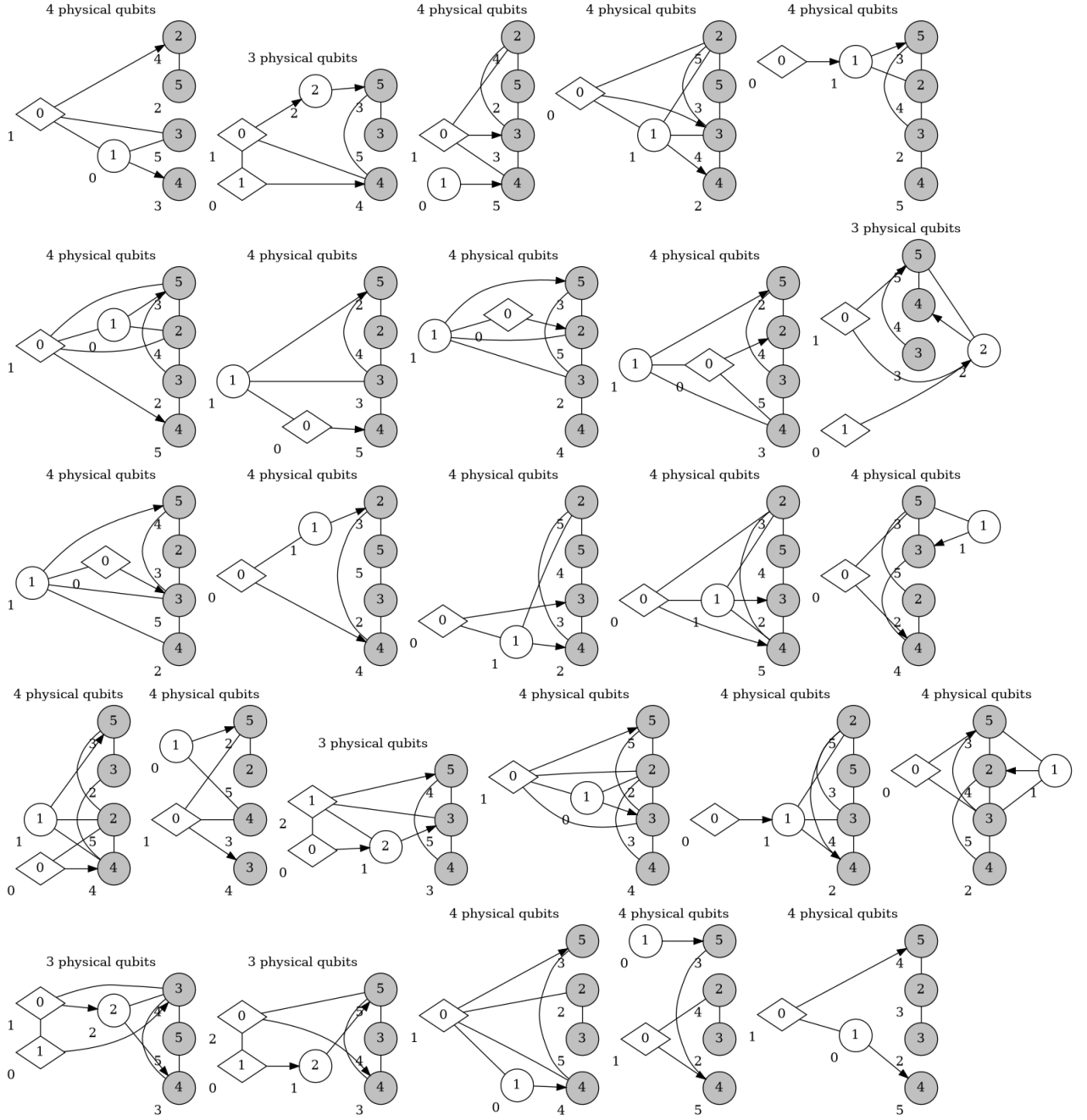
It is important to remark that, the resulting open graphs in [Algorithm A.1](#) are not necessarily unique. Here, we consider graphs without isomorphism, since we consider the position of input and output nodes are fundamental in determining flow. Thus, isomorphism is irrelevant for such graphs. However, room for improvement in [Algorithm A.1](#) is encouraged. For instance, one can provide an efficient non-isomorphic graphs iteration only for the auxiliary nodes¹, then connect the inputs and output nodes accordingly. This non-isomorphic graph iteration will be similar to `unique2net` algorithm in [Algorithm 6.2](#).

Finally, to increase the confidence of [Conjecture 2.1](#), we run [Algorithm A.1](#) for several cases, where the number of input nodes is larger than the output nodes, $n_I > n_O$. There, we did not find any open graph with a flow given such a restriction. However, since the algorithm grows exponentially, our observation is limited to eight nodes.

A.1.2 Loose bound cases

The following graphs are produced using `MBQC` program in [Section A.1](#). In particular, the following listed graphs show some cases with loose bound, according to [Conjecture 2.1](#).

¹ Auxiliary nodes refer to $V \setminus (I \cap O)$; it is often found in other literatures



A.2 `cgthesis_supplement`

`cgthesis_supplement` is a collection of small codes used in this thesis that comprises: `uniquedb` (Section A.2.1) and all **Mathematica** codes presented in Sections B.1 to B.5. The collection is openly accessible in [198].

A.2.1 `uniquedb`

Package `uniquedb` is written in **Python**, and it is included in the `cgthesis_supplement` (Section A.2). The engine of `uniquedb` is shown in Algorithm 6.4. `uniquedb` lists all unique combination of database items according to criteria discussed in Section 6.4.

Listing A.3: Unique database combination

```

1 >>> from uniquedb import uniquedb
2
3 >>> #list unique database with 5 items
4 >>> uniquedb(5)
5 [(0, 1, 2, 3, 4), (0, 1, 2, 4, 7), (0, 1, 2, 5, 6)]
6
7 >>> #list unique database with 8 items
8 >>> uniquedb(8)
9 [(0, 1, 2, 3, 4, 5, 6, 7)]

```

A.3 Quantum

Quantum is an add-on in **Mathematica** developed by Muñoz and Delgado [147]. The comprehensive tutorials and instructions are provided in [199]. **Quantum** has engine linear algebra and provides convenient syntax that follows bra-ket and Dirac's notations. After installing the add-on, the operators defined in **Quantum** can be loaded by command:

```
Needs["Quantum'Computing'];
```


B

MATHEMATICA CODES

B.1 GROVER'S ALGORITHM

```
Needs["Quantum`Computing`"];

SetComputingAliases [];

(* n: number of qubits, request: a set of requested data *)
Grover[n_, request_] := Module[
  (* temporary variables *)
  {K, X,  $\Psi$ init, Of,  $\Psi$ k, D},
  X = Intersection[request, Range[0, 2^n - 1]];
  (* preparation *)
   $\Psi$ init = ( $\mathcal{H}_i \cdot |0_i\rangle$ )n;
  | $\Psi$ k[0] =  $\Psi$ init;
  If[Length[X] > 0,
    (* Number of Grover's iteration *)
    K = IntegerPart[Round[ $\frac{\pi}{4 \text{ArcSin}[\sqrt{\text{Length}[X] / 2^n}]} - \frac{1}{2}$ ]];
    (* Oracle *)
    Of = 1 - 2 Sum[ |i>n · <i|n, {i, X}];
    (* Diffusion *)
    D = 1 - 2  $\Psi$ init · ( $\Psi$ init)*;
    ,
    (* No overlap cases *)
    K = 0; Of = 1; D = 1;
  ];
  Do[ | $\Psi$ k[k] = Expand[QuantumEvaluate[D · Of · | $\Psi$ k[k - 1]]], {k, 1, K}];
  Print["Number of qubits : ", n];
  Print["Requested items : ", X];
  Print["Number of iterations : ", K];
  (* final measurement *)
  TraditionalForm[
    N[QuantumEvaluate[QubitMeasurement[ | $\Psi$ k[K]>, Table[ $\hat{k}$ , {k, 1, 2^n}]]]]]]
]
```

Grover[2, {1}]

Number of qubits : 2

Requested items : {1}

Number of iterations : 1

itionalForm=

Probability	Measurement	State
1.	(0 ₁ 1 ₂)	$ 0\rangle \otimes (-1. 1\rangle)$
Probability	Measurement	State

Grover[4, {2, 3, 7, 11, 83}]

Number of qubits : 4

Requested items : {2, 3, 7, 11}

Number of iterations : 1

itionalForm=

Probability	Measurement	State
0.25	(0 ₁ 0 ₂ 1 ₃ 0 ₄)	$ 0\rangle \otimes 0\rangle \otimes 1\rangle \otimes (-1. 0\rangle)$
0.25	(0 ₁ 0 ₂ 1 ₃ 1 ₄)	$ 0\rangle \otimes 0\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
0.25	(0 ₁ 1 ₂ 1 ₃ 1 ₄)	$ 0\rangle \otimes 1\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
0.25	(1 ₁ 0 ₂ 1 ₃ 1 ₄)	$ 1\rangle \otimes 0\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
Probability	Measurement	State

Grover[3, {2, 3, 7, 11, 83}]

Number of qubits : 3

Requested items : {2, 3, 7}

Number of iterations : 1

itionalForm=

Probability	Measurement	State
0.03125	(0 ₁ 0 ₂ 0 ₃)	$ 0\rangle \otimes 0\rangle \otimes 0\rangle$
0.03125	(0 ₁ 0 ₂ 1 ₃)	$ 0\rangle \otimes 0\rangle \otimes 1\rangle$
0.28125	(0 ₁ 1 ₂ 0 ₃)	$ 0\rangle \otimes 1\rangle \otimes (-1. 0\rangle)$
0.28125	(0 ₁ 1 ₂ 1 ₃)	$ 0\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
0.03125	(1 ₁ 0 ₂ 0 ₃)	$ 1\rangle \otimes 0\rangle \otimes 0\rangle$
0.03125	(1 ₁ 0 ₂ 1 ₃)	$ 1\rangle \otimes 0\rangle \otimes 1\rangle$
0.03125	(1 ₁ 1 ₂ 0 ₃)	$ 1\rangle \otimes 1\rangle \otimes 0\rangle$
0.28125	(1 ₁ 1 ₂ 1 ₃)	$ 1\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
Probability	Measurement	State

B.2 GROVER'S ALGORITHM WITH ARBITRARY DATABASE CHOICE

```
Needs["Quantum`Computing`"];
```

```
SetComputingAliases[];
```

```
(* n: number of qubits, request: a set of requested data *)
Grover2[database_, request_] := Module[
  (* temporary variables *)
  {n, nd, m, K, X,  $\Psi$ init,  $\Psi$ k, Of, D},
  X = Intersection[request, database];
  n = Ceiling[Log2[Max[database]+1]]; (* max number to represent database *)
  nd = Length[database];
  m = Length[X];
  (* preparation *)
   $\Psi$ init =  $\frac{1}{\text{Sqrt}[nd]}$  Sum[ $|i\rangle_n$ , {i, database}];
   $|\Psi_k[0]\rangle = \Psi$ init;
  If[m > 0,
    (* Number of Grover's iteration *)
    K = IntegerPart[Round[ $\frac{\pi}{4 \text{ArcSin}[\sqrt{(m/nd)]}$  -  $\frac{1}{2}$ ]];
    (* Oracle *)
    Of = 1 - 2 Sum[ $|i\rangle_n \cdot \langle i|_n$ , {i, X}];
    (* Diffusion *)
    D = 1 - 2  $\Psi$ init · ( $\Psi$ init)*;
    ,
    (* No overlap cases *)
    K = 0; Of = 1; D = 1;
  ];
  Do[ $|\Psi_k[k]\rangle = \text{Expand}[\text{QuantumEvaluate}[-D \cdot \text{Of} \cdot |\Psi_k[k-1]\rangle]]$ , {k, 1, K}];
  Print["Number of database : ", nd, " ,number of qubits : ", n];
  Print["requested items : ", request, " ,number of iterations : ", K];
  (* final measurement *)
  TraditionalForm[
    N[QuantumEvaluate[QubitMeasurement[ $|\Psi_k[K]\rangle$ ], Table[ $\hat{k}$ , {k, 1, 2n}]]]]
]
```

Grover2[{2, 3, 5, 7}, {2}]

Number of database : 4 ,number of qubits : 3

requested items : {2} ,number of iterations : 1

ditionalForm=

Probability	Measurement	State
1.	(0 ₁ 1 ₂ 0 ₃)	0>⊗ 1>⊗ 0>
Probability	Measurement	State

Grover2[{11, 13, 17, 19, 23}, {10}]

Number of database : 5 ,number of qubits : 5

requested items : {10} ,number of iterations : 0

ditionalForm=

Probability	Measurement	State
0.2	(0 ₁ 1 ₂ 0 ₃ 1 ₄ 1 ₅)	0>⊗ 1>⊗ 0>⊗ 1>⊗ 1>
0.2	(0 ₁ 1 ₂ 1 ₃ 0 ₄ 1 ₅)	0>⊗ 1>⊗ 1>⊗ 0>⊗ 1>
0.2	(1 ₁ 0 ₂ 0 ₃ 0 ₄ 1 ₅)	1>⊗ 0>⊗ 0>⊗ 0>⊗ 1>
0.2	(1 ₁ 0 ₂ 0 ₃ 1 ₄ 1 ₅)	1>⊗ 0>⊗ 0>⊗ 1>⊗ 1>
0.2	(1 ₁ 0 ₂ 1 ₃ 1 ₄ 1 ₅)	1>⊗ 0>⊗ 1>⊗ 1>⊗ 1>
Probability	Measurement	State

Grover2[{29, 31, 37, 41, 43, 47}, {29}]

Number of database : 6 ,number of qubits : 6

requested items : {29} ,number of iterations : 1

ditionalForm=

Probability	Measurement	State
0.907407	(0 ₁ 1 ₂ 1 ₃ 1 ₄ 0 ₅ 1 ₆)	0>⊗ 1>⊗ 1>⊗ 1>⊗ 0>⊗ 1>
0.0185185	(0 ₁ 1 ₂ 1 ₃ 1 ₄ 1 ₅ 1 ₆)	0>⊗ 1>⊗ 1>⊗ 1>⊗ 1>⊗ 1>
0.0185185	(1 ₁ 0 ₂ 0 ₃ 1 ₄ 0 ₅ 1 ₆)	1>⊗ 0>⊗ 0>⊗ 1>⊗ 0>⊗ 1>
0.0185185	(1 ₁ 0 ₂ 1 ₃ 0 ₄ 0 ₅ 1 ₆)	1>⊗ 0>⊗ 1>⊗ 0>⊗ 0>⊗ 1>
0.0185185	(1 ₁ 0 ₂ 1 ₃ 0 ₄ 1 ₅ 1 ₆)	1>⊗ 0>⊗ 1>⊗ 0>⊗ 1>⊗ 1>
0.0185185	(1 ₁ 0 ₂ 1 ₃ 1 ₄ 1 ₅ 1 ₆)	1>⊗ 0>⊗ 1>⊗ 1>⊗ 1>⊗ 1>
Probability	Measurement	State

Grover2[{53, 59, 61, 67, 71, 73, 79}, {71}]

Number of database : 7 ,number of qubits : 7

requested items : {71} ,number of iterations : 2

AdditionalForm=

Probability	Measurement	State
0.0214791	(0 ₁ 1 ₂ 1 ₃ 0 ₄ 1 ₅ 0 ₆ 1 ₇)	$ 0\rangle \otimes 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes 1\rangle \otimes 0\rangle \otimes (-1. 1\rangle)$
0.0214791	(0 ₁ 1 ₂ 1 ₃ 1 ₄ 0 ₅ 1 ₆ 1 ₇)	$ 0\rangle \otimes 1\rangle \otimes 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
0.0214791	(0 ₁ 1 ₂ 1 ₃ 1 ₄ 1 ₅ 0 ₆ 1 ₇)	$ 0\rangle \otimes 1\rangle \otimes 1\rangle \otimes 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes (-1. 1\rangle)$
0.0214791	(1 ₁ 0 ₂ 0 ₃ 0 ₄ 0 ₅ 1 ₆ 1 ₇)	$ 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes 0\rangle \otimes 0\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
0.871125	(1 ₁ 0 ₂ 0 ₃ 0 ₄ 1 ₅ 1 ₆ 1 ₇)	$ 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes 0\rangle \otimes 1\rangle \otimes 1\rangle \otimes 1\rangle$
0.0214791	(1 ₁ 0 ₂ 0 ₃ 1 ₄ 0 ₅ 0 ₆ 1 ₇)	$ 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes (-1. 1\rangle)$
0.0214791	(1 ₁ 0 ₂ 0 ₃ 1 ₄ 1 ₅ 1 ₆ 1 ₇)	$ 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes 1\rangle \otimes 1\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
Probability	Measurement	State

Grover2[{83, 89, 97, 101, 103, 107, 109, 113}, {113}]

Number of database : 8 ,number of qubits : 7

requested items : {113} ,number of iterations : 2

AdditionalForm=

Probability	Measurement	State
0.0078125	(1 ₁ 0 ₂ 1 ₃ 0 ₄ 0 ₅ 1 ₆ 1 ₇)	$ 1\rangle \otimes 0\rangle \otimes 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
0.0078125	(1 ₁ 0 ₂ 1 ₃ 1 ₄ 0 ₅ 0 ₆ 1 ₇)	$ 1\rangle \otimes 0\rangle \otimes 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes (-1. 1\rangle)$
0.0078125	(1 ₁ 1 ₂ 0 ₃ 0 ₄ 0 ₅ 0 ₆ 1 ₇)	$ 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes 0\rangle \otimes 0\rangle \otimes (-1. 1\rangle)$
0.0078125	(1 ₁ 1 ₂ 0 ₃ 0 ₄ 1 ₅ 0 ₆ 1 ₇)	$ 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes 1\rangle \otimes 0\rangle \otimes (-1. 1\rangle)$
0.0078125	(1 ₁ 1 ₂ 0 ₃ 0 ₄ 1 ₅ 1 ₆ 1 ₇)	$ 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes 1\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
0.0078125	(1 ₁ 1 ₂ 0 ₃ 1 ₄ 0 ₅ 1 ₆ 1 ₇)	$ 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes 1\rangle \otimes 0\rangle \otimes 1\rangle \otimes (-1. 1\rangle)$
0.0078125	(1 ₁ 1 ₂ 0 ₃ 1 ₄ 1 ₅ 0 ₆ 1 ₇)	$ 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes (-1. 1\rangle)$
0.945313	(1 ₁ 1 ₂ 1 ₃ 0 ₄ 0 ₅ 0 ₆ 1 ₇)	$ 1\rangle \otimes 1\rangle \otimes 1\rangle \otimes 0\rangle \otimes 0\rangle \otimes 0\rangle \otimes 1\rangle$
Probability	Measurement	State

B.3 EXACT GROVER-HØYER ALGORITHM

```
Needs["Quantum`Computing`"];

SetComputingAliases [];

Clear[Diffusion, Oracle, EGroverHoyer]

(* Diffusion, Oracle, and Phase operators. The number 1 is an identity matrix,
ALWAYS represented in the computational basis. Here,
in Chapter 4, we refer to basis  $\{\Psi_0, \Psi_1\}$  *)

Diffusion[Ψinit_, ϕ_] := QuantumEvaluate [1 - (1 - Exp[I ϕ]) * Ψinit · (Ψinit)*];

(* X is the set of targets; Oracle and Phase are identical *)

Oracle[X_, n_, ϕ_] := QuantumEvaluate [1 - (1 - Exp[I ϕ]) * Sum[ | i⟩n · ⟨i|n, {i, X}]];

(* database: a set of items, request: a set of requested data *)
EGroverHoyer [database_, request_] := Module[
  (* temporary variables *)
  {K, X, m, n, nd, a, θ, Θ, ϕ, ψ, ζ, ξ, Ψinit, Ψk},
  X = Intersection[request, database];
  m = Length[X];
  nd = Length[database];
  n = Ceiling[Log2[Max[database] + 1]];
  Print["Number of database : ", nd,
    ", number of qubits : ", n, ", requested items : ", request];
  (* preparation *)
  Ψinit =  $\frac{1}{\text{Sqrt}[nd]}$  Sum[ | i⟩n, {i, database}];
  | Ψk[0]⟩ = Ψinit // QuantumEvaluate ;
  If[m > 0,
    (* the resquest exists in database*)
    a = m/nd;
    θ = ArcSin[Sqrt[a]] // N; (* initial angle *)
    K = Floor[ $\left(\frac{\pi}{2} - \theta\right) / (2 \theta)$ ]; (* Number of Grover's iteration *)
    Θ =  $\frac{\pi}{2} - 2 K \theta - \theta$ ; (* The remaining rotation *)
```

```

(* Condition on  $\Theta$  *)
On[Assert];
Assert[Abs[ $\Theta$ ]  $\leq 2 \theta$ ];

(* Høyer's angles *)
 $\psi = \text{ArcCos}\left[1 - \frac{\text{Sin}[\Theta]^2}{2 a (1 - a)}\right] // N;$ 
 $\phi = 2 \text{ArcTan}[\text{Tan}[\psi / 2] (1 - 2 a)] // N;$ 
 $\xi = \text{Arg}[-a (1 - \text{Exp}[I \psi]) - \text{Exp}[I \psi]];$ 
 $\zeta = \text{Arg}[(1 - \text{Exp}[I \psi]) \text{Sqrt}[a] \text{Sqrt}[1 - a]] - \xi;$ 

(* Grover iterations *)
Do[ |  $\Psi_k[k]$  = Expand[QuantumEvaluate [
    -Diffusion[ $\Psi_{\text{init}}$ ,  $\pi$ ]·Oracle[X, n,  $\pi$ ]· |  $\Psi_k[k-1]$ ]], {k, 1, K}];

(* Høyer's amplitude amplification *)
|  $\Psi_k[K+1]$  = Exp[-I ( $\xi + \zeta$ )] *
    QuantumEvaluate [-Diffusion[ $\Psi_{\text{init}}$ ,  $\psi$ ]·Oracle[X, n,  $\phi + \zeta$ ]· |  $\Psi_k[K]$ ]];

Print["Number of Grover's iterations : ", K, ", Høyer's iterations: ", 1];
Print["Total Grover's rotation : ",
    2 K  $\theta$ , " rad, Høyer's rotation: ",  $\Theta$ , " rad"];
,
(* the request does not exists in database *)
K = -1;
];
TraditionalForm [
    N[QuantumEvaluate [QubitMeasurement [ |  $\Psi_k[K+1]$ ], Table[ $\hat{k}$ , {k, 1, 2n}]]]]]]
]

```

EGroverHoyer [{0, 1, 11, 107, 13, 17, 19, 23, 31, 2, 101, 103}, {23, 19, 23}]

Number of database : 12, number of qubits : 7, requested items : {23, 19, 23}

Number of Grover's iterations : 1, Høyer's iterations : 1

Total Grover's rotation : 0.841069 rad, Høyer's rotation : 0.309193 rad

ditionalForm=

Probability	Measurement	State
0.5	(0 ₁ 0 ₂ 1 ₃ 0 ₄ 0 ₅ 1 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 1⟩⊗ 0⟩⊗ 0⟩⊗ 1⟩⊗(1. 1⟩)
0.5	(0 ₁ 0 ₂ 1 ₃ 0 ₄ 1 ₅ 1 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 1⟩⊗ 0⟩⊗ 1⟩⊗ 1⟩⊗(1. 1⟩)
Probability	Measurement	State

EGroverHoyer [{0, 1, 11, 107, 13, 17, 19, 23, 31, 2, 101, 103}, {1}]

Number of database : 12, number of qubits : 7, requested items : {1}

Number of Grover's iterations : 2, Høyer's iterations : 1

Total Grover's rotation : 1.17137 rad, Høyer's rotation : 0.106582 rad

ditionalForm=

Probability	Measurement	State
1.	(0 ₁ 0 ₂ 0 ₃ 0 ₄ 0 ₅ 0 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗(1. 1⟩)
Probability	Measurement	State

EGroverHoyer [{0, 1, 11, 107, 13, 17, 19, 23, 31, 2, 101, 103}, {100}]

Number of database : 12, number of qubits : 7, requested items : {100}

ditionalForm=

Probability	Measurement	State
0.0833333	(0 ₁ 0 ₂ 0 ₃ 0 ₄ 0 ₅ 0 ₆ 0 ₇)	0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩
0.0833333	(0 ₁ 0 ₂ 0 ₃ 0 ₄ 0 ₅ 0 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 1⟩
0.0833333	(0 ₁ 0 ₂ 0 ₃ 0 ₄ 0 ₅ 1 ₆ 0 ₇)	0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 1⟩⊗ 0⟩
0.0833333	(0 ₁ 0 ₂ 0 ₃ 1 ₄ 0 ₅ 1 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 0⟩⊗ 1⟩⊗ 0⟩⊗ 1⟩⊗ 1⟩
0.0833333	(0 ₁ 0 ₂ 0 ₃ 1 ₄ 1 ₅ 0 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 0⟩⊗ 1⟩⊗ 1⟩⊗ 0⟩⊗ 1⟩
0.0833333	(0 ₁ 0 ₂ 1 ₃ 0 ₄ 0 ₅ 0 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 1⟩⊗ 0⟩⊗ 0⟩⊗ 0⟩⊗ 1⟩
0.0833333	(0 ₁ 0 ₂ 1 ₃ 0 ₄ 0 ₅ 1 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 1⟩⊗ 0⟩⊗ 0⟩⊗ 1⟩⊗ 1⟩
0.0833333	(0 ₁ 0 ₂ 1 ₃ 0 ₄ 1 ₅ 1 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 1⟩⊗ 0⟩⊗ 1⟩⊗ 1⟩⊗ 1⟩
0.0833333	(0 ₁ 0 ₂ 1 ₃ 1 ₄ 1 ₅ 1 ₆ 1 ₇)	0⟩⊗ 0⟩⊗ 1⟩⊗ 1⟩⊗ 1⟩⊗ 1⟩⊗ 1⟩
0.0833333	(1 ₁ 1 ₂ 0 ₃ 0 ₄ 1 ₅ 0 ₆ 1 ₇)	1⟩⊗ 1⟩⊗ 0⟩⊗ 0⟩⊗ 1⟩⊗ 0⟩⊗ 1⟩
0.0833333	(1 ₁ 1 ₂ 0 ₃ 0 ₄ 1 ₅ 1 ₆ 1 ₇)	1⟩⊗ 1⟩⊗ 0⟩⊗ 0⟩⊗ 1⟩⊗ 1⟩⊗ 1⟩
0.0833333	(1 ₁ 1 ₂ 0 ₃ 1 ₄ 0 ₅ 1 ₆ 1 ₇)	1⟩⊗ 1⟩⊗ 0⟩⊗ 1⟩⊗ 0⟩⊗ 1⟩⊗ 1⟩
Probability	Measurement	State

B.4 PROOF OF EXAMPLE 6.5

Needs["Quantum`Computing` "];
SetComputingAliases [];
$A = \# \rangle_3 \& /@ \{0, 1, 2, 3, 4\}$ $B = \# \rangle_3 \& /@ \{1, 0, 5, 4, 3\}$
$\{ 0_1, 0_2, 0_3 \rangle, 0_1, 0_2, 1_3 \rangle, 0_1, 1_2, 0_3 \rangle, 0_1, 1_2, 1_3 \rangle, 1_1, 0_2, 0_3 \rangle \}$
$\{ 0_1, 0_2, 1_3 \rangle, 0_1, 0_2, 0_3 \rangle, 1_1, 0_2, 1_3 \rangle, 1_1, 0_2, 0_3 \rangle, 0_1, 1_2, 1_3 \rangle \}$
(* A and B can be obtained from each other, by complement the third bit, and permute the first and the second bit *) QuantumEvaluate [SWAP _{1,2} · X ₃ · #] & /@ A QuantumEvaluate [SWAP _{1,2} · X ₃ · #] & /@ B
$\{ 0_1, 0_2, 1_3 \rangle, 0_1, 0_2, 0_3 \rangle, 1_1, 0_2, 1_3 \rangle, 1_1, 0_2, 0_3 \rangle, 0_1, 1_2, 1_3 \rangle \}$
$\{ 0_1, 0_2, 0_3 \rangle, 0_1, 0_2, 1_3 \rangle, 0_1, 1_2, 0_3 \rangle, 0_1, 1_2, 1_3 \rangle, 1_1, 0_2, 0_3 \rangle \}$
(* Marking: 0 on A --- 0 on B *) QuantumEvaluate [X ₁ · X ₂ · X ₃ · C ^(1,2) [Z ₃] · X ₁ · X ₂ · X ₃ · #] & /@ A QuantumEvaluate [X ₁ · X ₂ · X ₃ · C ^(1,2) [Z ₃] · X ₁ · X ₂ · X ₃ · #] & /@ B
$\{- 0_1, 0_2, 0_3 \rangle, 0_1, 0_2, 1_3 \rangle, 0_1, 1_2, 0_3 \rangle, 0_1, 1_2, 1_3 \rangle, 1_1, 0_2, 0_3 \rangle \}$
$\{ 0_1, 0_2, 1_3 \rangle, - 0_1, 0_2, 0_3 \rangle, 1_1, 0_2, 1_3 \rangle, 1_1, 0_2, 0_3 \rangle, 0_1, 1_2, 1_3 \rangle \}$
(* Marking: 1 on A --- 1 on B *) QuantumEvaluate [C ⁽³⁾ [Z ₂] · C ⁽¹⁾ [Z ₃] · Z ₃ · #] & /@ A QuantumEvaluate [C ⁽³⁾ [Z ₂] · C ⁽¹⁾ [Z ₃] · Z ₃ · #] & /@ B
$\{ 0_1, 0_2, 0_3 \rangle, - 0_1, 0_2, 1_3 \rangle, 0_1, 1_2, 0_3 \rangle, 0_1, 1_2, 1_3 \rangle, 1_1, 0_2, 0_3 \rangle \}$
$\{- 0_1, 0_2, 1_3 \rangle, 0_1, 0_2, 0_3 \rangle, 1_1, 0_2, 1_3 \rangle, 1_1, 0_2, 0_3 \rangle, 0_1, 1_2, 1_3 \rangle \}$

(* Marking: 2 on A --- 5 on B *)

QuantumEvaluate [$\chi_3 \cdot C^{(2)}[Z_3] \cdot \chi_3 \cdot C^{(1)}[Z_3] \cdot \#$] &/@ A

QuantumEvaluate [$\chi_3 \cdot C^{(2)}[Z_3] \cdot \chi_3 \cdot C^{(1)}[Z_3] \cdot \#$] &/@ B

{ | $\theta_1, \theta_2, \theta_3$ ⟩, | $\theta_1, \theta_2, 1_3$ ⟩, - | $\theta_1, 1_2, \theta_3$ ⟩, | $\theta_1, 1_2, 1_3$ ⟩, | $1_1, \theta_2, \theta_3$ ⟩}

{ | $\theta_1, \theta_2, 1_3$ ⟩, | $\theta_1, \theta_2, \theta_3$ ⟩, - | $1_1, \theta_2, 1_3$ ⟩, | $1_1, \theta_2, \theta_3$ ⟩, | $\theta_1, 1_2, 1_3$ ⟩}

(* Marking: 3 on A --- 3 on B *)

QuantumEvaluate [$C^{(2)}[Z_3] \cdot \#$] &/@ A

QuantumEvaluate [$C^{(2)}[Z_3] \cdot \#$] &/@ B

{ | $\theta_1, \theta_2, \theta_3$ ⟩, | $\theta_1, \theta_2, 1_3$ ⟩, | $\theta_1, 1_2, \theta_3$ ⟩, - | $\theta_1, 1_2, 1_3$ ⟩, | $1_1, \theta_2, \theta_3$ ⟩}

{ | $\theta_1, \theta_2, 1_3$ ⟩, | $\theta_1, \theta_2, \theta_3$ ⟩, | $1_1, \theta_2, 1_3$ ⟩, | $1_1, \theta_2, \theta_3$ ⟩, - | $\theta_1, 1_2, 1_3$ ⟩}

(* Marking: 4 on A --- 4 on B *)

QuantumEvaluate [$C^{(1)}[Z_3] \cdot Z_1 \cdot \#$] &/@ A

QuantumEvaluate [$C^{(1)}[Z_3] \cdot Z_1 \cdot \#$] &/@ B

{ | $\theta_1, \theta_2, \theta_3$ ⟩, | $\theta_1, \theta_2, 1_3$ ⟩, | $\theta_1, 1_2, \theta_3$ ⟩, | $\theta_1, 1_2, 1_3$ ⟩, - | $1_1, \theta_2, \theta_3$ ⟩}

{ | $\theta_1, \theta_2, 1_3$ ⟩, | $\theta_1, \theta_2, \theta_3$ ⟩, | $1_1, \theta_2, 1_3$ ⟩, - | $1_1, \theta_2, \theta_3$ ⟩, | $\theta_1, 1_2, 1_3$ ⟩}

B.5 SIMON'S ALGORITHM WITH USELESS ORACLES

Simon's algorithm with the useless oracle defined in [Definition 6.3](#).

Needs["Quantum`Computing` "];
SetComputingAliases [];
Clear[UselessSimonOracle , SimonAlgorithm]

Useless Simon oracle

a = period
nq = number of qubits

```

UselessSimonOracle [a_, n_] := Module[
{B, k, 01, 02, C, 0, H, X, r},
r =
  # + n & /@ Flatten[Position[Reverse[IntegerDigits[RandomInteger[2^n - 1], 2, n]], 1]];
If[Length[r] > 0, X = X_r, X = 1];
On[Assert];
If[a > 0, Assert[Ceiling[Log2[a]] ≤ n],

  Return[X ·  $\bigotimes_{j=1}^n C^{(j)}[NOT_{j+n}]$ ]]

];
B = Flatten[Position[Reverse[IntegerDigits[a, 2, n]], 1]];
k = B[[1]];
C = Delete[Range[1, n], k];

If[Length[B] == 1, 01 = I_k, 01 =  $\bigotimes_{j=2}^{\text{Length}[B]} C^{(k)}[NOT_{B[[j]]}]$ ];

02 =  $\bigotimes_{j=1}^{n-1} C^{(C[[j]])}[NOT_{C[[j]]+n}]$ ;

H = H_{k+n};
Return[ExpandAll[X · H · 01 · 02 · 01]];
]

```

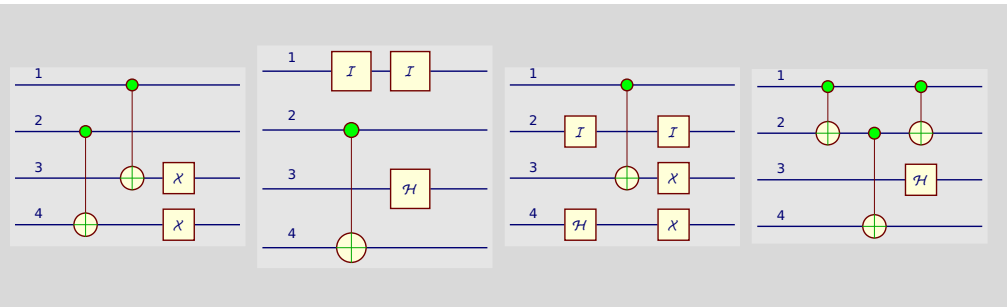
The following measurement outcomes show for all possible periods $a \in \{0, 1\}^n$.

```

n = 2;
Print["Useless Simon oracles for ", n, " qubits"]
Partition[
  QuantumPlot [UselessSimonOracle [#, n]] & /@ Range[0, 2^n - 1], UpTo[n + 2]] // Grid
Partition[{QuantumEvaluate [SimonAlgorithm [#, n]], #} & /@ Range[0, 2^n - 1], UpTo[1]] //
  Grid

```

Useless Simon oracles for 2 qubits



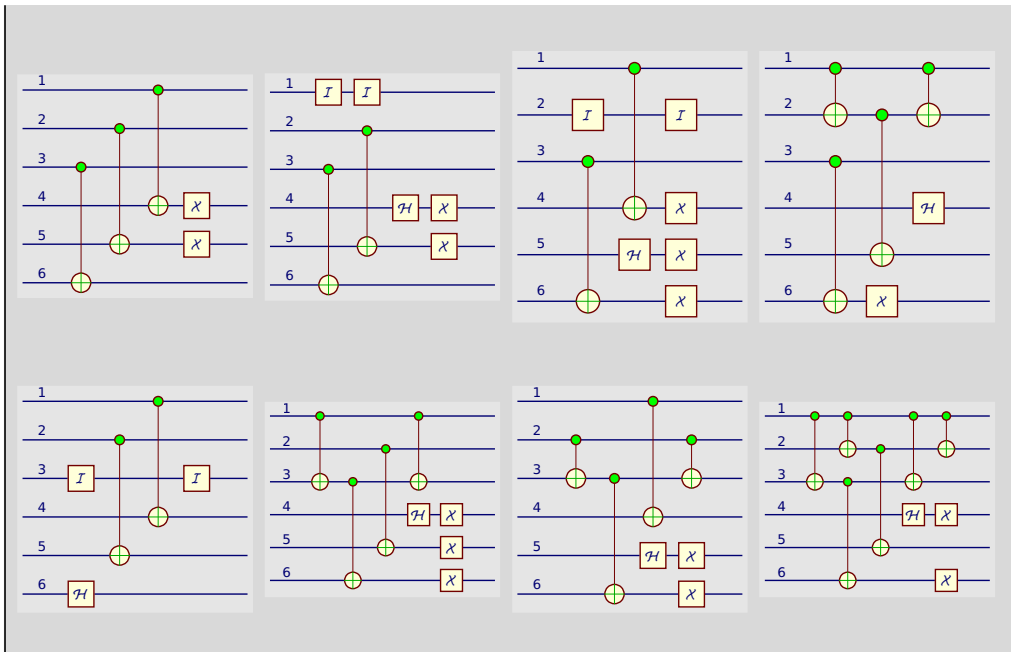
Probability	Measurement	State	
$\frac{1}{4}$	$\{\{0_1, 0_2\}\}$	$ 0_1\rangle \otimes 0_2\rangle \otimes (0_3\rangle + 1_3\rangle) \otimes (\frac{1}{2} 0_4\rangle + \frac{1}{2} 1_4\rangle)$	, 0}
$\frac{1}{4}$	$\{\{0_1, 1_2\}\}$	$ 0_1\rangle \otimes 1_2\rangle \otimes (0_3\rangle + 1_3\rangle) \otimes (-\frac{1}{2} 0_4\rangle + \frac{1}{2} 1_4\rangle)$	
$\frac{1}{4}$	$\{\{1_1, 0_2\}\}$	$ 1_1\rangle \otimes 0_2\rangle \otimes (0_3\rangle - 1_3\rangle) \otimes (\frac{1}{2} 0_4\rangle + \frac{1}{2} 1_4\rangle)$	
$\frac{1}{4}$	$\{\{1_1, 1_2\}\}$	$ 1_1\rangle \otimes 1_2\rangle \otimes (0_3\rangle - 1_3\rangle) \otimes (-\frac{1}{2} 0_4\rangle + \frac{1}{2} 1_4\rangle)$	
Probability	Measurement	State	
Probability	Measurement	State	
$\frac{1}{2}$	$\{\{0_1, 0_2\}\}$	$ 0_1\rangle \otimes 0_2\rangle \otimes (0_3\rangle + 1_3\rangle) \otimes (\frac{1}{2} 0_4\rangle + \frac{1}{2} 1_4\rangle)$	, 1}
$\frac{1}{2}$	$\{\{0_1, 1_2\}\}$	$ 0_1\rangle \otimes 1_2\rangle \otimes (0_3\rangle + 1_3\rangle) \otimes (-\frac{1}{2} 0_4\rangle + \frac{1}{2} 1_4\rangle)$	
Probability	Measurement	State	
Probability	Measurement	State	
$\frac{1}{2}$	$\{\{0_1, 0_2\}\}$	$ 0_1\rangle \otimes 0_2\rangle \otimes (0_3\rangle + 1_3\rangle) \otimes (\frac{1}{2} 0_4\rangle + \frac{1}{2} 1_4\rangle)$	, 2}
$\frac{1}{2}$	$\{\{1_1, 0_2\}\}$	$ 1_1\rangle \otimes 0_2\rangle \otimes (0_3\rangle - 1_3\rangle) \otimes (\frac{1}{2} 0_4\rangle + \frac{1}{2} 1_4\rangle)$	
Probability	Measurement	State	
Probability	Measurement	State	
$\frac{1}{2}$	$\{\{0_1, 0_2\}\}$	$ 0_1\rangle \otimes 0_2\rangle \otimes (0_3\rangle + 1_3\rangle) \otimes (\frac{1}{2} 0_4\rangle + \frac{1}{2} 1_4\rangle)$	, 3}
$\frac{1}{2}$	$\{\{1_1, 1_2\}\}$	$ 1_1\rangle \otimes 1_2\rangle \otimes (0_3\rangle + 1_3\rangle) \otimes (\frac{1}{2} 0_4\rangle - \frac{1}{2} 1_4\rangle)$	
Probability	Measurement	State	

```

n = 3;
Print["Useless Simon oracles for ", n, " qubits"]
Partition[QuantumPlot[UselessSimonOracle[#, n]] &/@ Range[0, 2^n - 1], UpTo[4]] // Grid

```

Useless Simon oracles for 3 qubits



BIBLIOGRAPHY

- [1] S. Barz et al. “Demonstration of Blind Quantum Computing”. In: *Science* 335.6066 (2012), pp. 303–308. DOI: [10.1126/science.1214707](https://doi.org/10.1126/science.1214707).
- [2] P. Benioff. “The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines”. In: *Journal of statistical physics* 22.5 (1980), pp. 563–591. DOI: [10.1007/BF01011339](https://doi.org/10.1007/BF01011339).
- [3] P. Benioff. “Quantum mechanical models of Turing machines that dissipate no energy”. In: *Physical Review Letters* 48.23 (1982), p. 1581. DOI: [10.1103/PhysRevLett.48.1581](https://doi.org/10.1103/PhysRevLett.48.1581).
- [4] P. Benioff. “Quantum mechanical Hamiltonian models of Turing machines”. In: *Journal of Statistical Physics* 29.3 (1982), pp. 515–546. DOI: [10.1007/BF01342185](https://doi.org/10.1007/BF01342185).
- [5] C. H. Bennett. “Logical reversibility of computation”. In: *IBM journal of Research and Development* 17.6 (1973), pp. 525–532. DOI: [10.1147/rd.176.0525](https://doi.org/10.1147/rd.176.0525).
- [6] R. P. Feynman. “Simulating Physics with Computers”. In: *International Journal of Theoretical Physics* 21.6/7 (1982).
- [7] R. P. Feynman. “Quantum mechanical computers”. In: *Optics news* 11.2 (1985), pp. 11–20.
- [8] D. Deutsch. “Quantum theory, the Church–Turing principle and the universal quantum computer”. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (1985), pp. 97–117. DOI: <https://doi.org/10.1098/rspa.1985.0070>.
- [9] D. Deutsch and R. Jozsa. “Rapid solution of problems by quantum computation”. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558. DOI: <https://doi.org/10.1098/rspa.1992.0167>.
- [10] A. Peres. “Reversible logic and quantum computers”. In: *Physical review A* 32.6 (1985), p. 3266. DOI: [10.1103/PhysRevA.32.3266](https://doi.org/10.1103/PhysRevA.32.3266).

- [11] C. H. Bennett and G. Brassard. “Quantum cryptography: Public key distribution and coin tossing”. In: *Proc. of IEEE Int. Conf. on Comp., Syst. and Signal Proc., Bangalore, India, Dec. 10-12, 1984*. 1984. DOI: [10.1016/j.tcs.2014.05.025](https://doi.org/10.1016/j.tcs.2014.05.025).
- [12] A. K. Ekert. “Quantum cryptography based on Bell’s theorem”. In: *Physical review letters* 67.6 (1991), p. 661. DOI: <https://doi.org/10.1103/PhysRevLett.67.661>.
- [13] C. H. Bennett et al. “Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels”. In: *Physical review letters* 70.13 (1993), p. 1895. DOI: [10.1103/PhysRevLett.70.1895](https://doi.org/10.1103/PhysRevLett.70.1895).
- [14] A. C.-C. Yao. “Quantum circuit complexity”. In: *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*. IEEE. 1993, pp. 352–361. DOI: [10.1109/SFCS.1993.366852](https://doi.org/10.1109/SFCS.1993.366852).
- [15] P. W. Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134. DOI: [10.1109/SFCS.1994.365700](https://doi.org/10.1109/SFCS.1994.365700).
- [16] D. P. DiVincenzo. “Two-bit gates are universal for quantum computation”. In: *Physical Review A* 51.2 (1995), p. 1015. DOI: [10.1103/PhysRevA.51.1015](https://doi.org/10.1103/PhysRevA.51.1015).
- [17] B. Schumacher. “Quantum coding”. In: *Physical Review A* 51.4 (1995), p. 2738. DOI: [10.1103/PhysRevA.51.2738](https://doi.org/10.1103/PhysRevA.51.2738).
- [18] A. Barenco et al. “Elementary gates for quantum computation”. In: *Physical review A* 52.5 (1995), p. 3457. DOI: [10.1103/PhysRevA.52.3457](https://doi.org/10.1103/PhysRevA.52.3457).
- [19] L. K. Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM. 1996, pp. 212–219. DOI: [10.1145/237814.237866](https://doi.org/10.1145/237814.237866).
- [20] E. Bernstein and U. Vazirani. “Quantum complexity theory”. In: *SIAM Journal on computing* 26.5 (1997), pp. 1411–1473. DOI: <https://doi.org/10.1137/S0097539796300921>.
- [21] S. Wiesner. “Conjugate Coding”. In: *SIGACT News* 15.1 (Jan. 1983), pp. 78–88. DOI: [10.1145/1008908.1008920](https://doi.org/10.1145/1008908.1008920).
- [22] M. Hillery, V. Bužek, and A. Berthiaume. “Quantum secret sharing”. In: *Physical Review A* 59.3 (1999), p. 1829. DOI: [10.1103/PhysRevA.69.052307](https://doi.org/10.1103/PhysRevA.69.052307).
- [23] D. Mayers. “Unconditional security in quantum cryptography”. In: *Journal of the ACM (JACM)* 48.3 (2001), pp. 351–406. DOI: [10.1145/382780.382781](https://doi.org/10.1145/382780.382781).

- [24] Y. Zhao et al. “Quantum hacking: Experimental demonstration of time-shift attack against practical quantum-key-distribution systems”. In: *Physical Review A* 78.4 (2008), p. 042333. DOI: [10.1103/PhysRevA.78.042333](https://doi.org/10.1103/PhysRevA.78.042333).
- [25] A. Broadbent, J. Fitzsimons, and E. Kashefi. “Universal blind quantum computation”. In: *Foundations of Computer Science, 2009. FOCS’09. 50th Annual IEEE Symposium on*. IEEE. 2009, pp. 517–526. DOI: [10.1109/FOCS.2009.36](https://doi.org/10.1109/FOCS.2009.36).
- [26] D. J. Bernstein and T. Lange. “Post-quantum cryptography”. In: *Nature* 549.7671 (2017), pp. 188–194. DOI: [10.1038/nature23461](https://doi.org/10.1038/nature23461).
- [27] J. I. Cirac et al. “Quantum state transfer and entanglement distribution among distant nodes in a quantum network”. In: *Physical Review Letters* 78.16 (1997), p. 3221. DOI: [10.1103/PhysRevLett.78.3221](https://doi.org/10.1103/PhysRevLett.78.3221).
- [28] R. Reichle et al. “Experimental purification of two-atom entanglement”. In: *Nature* 443.7113 (2006), pp. 838–841. DOI: [10.1038/nature05146](https://doi.org/10.1038/nature05146).
- [29] H. J. Kimble. “The quantum internet”. In: *Nature* 453.7198 (2008), pp. 1023–1030. DOI: [10.1038/nature07127](https://doi.org/10.1038/nature07127).
- [30] L. Jiang et al. “Quantum repeater with encoding”. In: *Physical Review A* 79.3 (2009), p. 032325. DOI: [10.1103/PhysRevA.79.032325](https://doi.org/10.1103/PhysRevA.79.032325).
- [31] H. De Riedmatten et al. “Long distance quantum teleportation in a quantum relay configuration”. In: *Physical Review Letters* 92.4 (2004), p. 047904. DOI: [10.1103/PhysRevLett.92.047904](https://doi.org/10.1103/PhysRevLett.92.047904).
- [32] S. Abramsky and B. Coecke. “Categorical quantum mechanics”. In: *Handbook of quantum logic and quantum structures* 2 (2009), pp. 261–325.
- [33] B. Coecke and R. Duncan. “Interacting quantum observables: categorical algebra and diagrammatics”. In: *New Journal of Physics* 13.4 (2011), p. 043016. DOI: [10.1088/1367-2630/13/4/043016](https://doi.org/10.1088/1367-2630/13/4/043016).
- [34] J. D. Biamonte, S. R. Clark, and D. Jaksch. “Categorical tensor network states”. In: *AIP Advances* 1.4 (2011), p. 042172. DOI: [10.1063/1.3672009](https://doi.org/10.1063/1.3672009).
- [35] S. Lloyd, M. Mohseni, and P. Rebentrost. “Quantum algorithms for supervised and unsupervised machine learning”. In: *arXiv preprint arXiv:1307.0411* (2013).
- [36] M. Schuld, I. Sinayskiy, and F. Petruccione. “An introduction to quantum machine learning”. In: *Contemporary Physics* 56.2 (2015), pp. 172–185.
- [37] J. Biamonte et al. “Quantum machine learning”. In: *Nature* 549.7671 (2017), pp. 195–202. DOI: [10.1038/nature23474](https://doi.org/10.1038/nature23474).
- [38] S. Jordan. *Quantum Algorithm Zoo*. Accessed: 2018-06-25. 2011–. URL: <https://math.nist.gov/quantum/zoo/#oracular>.

- [39] S. Barz et al. “Experimental verification of quantum computation”. In: *Nature physics* 9.11 (2013), pp. 727–731. DOI: [10.1038/nphys2763](https://doi.org/10.1038/nphys2763).
- [40] T. Baker, J. Gill, and R. Solovay. “Relativizations of the P=?NP question”. In: *SIAM Journal on computing* 4.4 (1975), pp. 431–442. DOI: [10.1137/0204037](https://doi.org/10.1137/0204037).
- [41] D. R. Simon. “On the Power of Quantum Computation”. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. SFCS ’94. USA: IEEE Computer Society, 1994, pp. 116–123. ISBN: 0818665807. DOI: [10.1109/SFCS.1994.365701](https://doi.org/10.1109/SFCS.1994.365701). URL: <https://doi.org/10.1109/SFCS.1994.365701>.
- [42] R. Raz and A. Tal. “Oracle Separation of BQP and PH”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, 2019, pp. 13–23. ISBN: 9781450367059. DOI: [10.1145/3313276.3316315](https://doi.org/10.1145/3313276.3316315). URL: <https://doi.org/10.1145/3313276.3316315>.
- [43] A. W. Harrow and D. J. Rosenbaum. “Uselessness for an Oracle Model with Internal Randomness”. In: *arXiv preprint arXiv:1111.1462* (2011).
- [44] F. Shi et al. “Room-temperature implementation of the Deutsch-Jozsa algorithm with a single electronic spin in diamond”. In: *Physical review letters* 105.4 (2010), p. 040504. DOI: [10.1103/PhysRevLett.105.040504](https://doi.org/10.1103/PhysRevLett.105.040504).
- [45] T. Taminiau et al. “Detection and control of individual nuclear spins using a weakly coupled electron spin”. In: *Physical review letters* 109.13 (2012), p. 137602. DOI: [10.1103/PhysRevLett.109.137602](https://doi.org/10.1103/PhysRevLett.109.137602).
- [46] T. H. Taminiau et al. “Universal control and error correction in multi-qubit spin registers in diamond”. In: *Nature nanotechnology* 9.3 (2014), p. 171. DOI: [10.1038/nnano.2014.2](https://doi.org/10.1038/nnano.2014.2).
- [47] J. Cramer et al. “Repeated quantum error correction on a continuously encoded qubit by real-time feedback”. In: *Nature communications* 7 (2016), p. 11526. DOI: [10.1038/ncomms11526](https://doi.org/10.1038/ncomms11526).
- [48] A. Cooper et al. “Identification and Control of Electron-Nuclear Spin Defects in Diamond”. In: *Physical Review Letters* 124.8 (2020), p. 083602. DOI: <https://doi.org/10.1103/PhysRevLett.124.083602>.
- [49] F. Dolde et al. “Room-temperature entanglement between single defect spins in diamond”. In: *Nature Physics* 9.3 (2013), p. 139. DOI: [10.1126/science.1220513](https://doi.org/10.1126/science.1220513).
- [50] H. Bernien et al. “Heralded entanglement between solid-state qubits separated by three metres”. In: *Nature* 497.7447 (2013), p. 86. DOI: [10.1038/nature12016](https://doi.org/10.1038/nature12016).

- [51] W. Pfaff et al. “Unconditional quantum teleportation between distant solid-state quantum bits”. In: *Science* 345.6196 (2014), pp. 532–535. DOI: [10.1126/science.1253512](https://doi.org/10.1126/science.1253512).
- [52] B. Hensen et al. “Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres”. In: *Nature* 526.7575 (2015), p. 682. DOI: [10.1038/nature15759](https://doi.org/10.1038/nature15759).
- [53] D. A. Meyer and J. Pommersheim. “On the uselessness of quantum queries”. In: *Theoretical Computer Science* 412.51 (2011), pp. 7068–7074. DOI: [10.1016/j.tcs.2011.06.037](https://doi.org/10.1016/j.tcs.2011.06.037).
- [54] M. Houshmand, M. Houshmand, and J. F. Fitzsimons. “Minimal qubit resources for the realization of measurement-based quantum computation”. In: *Phys. Rev. A* 98 (1 2018), p. 012318. DOI: [10.1103/PhysRevA.98.012318](https://doi.org/10.1103/PhysRevA.98.012318).
- [55] D. P. DiVincenzo and J. Smolin. “Results on two-bit gate design for quantum computers”. In: *Proceedings of the Workshop on Physics and Computation, PhysComp '94*. eprint arXiv: cond-mat/9409111. IEEE Press, 1994, p. 14. DOI: [10.1109/PHYCMP.1994.363704](https://doi.org/10.1109/PHYCMP.1994.363704).
- [56] R. Raussendorf and H. J. Briegel. “A One-Way Quantum Computer”. In: *Phys. Rev. Lett.* 86 (22 2001), pp. 5188–5191. DOI: [10.1103/PhysRevLett.86.5188](https://doi.org/10.1103/PhysRevLett.86.5188). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.86.5188>.
- [57] R. Raussendorf, D. E. Browne, and H. J. Briegel. “Measurement-based quantum computation on cluster states”. In: *Phys. Rev. A* 68 (2 2003), p. 022312. DOI: [10.1103/PhysRevA.68.022312](https://doi.org/10.1103/PhysRevA.68.022312). URL: <https://link.aps.org/doi/10.1103/PhysRevA.68.022312>.
- [58] M. A. Nielsen. “Quantum computation by measurement and quantum memory”. In: *Physics Letters A* 308.2 (2003), pp. 96–100. DOI: [10.1016/S0375-9601\(02\)01803-0](https://doi.org/10.1016/S0375-9601(02)01803-0).
- [59] D. Gottesman and I. L. Chuang. “Quantum teleportation is a universal computational primitive”. In: *arXiv preprint quant-ph/9908010* (1999).
- [60] S. A. Fenner and Y. Zhang. “Universal quantum computation with two-and three-qubit projective measurements”. In: *arXiv preprint quant-ph/0111077* (2001).
- [61] D. W. Leung. “Quantum computation by measurements”. In: *International Journal of Quantum Information* 2.01 (2004), pp. 33–43.

- [62] P. Aliferis and D. W. Leung. “Computation by measurements: A unifying picture”. In: *Phys. Rev. A* 70 (6 2004), p. 062314. DOI: [10.1103/PhysRevA.70.062314](https://doi.org/10.1103/PhysRevA.70.062314). URL: <https://link.aps.org/doi/10.1103/PhysRevA.70.062314>.
- [63] R. Jozsa. “An introduction to measurement based quantum computation”. In: *NATO Science Series, III: Computer and Systems Sciences. Quantum Information Processing-From Theory to Experiment* 199 (2006), pp. 137–158.
- [64] R. Raussendorf. *Robert Raussendorf*. 2020. URL: <https://www.phas.ubc.ca/~raussen/> (visited on 02/12/2020).
- [65] R. Raussendorf. “Measurement-based quantum computation with cluster states”. PhD thesis. Ludwig-Maximilians-Universität München, July 2003. URL: <http://nbn-resolving.de/urn:nbn:de:bvb:19-13674>.
- [66] D. Jaksch et al. “Entanglement of Atoms via Cold Controlled Collisions”. In: *Phys. Rev. Lett.* 82 (9 1999), pp. 1975–1978. DOI: [10.1103/PhysRevLett.82.1975](https://doi.org/10.1103/PhysRevLett.82.1975). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.82.1975>.
- [67] H. J. Briegel and R. Raussendorf. “Persistent Entanglement in Arrays of Interacting Particles”. In: *Phys. Rev. Lett.* 86 (5 2001), pp. 910–913. DOI: [10.1103/PhysRevLett.86.910](https://doi.org/10.1103/PhysRevLett.86.910). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.86.910>.
- [68] X Zhou, D. W. Leung, and I. L. Chuang. *Quantum logic gate constructions with one-bit "teleportation"*. Tech. rep. quant-ph/0002039. 2000. URL: <http://cds.cern.ch/record/426912>.
- [69] L. Euler. *Novi Commentarii academiae scientiarum Petropolitanae* 20. 1776, pp. 189–207.
- [70] V. Danos, E. Kashefi, and P. Panangaden. “The measurement calculus”. In: *Journal of the ACM (JACM)* 54.2 (2007), p. 8. DOI: [10.1145/1219092.1219096](https://doi.org/10.1145/1219092.1219096).
- [71] V. Danos and E. Kashefi. “Determinism in the one-way model”. In: *Physical Review A* 74.5 (2006), p. 052310. DOI: [10.1103/PhysRevA.74.052310](https://doi.org/10.1103/PhysRevA.74.052310).
- [72] W. T. Tutte. *Graph Theory Volume 21 of Encyclopedia of mathematics and its applications Encyclopedia of mathematics and its applications: Section, Combinatorics*. illustrated. Vol. 21. 1. Addison-Wesley Publishing Company, 1984. ISBN: 0201135205.
- [73] J. A. Bondy, U. S. R. Murty, et al. *Graph theory with applications*. Vol. 290. Macmillan London, 1976.

- [74] D. Gottesman. “Stabilizer codes and quantum error correction”. In: *arXiv preprint quant-ph/9705052* (1997).
- [75] S. Aaronson and D. Gottesman. “Improved simulation of stabilizer circuits”. In: *Physical Review A* 70.5 (2004), p. 052328. DOI: <https://doi.org/10.1103/PhysRevA.70.052328>.
- [76] D. Grier and L. Schaeffer. “The classification of stabilizer operations over qubits”. In: *arXiv preprint arXiv:1603.03999* (2016).
- [77] M. Van den Nest, J. Dehaene, and B. De Moor. “Graphical description of the action of local Clifford transformations on graph states”. In: *Phys. Rev. A* 69 (2 2004), p. 022316. DOI: [10.1103/PhysRevA.69.022316](https://link.aps.org/doi/10.1103/PhysRevA.69.022316). URL: <https://link.aps.org/doi/10.1103/PhysRevA.69.022316>.
- [78] M. Hein, J. Eisert, and H. J. Briegel. “Multiparty entanglement in graph states”. In: *Phys. Rev. A* 69 (6 2004), p. 062311. DOI: [10.1103/PhysRevA.69.062311](https://link.aps.org/doi/10.1103/PhysRevA.69.062311). URL: <https://link.aps.org/doi/10.1103/PhysRevA.69.062311>.
- [79] M. Hein et al. “Entanglement in graph states and its applications”. In: *arXiv preprint quant-ph/0602096* (2006).
- [80] A. Bouchet. “Recognizing locally equivalent graphs”. In: *Discrete Mathematics* 114.1-3 (1993), pp. 75–86. DOI: [10.1016/0012-365X\(93\)90357-Y](https://doi.org/10.1016/0012-365X(93)90357-Y).
- [81] J. C. Adcock et al. “Mapping graph state orbits under local complementation”. In: *arXiv preprint arXiv:1910.03969* (2019).
- [82] R. Raussendorf, D. E. Browne, and H. J. Briegel. “Measurement-based quantum computation on cluster states”. In: *Physical review A* 68.2 (2003), p. 022312. DOI: [10.1103/PhysRevA.68.022312](https://doi.org/10.1103/PhysRevA.68.022312).
- [83] A. M. Childs, D. W. Leung, and M. A. Nielsen. “Unified derivations of measurement-based schemes for quantum computation”. In: *Phys. Rev. A* 71 (3 2005), p. 032318. DOI: [10.1103/PhysRevA.71.032318](https://link.aps.org/doi/10.1103/PhysRevA.71.032318). URL: <https://link.aps.org/doi/10.1103/PhysRevA.71.032318>.
- [84] M. Mhalla and S. Perdrix. “Finding optimal flows efficiently”. In: *International Colloquium on Automata, Languages, and Programming*. Springer, 2008, pp. 857–868. DOI: [10.1007/978-3-540-70575-8_70](https://doi.org/10.1007/978-3-540-70575-8_70).
- [85] N. de Beaudrap. “Finding flows in the one-way measurement model”. In: *Phys. Rev. A* 77 (2 2008), p. 022328. DOI: [10.1103/PhysRevA.77.022328](https://link.aps.org/doi/10.1103/PhysRevA.77.022328). URL: <https://link.aps.org/doi/10.1103/PhysRevA.77.022328>.
- [86] D. E. Browne et al. “Generalized flow and determinism in measurement-based quantum computation”. In: *New Journal of Physics* 9.8 (2007), p. 250. DOI: [doi:10.1088/1367-2630/9/8/250](https://doi.org/10.1088/1367-2630/9/8/250).

- [87] D. Watt. *Programming Language Design Concepts*. Wiley, 2004. ISBN: 9780470853207. URL: <https://books.google.de/books?id=KIYhAQAAIAAJ>.
- [88] J. F. Fitzsimons and E. Kashefi. “Unconditionally verifiable blind quantum computation”. In: *Physical Review A* 96.1 (2017), p. 012303. DOI: [10.1103/PhysRevA.96.012303](https://doi.org/10.1103/PhysRevA.96.012303).
- [89] N. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1986, p. 10.
- [90] A. Syropoulos. “Mathematics of multisets”. In: *Workshop on Membrane Computing*. Springer. 2000, pp. 347–358.
- [91] IBM. *IBM Quantum Experience*. Accessed: 2020-03-19. URL: <https://quantum-computing.ibm.com/>.
- [92] Rigetti. *Rigetti Think quantum*. Accessed: 2020-03-19. URL: <https://rigetti.com/>.
- [93] U. of Bristol. *Quantum in the Cloud*. Accessed: 2020-03-19. URL: <https://cnotmz.appspot.com/>.
- [94] A. M. Childs. “Secure assisted quantum computation”. In: *arXiv preprint quant-ph/0111046* (2001). DOI: [10.26421/QIC5.6](https://doi.org/10.26421/QIC5.6).
- [95] V. Dunjko and E. Kashefi. “Blind quantum computing with two almost identical states”. In: *arXiv preprint arXiv:1604.01586* (2016).
- [96] T. Morimae, V. Dunjko, and E. Kashefi. “Ground state blind quantum computation on AKLT state”. In: *Quantum information & computation* 15.3-4 (2015), pp. 200–234.
- [97] M. Hayashi and T. Morimae. “Verifiable measurement-only blind quantum computing with stabilizer testing”. In: *Physical review letters* 115.22 (2015), p. 220502. DOI: <https://doi.org/10.1103/PhysRevLett.115.220502>.
- [98] V. Giovannetti et al. “Efficient universal blind quantum computation”. In: *Physical review letters* 111.23 (2013), p. 230501. DOI: <https://doi.org/10.1103/PhysRevLett.111.230501>.
- [99] T. Sueki, T. Koshihara, and T. Morimae. “Ancilla-driven universal blind quantum computation”. In: *Physical Review A* 87.6 (2013), p. 060301. DOI: <https://doi.org/10.1103/PhysRevA.87.060301>.
- [100] T. Morimae and K. Fujii. “Secure entanglement distillation for double-server blind quantum computation”. In: *Physical review letters* 111.2 (2013), p. 020502. URL: <https://doi.org/10.1103/PhysRevLett.111.020502>.
- [101] T. Morimae and K. Fujii. “Blind quantum computation protocol in which Alice only makes measurements”. In: *Physical Review A* 87.5 (2013), p. 050301. DOI: <https://doi.org/10.1103/PhysRevA.87.050301>.

- [102] V. Dunjko, E. Kashefi, and A. Leverrier. “Blind quantum computing with weak coherent pulses”. In: *Physical review letters* 108.20 (2012), p. 200502. DOI: [10.1103/PhysRevLett.108.200502](https://doi.org/10.1103/PhysRevLett.108.200502).
- [103] T. Morimae and K. Fujii. “Blind topological measurement-based quantum computation”. In: *Nature communications* 3.1 (2012), pp. 1–6. DOI: <https://doi.org/10.1038/ncomms2043>.
- [104] T. Morimae. “Continuous-variable blind quantum computation”. In: *Physical review letters* 109.23 (2012), p. 230502. DOI: <https://doi.org/10.1103/PhysRevLett.109.230502>.
- [105] C.-Y. Lai and K.-M. Chung. “On statistically-secure quantum homomorphic encryption”. In: *arXiv preprint arXiv:1705.00139* (2017).
- [106] S.-H. Tan et al. “A quantum approach to homomorphic encryption”. In: *Scientific reports* 6.1 (2016), pp. 1–8. DOI: <https://doi.org/10.1038/srep33467>.
- [107] Y. Ouyang, S.-H. Tan, and J. F. Fitzsimons. “Quantum homomorphic encryption from quantum codes”. In: *Physical Review A* 98.4 (2018), p. 042334. DOI: <https://doi.org/10.1103/PhysRevA.98.042334>.
- [108] A. Broadbent and S. Jeffery. “Quantum homomorphic encryption for circuits of low T-gate complexity”. In: *Annual Cryptology Conference*. Springer. 2015, pp. 609–629. DOI: https://doi.org/10.1007/978-3-662-48000-7_30.
- [109] Y. Dulek, C. Schaffner, and F. Speelman. “Quantum homomorphic encryption for polynomial-sized circuits”. In: *Annual International Cryptology Conference*. Springer. 2016, pp. 3–32. DOI: https://doi.org/10.1007/978-3-662-53015-3_1.
- [110] U. Mahadev. “Classical homomorphic encryption for quantum circuits”. In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2018, pp. 332–338.
- [111] L. Yu, C. A. Pérez-Delgado, and J. F. Fitzsimons. “Limitations on information-theoretically-secure quantum homomorphic encryption”. In: *Physical Review A* 90.5 (2014), p. 050303. DOI: [10.1103/PhysRevA.90.050303](https://doi.org/10.1103/PhysRevA.90.050303).
- [112] J. Fitzsimons. “Blind and Verifiable Quantum Computation”. Quantum Games and Protocols. 2014. URL: <https://simons.berkeley.edu/talks/joe-fitzsimons-2014-02-24>.
- [113] V. Dunjko et al. “Composable security of delegated quantum computation”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2014, pp. 406–425. DOI: [10.1007/978-3-662-45608-8_22](https://doi.org/10.1007/978-3-662-45608-8_22).

- [114] C. M. Dawson and M. A. Nielsen. “The solovay-kitaev algorithm”. In: *arXiv preprint quant-ph/0505030* (2005).
- [115] A. Ambainis and A. Smith. “Small pseudo-random families of matrices: Derandomizing approximate quantum encryption”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2004, pp. 249–260. DOI: https://doi.org/10.1007/978-3-540-27821-4_23.
- [116] P. Hayden et al. “Randomizing quantum states: Constructions and applications”. In: *Communications in Mathematical Physics* 250.2 (2004), pp. 371–391. DOI: <https://doi.org/10.1007/s00220-004-1087-6>.
- [117] P. A. Dickinson and A. Nayak. “Approximate randomization of quantum states with fewer bits of key”. In: *AIP Conference Proceedings*. Vol. 864. 1. American Institute of Physics. 2006, pp. 18–36. DOI: <https://doi.org/10.1063/1.2400876>.
- [118] A. Ambainis, J. Bouda, and A. Winter. “Nonmalleable encryption of quantum information”. In: *Journal of Mathematical Physics* 50.4 (2009), p. 042106. DOI: <https://doi.org/10.1063/1.3094756>.
- [119] B. Pfitzmann and M. Waidner. “Composition and integrity preservation of secure reactive systems”. In: *Proceedings of the 7th ACM conference on Computer and communications security*. 2000, pp. 245–254. DOI: <https://doi.org/10.1145/352600.352639>.
- [120] R. Canetti. “Universally composable security: A new paradigm for cryptographic protocols”. In: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE. 2001, pp. 136–145. DOI: [10.1109/SFCS.2001.959888](https://doi.org/10.1109/SFCS.2001.959888).
- [121] M. Ben-Or and D. Mayers. “General security definition and composability for quantum & classical protocols”. In: *arXiv preprint quant-ph/0409062* (2004).
- [122] D. Unruh. “Simulatable security for quantum protocols”. In: *arXiv preprint quant-ph/0409125* (2004).
- [123] U. Maurer and R. Renner. “Abstract cryptography”. In: *IN INNOVATIONS IN COMPUTER SCIENCE*. Tsinghua University Press, 2011.
- [124] J. Watrous. “Semidefinite programs for completely bounded norms”. In: *arXiv preprint arXiv:0901.4709* (2009).
- [125] G. Chiribella, G. M. D’Ariano, and P. Perinotti. “Quantum circuit architecture”. In: *Physical review letters* 101.6 (2008), p. 060401. DOI: <https://doi.org/10.1103/PhysRevLett.101.060401>.

- [126] O. Goldreich. “On Promise Problems: A Survey”. In: *Theoretical Computer Science: Essays in Memory of Shimon Even*. Ed. by O. Goldreich, A. L. Rosenberg, and A. L. Selman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 254–290. URL: https://doi.org/10.1007/11685654_12.
- [127] P. Høyer. “Arbitrary phases in quantum amplitude amplification”. In: *Phys. Rev. A* 62 (5 2000), p. 052304. DOI: [10.1103/PhysRevA.62.052304](https://doi.org/10.1103/PhysRevA.62.052304).
- [128] S. Aaronson et al. “Complexity zoo, 2005”. In: URL <http://www.complexityzoo.com> (2014).
- [129] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. 1st. USA: Cambridge University Press, 2009. ISBN: 0521424267.
- [130] J. Watrous. “Quantum computational complexity”. In: *arXiv preprint arXiv:0804.3401* (2008).
- [131] E. Bernstein and U. Vazirani. “Quantum Complexity Theory”. In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '93. San Diego, California, USA: Association for Computing Machinery, 1993, pp. 11–20. ISBN: 0897915917. DOI: [10.1145/167088.167097](https://doi.org/10.1145/167088.167097). URL: <https://doi.org/10.1145/167088.167097>.
- [132] C. H. Bennett et al. “Strengths and weaknesses of quantum computing”. In: *SIAM journal on Computing* 26.5 (1997), pp. 1510–1523. DOI: <https://doi.org/10.1137/S0097539796300933>.
- [133] M. A. Nielsen and I. Chuang. *Quantum computation and quantum information*. 2002.
- [134] S. Even, A. L. Selman, and Y. Yacobi. “The Complexity of Promise Problems with Applications to Public-Key Cryptography”. In: *Inf. Control* 61.2 (May 1984), pp. 159–173. ISSN: 0019-9958. DOI: [10.1016/S0019-9958\(84\)80056-X](https://doi.org/10.1016/S0019-9958(84)80056-X). URL: [https://doi.org/10.1016/S0019-9958\(84\)80056-X](https://doi.org/10.1016/S0019-9958(84)80056-X).
- [135] Y. Shi. “Both Toffoli and Controlled-NOT Need Little Help to Do Universal Quantum Computing”. In: *Quantum Info. Comput.* 3.1 (Jan. 2003), pp. 84–92. ISSN: 1533-7146.
- [136] D. Aharonov. “A simple proof that Toffoli and Hadamard are quantum universal”. In: *arXiv preprint quant-ph/0301040* (2003).
- [137] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Communications of the ACM* 21.2 (1978), pp. 120–126. DOI: [10.1145/359340.359342](https://doi.org/10.1145/359340.359342).
- [138] M. Zhandry. *How to Record Quantum Queries, and Applications to Quantum Indifferentiability*. Cryptology ePrint Archive, Report 2018/276. <https://eprint.iacr.org/2018/276>. 2018.

- [139] D. A. Meyer and J. Pommersheim. “Single query learning from abelian and non-abelian Hamming distance oracles”. In: *arXiv preprint arXiv:0912.0583* (2009).
- [140] R. Cleve et al. “Quantum algorithms revisited”. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 339–354.
- [141] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990. ISBN: 0716710455.
- [142] S. Aaronson. “Guest column: NP-complete problems and physical reality”. In: *ACM Sigact News* 36.1 (2005), pp. 30–52.
- [143] C. Gustiani and D. P. DiVincenzo. “Three-qubit exact Grover within the blind oracular quantum computation scheme”. In: *arXiv preprint arXiv:1902.05534* (2019).
- [144] C. Zalka. “Grover’s quantum searching algorithm is optimal”. In: *Physical Review A* 60.4 (1999), p. 2746. DOI: [10.1103/PhysRevA.60.2746](https://doi.org/10.1103/PhysRevA.60.2746).
- [145] B. Broda. “Quantum search of a real unstructured database”. In: *The European Physical Journal Plus* 131.2 (2016), pp. 1–4. DOI: [10.1140/epjp/i2016-16038-2](https://doi.org/10.1140/epjp/i2016-16038-2).
- [146] C. Zalka. “Using Grover’s quantum algorithm for searching actual databases”. In: *Physical Review A* 62.5 (2000), p. 052305. DOI: [10.1103/PhysRevA.62.052305](https://doi.org/10.1103/PhysRevA.62.052305).
- [147] J. G. Muñoz and F. Delgado. “QUANTUM: A Wolfram Mathematica add-on for Dirac Bra-Ket Notation, Non-Commutative Algebra, and Simulation of Quantum Computing Circuits”. In: *Journal of Physics: Conference Series*. Vol. 698. 1. IOP Publishing, 2016, p. 012019. DOI: [10.1088/1742-6596/698/1/012019](https://doi.org/10.1088/1742-6596/698/1/012019).
- [148] Z. Diao. “Exactness of the original Grover search algorithm”. In: *Physical Review A* 82.4 (2010), p. 044301. DOI: [10.1103/PhysRevA.82.044301](https://doi.org/10.1103/PhysRevA.82.044301).
- [149] C. A. Trugenberger. “Quantum pattern recognition”. In: *Quantum Information Processing* 1.6 (2002), pp. 471–493. DOI: [10.1023/A:1024022632303](https://doi.org/10.1023/A:1024022632303).
- [150] S. L. Braunstein et al. “Exact quantum algorithm to distinguish Boolean functions of different weights”. In: *Journal of Physics A: Mathematical and Theoretical* 40.29 (2007), p. 8441. DOI: [10.1088/1751-8113/40/29/017](https://doi.org/10.1088/1751-8113/40/29/017).

- [151] L. K. Grover and J. Radhakrishnan. “Is Partial Quantum Search of a Database Any Easier?” In: *Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*. SPAA '05. Las Vegas, Nevada, USA: Association for Computing Machinery, 2005, 186–194. ISBN: 1581139861. DOI: [10.1145/1073970.1073997](https://doi.org/10.1145/1073970.1073997). URL: <https://doi.org/10.1145/1073970.1073997>.
- [152] X. D. WU and G. L. Long. “Verifier-based algorithm for unsorted database search problem”. In: *International Journal of Quantum Information* 5.04 (2007), pp. 597–604. DOI: [10.1142/S0219749907003067](https://doi.org/10.1142/S0219749907003067).
- [153] G.-L. Long. “Grover algorithm with zero theoretical failure rate”. In: *Physical Review A* 64.2 (2001), p. 022307. DOI: [10.1103/PhysRevA.64.022307](https://doi.org/10.1103/PhysRevA.64.022307).
- [154] Y. Liu. “An exact quantum search algorithm with arbitrary database”. In: *International Journal of Theoretical Physics* 53.8 (2014), pp. 2571–2578. DOI: [10.1007/s10773-014-2055-3](https://doi.org/10.1007/s10773-014-2055-3).
- [155] G. L. Long et al. “Phase matching in quantum searching”. In: *Physics Letters A* 262.1 (1999), pp. 27–34. DOI: [10.1016/S0375-9601\(99\)00631-3](https://doi.org/10.1016/S0375-9601(99)00631-3).
- [156] G.-L. Long, X. Li, and Y. Sun. “Phase matching condition for quantum search with a generalized initial state”. In: *Physics Letters A* 294.3-4 (2002), pp. 143–152. DOI: [10.1016/S0375-9601\(02\)00055-5](https://doi.org/10.1016/S0375-9601(02)00055-5).
- [157] D. R. Simon. “On the Power of Quantum Computation”. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. SFCS '94. USA: IEEE Computer Society, 1994, 116–123. ISBN: 0818665807. DOI: [10.1109/SFCS.1994.365701](https://doi.org/10.1109/SFCS.1994.365701). URL: <https://doi.org/10.1109/SFCS.1994.365701>.
- [158] D. R. Simon. “On the power of quantum computation”. In: *Foundations of Computer Science, IEEE Computer Society Press*. IEEE Computer Society. Los Alamitos, CA, 1994, pp. 116–123. ISBN: 978-0-8186-6580-6. DOI: <https://doi.org/10.1109/SFCS.1994.365701>.
- [159] G. Brassard, P. Hoyer, and A. Tapp. “Quantum algorithm for the collision problem”. In: *arXiv preprint quant-ph/9705002* (1997).
- [160] O. Regev and L. Schiff. “Impossibility of a quantum speed-up with a faulty oracle”. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2008, pp. 773–781.
- [161] T. Morimae and T. Koshihara. “Impossibility of perfectly-secure delegated quantum computing for classical client”. In: *arXiv preprint arXiv:1407.1636* (2014).

- [162] D. Rountree. “2 - Cryptography”. In: *Security for Microsoft Windows System Administrators*. Ed. by D. Rountree. Boston: Syngress, 2011, pp. 29–69. ISBN: 978-1-59749-594-3. DOI: <https://doi.org/10.1016/B978-1-59749-594-3.00002-8>. URL: <http://www.sciencedirect.com/science/article/pii/B9781597495943000028>.
- [163] C. H. Bennett et al. “Remote state preparation”. In: *Physical Review Letters* 87.7 (2001), p. 077902. DOI: [10.1103/PhysRevLett.87.077902](https://doi.org/10.1103/PhysRevLett.87.077902).
- [164] D. E. Deutsch. “Quantum computational networks”. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 425.1868 (1989), pp. 73–90. DOI: <https://doi.org/10.1098/rspa.1989.0099>.
- [165] W. Press et al. “Numerical Recipes (Cambridge, Univ. Press, Cambridge)”. In: (1986).
- [166] C. Gustiani. *unique2net*. Accessed: 2020-04-29. 2019–. URL: <https://github.com/cicacica/unique2net>.
- [167] V. Danos, E. Kashefi, and P. Panangaden. “Parsimonious and robust realizations of unitary maps in the one-way model”. In: *Physical Review A* 72.6 (2005), p. 064301. DOI: <https://doi.org/10.1103/PhysRevA.72.064301>.
- [168] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. Accessed: 2018-06-25. 2001–. URL: <http://www.scipy.org>.
- [169] R. Sachdeva. *Exploring remote quantum computation using Simon Algorithm within Blind Oracular Quantum Computation scheme*. Nov. 2019. URL: https://www.quantuminfo.physik.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaihqxwe.
- [170] T. Kennedy et al. “Long coherence times at 300 K for nitrogen-vacancy center spins in diamond grown by chemical vapor deposition”. In: *Applied Physics Letters* 83.20 (2003), pp. 4190–4192. DOI: [10.1063/1.1626791](https://doi.org/10.1063/1.1626791).
- [171] P. C. Maurer et al. “Room-temperature quantum bit memory exceeding one second”. In: *Science* 336.6086 (2012), pp. 1283–1286. DOI: [10.1126/science.1220513](https://doi.org/10.1126/science.1220513).
- [172] N. Bar-Gill et al. “Solid-state electronic spin coherence time approaching one second”. In: *Nature communications* 4.1 (2013), pp. 1–6. DOI: [10.1038/ncomms2771](https://doi.org/10.1038/ncomms2771).
- [173] M. H. Abobeih et al. “One-second coherence for a single electron spin coupled to a multi-qubit nuclear-spin environment”. In: *Nature communications* 9.1 (2018), pp. 1–8. DOI: <https://doi.org/10.1038/s41467-018-04916-z>.

- [174] P Neumann et al. “Multipartite entanglement among single spins in diamond”. In: *science* 320.5881 (2008), pp. 1326–1329. DOI: [10.1126/science.1157233](https://doi.org/10.1126/science.1157233).
- [175] P. C. Humphreys et al. “Deterministic delivery of remote entanglement on a quantum network”. In: *Nature* 558.7709 (2018), p. 268. DOI: [10.1038/s41586-018-0200-5](https://doi.org/10.1038/s41586-018-0200-5).
- [176] G. Davies. “Charge states of the vacancy in diamond”. In: *Nature* 269.5628 (1977), pp. 498–500. DOI: <https://doi.org/10.1038/269498a0>.
- [177] A Lenef and S. Rand. “Electronic structure of the N-V center in diamond: Theory”. In: *Physical Review B* 53.20 (1996), p. 13441. DOI: <https://doi.org/10.1103/PhysRevB.53.13441>.
- [178] N. Manson, J. Harrison, and M. Sellars. “Nitrogen-vacancy center in diamond: Model of the electronic structure and associated dynamics”. In: *Physical Review B* 74.10 (2006), p. 104303. DOI: <https://doi.org/10.1103/PhysRevB.74.104303>.
- [179] A. Gali, M. Fyta, and E. Kaxiras. “Ab initio supercell calculations on nitrogen-vacancy center in diamond: Electronic structure and hyperfine tensors”. In: *Physical Review B* 77.15 (2008), p. 155206. DOI: <https://doi.org/10.1103/PhysRevB.77.155206>.
- [180] M. W. Doherty et al. “The nitrogen-vacancy colour centre in diamond”. In: *Physics Reports* 528.1 (2013), pp. 1–45. DOI: <https://doi.org/10.1016/j.physrep.2013.02.001>.
- [181] D. M. Toyli et al. “Chip-scale nanofabrication of single spins and spin arrays in diamond”. In: *Nano letters* 10.8 (2010), pp. 3168–3172. DOI: [10.1021/nl102066q](https://doi.org/10.1021/nl102066q).
- [182] K. Ohno et al. “Engineering shallow spins in diamond with nitrogen delta-doping”. In: *Applied Physics Letters* 101.8 (2012), p. 082413. DOI: [10.1063/1.4748280](https://doi.org/10.1063/1.4748280).
- [183] J. Cramer. “Quantum error correction with spins in diamond”. PhD thesis. Delft University of Technology, Dec. 2016.
- [184] J. R. Maze et al. “Properties of nitrogen-vacancy centers in diamond: the group theoretic approach”. In: *New Journal of Physics* 13.2 (2011), p. 025025. DOI: [10.1088/1367-2630/13/2/025025](https://doi.org/10.1088/1367-2630/13/2/025025).
- [185] H. Bernien. “Control, measurement and entanglement of remote quantum spin registers in diamond”. PhD thesis. Delft University of Technology, Feb. 2014. URL: <https://repository.tudelft.nl/islandora/object/uuid%3A75130c37-edb5-4a34-ac2f-c156d377ca55>.

- [186] B. J. Hensen. “Quantum Nonlocality with Spins in Diamond”. PhD thesis. Delft University of Technology, Nov. 2016. DOI: [10.4233/uuid:b307dca0-79a4-4cc9-af91-a2927a61088e](https://doi.org/10.4233/uuid:b307dca0-79a4-4cc9-af91-a2927a61088e).
- [187] G De Lange et al. “Universal dynamical decoupling of a single solid-state spin from a spin bath”. In: *Science* 330.6000 (2010), pp. 60–63. DOI: [10.1126/science.1192739](https://doi.org/10.1126/science.1192739).
- [188] H. Y. Carr and E. M. Purcell. “Effects of diffusion on free precession in nuclear magnetic resonance experiments”. In: *Physical review* 94.3 (1954), p. 630. DOI: [10.1103/PhysRev.94.630](https://doi.org/10.1103/PhysRev.94.630).
- [189] S. D. Barrett and P. Kok. “Efficient high-fidelity quantum computation using matter qubits and linear optics”. In: *Physical Review A* 71.6 (2005), p. 060310. DOI: [10.1103/PhysRevA.71.060310](https://doi.org/10.1103/PhysRevA.71.060310).
- [190] N. Kalb et al. “Entanglement distillation between solid-state quantum network nodes”. In: *Science* 356.6341 (2017), pp. 928–932. DOI: [10.1126/science.aan0070](https://doi.org/10.1126/science.aan0070).
- [191] H.-K. Lo. “Classical-communication cost in distributed quantum-information processing: A generalization of quantum-communication complexity”. In: *Phys. Rev. A* 62 (1 2000), p. 012313. DOI: [10.1103/PhysRevA.62.012313](https://doi.org/10.1103/PhysRevA.62.012313).
- [192] V. V. Shende, I. L. Markov, and S. S. Bullock. “Minimal universal two-qubit controlled-NOT-based circuits”. In: *Physical Review A* 69.6 (2004), p. 062321. DOI: [10.1103/PhysRevA.69.062321](https://doi.org/10.1103/PhysRevA.69.062321).
- [193] M. Möttönen et al. “Quantum circuits for general multiqubit gates”. In: *Physical review letters* 93.13 (2004), p. 130502. DOI: [10.1103/PhysRevLett.93.130502](https://doi.org/10.1103/PhysRevLett.93.130502).
- [194] C.-K. Li, R. Roberts, and X. Yin. “Decomposition of unitary matrices and quantum gates”. In: *International Journal of Quantum Information* 11.01 (2013), p. 1350015. DOI: [10.1142/S0219749913500159](https://doi.org/10.1142/S0219749913500159).
- [195] A. May and L. Schlieper. “Quantum Period Finding is Compression Robust”. In: *arXiv preprint arXiv:1905.10074* (2019).
- [196] G.-Y. Xiang et al. “Remote preparation of mixed states via noisy entanglement”. In: *Physical Review A* 72.1 (2005), p. 012315. DOI: [10.1103/PhysRevA.72.012315](https://doi.org/10.1103/PhysRevA.72.012315).
- [197] C. Gustiani. *MBQC*. Accessed: 2020-07-01. 2020–. URL: <https://github.com/cicacica/MBQC>.
- [198] C. Gustiani. *cgthesis_supplement*. Accessed: 2020-07/01. 2020–. URL: https://github.com/cicacica/cgthesis_supplement.
- [199] J. L. Gómez-Muñoz and F. Delgado. *Quantum*. Accessed: 2020-07-01. URL: <http://homepage.cem.itesm.mx/lgomez/quantum/>.