UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS FÍSICAS

TESIS DOCTORAL

Nuevos Algoritmos de Caminos Cuánticos: Extensiones de Fases y el Método Semiclásico
Novel Quantum Walk Algorithms: Phase Extensions and the Semiclassical Framework

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Sergio Ángel Ortega

DIRECTOR

Miguel Ángel Martín-Delgado Alcántara

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS FÍSICAS

TESIS DOCTORAL

Nuevos Algoritmos de Caminos Cuánticos: Extensiones de Fases y el Método Semiclásico

Novel Quantum Walk Algorithms: Phase Extensions and the Semiclassical Framework

Sergio Ángel Ortega

Director

Miguel Ángel Martín-Delgado Alcántara

Doctorado en Física

Madrid, 2025

A Pili y Melli

Agradecimientos

Dicen que tres cosas hay en la vida: salud, dinero, y amor. En el caso de una tesis doctoral, que es como una vida dentro de una vida, algo similar sucede, pudiendo reinterpretar en cierto modo la expresión.

En primer lugar, tener dinero es un pilar fundamental para la realización de una tesis. En este sentido, quiero agradecer al Banco Santander junto a la UCM por su beca CT58/21-CT59/21.

Además de la financiación oficial recibida como salario, la vida en Madrid es cara, y por ello quiero agradecer también a mi mecenas particular, el Dr. Ángel, que ha hecho posible no sólo que pudiera realizar esta tesis, sino también la carrera de Física.

En segundo lugar, una buena salud en una tesis doctoral viene representada por un grupo de investigación capaz de habilitar los medios para su realización, como este precioso Macbook Air M2 de 13.6 pulgadas con 512 GB de memoria SSD y 16 GB de RAM, en el que me encuentro ahora mismo escribiendo. Quiero agradecer a mi director y tutor, Miguel Ángel, Big Boss del GICC, por acogerme bajo su tutela durante todos estos años. También agradecer a aquél pilar fundamental del GICC que mantiene sobre sus hombros todo el peso administrativo del grupo, Carmen, ya que sin ella no podría... es que no quiero ni imaginarlo. Gracias por estar ahí siempre que lo he necesitado fuese el momento que fuese.

Finalmente, y ahora viene la parte larga, estaría la amistad, porque el amor es como un ordenador cuántico tolerante a fallos, salvo por el hecho de que este último se supone que llegará algún día o no sé para qué he hecho esta tesis. He conocido mucha gente importante desde que entré en la Facultad de Facultad de Ciencias Físicas. Así pues, agradezco:

A mi director Miguel Ángel, ahora de una forma más personal, por compartir, aunque breves, buenos y divertidos momentos conmigo, y haber estado ahí cuando lo he necesitado, incluso cuando la voz le ha faltado.

A mis compañeros de Mardel Students: Roberto, Gabriel, Pablo Doctor, Pablo y Sara. Quiero agradecer especialmente a Roberto, que me introdujo en la dinámica del GICC cuando llegué, me presentó a Alcaudón y Ummon, y me ha estado apoyando con todo lo que he necesitado. También quiero agradecer a Pablo Fernández por esa colaboración conjunta que hemos tenido. Por otra parte, aunque no un miembro oficial, pero para mí es como si lo hubiera sido, quiero agradecer a Jaime Torres por la colaboración que hemos tenido, y los buenos momentos que hemos pasado con las bitcoins. Has trabajado muy bien para ser un completo novato, y ello se refleja en que esta tesis contiene una pequeña parte de ti.

A Carmen, por haber sido tan maja desde que llegué al GICC. También a Ángel Rivas, que se ha mostrado muy cercano conmigo, a pesar de que nuestros caminos investigadores estaban separados, y al resto de compañeros del GICC.

A mis compañeros de despacho: Pablo Rabán, Roberto, Gabriel, Juanjo y Néstor. Gracias por los buenos momentos, y los no tan buenos... Sobre todo, quiero agradecer especialmente a Juanjo por controlar la situación cuando era necesario. Y a Pablo Rabán quiero dedicarle unas palabras especiales, que espero recuerde cuando ya no nos veamos: hahahahahahahahahahahahahahahaha.

A Silvia Baquero, que fue la primera persona que conocí al entrar en Física y ha estado siempre ahí hasta el final.

A Jesús Huertas, que fue mi compañero (y rival) científico durante los cuatro años de carrera.

A Itsaso Fernández y Javier Grandes, con los que compartí una batalla más dura que la de Cloud y su fiesta contra Sephiroth, así como los mejores momentos desde que entré en Física.

A Cristina Martínez Pérez, quien aparte de ser la mejor profesora que he tenido, se convirtió en una buena amiga durante todos estos años.

A Paco Navarro, por su cercanía y sus buenos consejos como profesor.

A Luis Garay y Alberto Cembranos, con quienes compartí muchos buenos momentos durante el TFG y el TFM, y que me formaron para comenzar mi investigación en el departamento de Teórica.

A Felipe Llanes, por convertirse en mi referente.

Al resto del departamento de Física Teórica, por todos los buenos ratos que hemos pasado. Por hacer alguna mención, a Adrián Casado Turrión (no te vayas), por su amistad estos años. También al resto de mis compañeros de la carrera. Por mencionar algunos: Víctor Fuentes, Israel, Danielo, Josemi, Charli, Jaime, Silvia, Noelia, David, Cartone, Iván, Julián, Lachén, Rafa, Javi, Odette, Marta, etc.

A mis alumnos de Ingeniería Electrónica, que me apoyaron durante mis inicios como profesor, y estuvieron ahí al pie del cañón en los Calculus Fest. Nunca olvidaré las clases que tuve con vosotros. Especialmente, quiero agradecer a Carolina y Cristina, por su cercanía este tiempo. También quiero agradecer a los alumnos de Relatividad que decidieron quedarse a ver los problemas en lugar de irse de clase. Gracias a vosotros ahora sé cómo hablar con una pared. Os deseo que algún día podáis recuperar el habla.

A Ana Carmen, por esos momentos surrealistas que empezaron en un karaoke, pasaron por casinos y llegaron a Arandel.

A Alejandro Rubio Elías, porque sí.

Quiero, antes de acabar, hacer una mención especial a Alberto Lozano, que fue como un hermano para mí durante un periodo de tiempo que se me hizo demasiado corto.

Y ya para acabar, ahora sí, agradezco a mis padres, Pilar y José Antonio, que me han apoyado en esta travesía, y sobre todo a mi hermano Omar, que ha cuidado de mí todo este tiempo.

Contents

Abstract

Quantum computing is an emerging paradigm that leverages the principles of quantum mechanics to process information, to provide a speedup with respect to classical computers. Among the different types of quantum algorithms that exist nowadays, quantum walks, which are quantized Markov chains, have shown good performance in a wide range of domains. In particular, a quantum walk introduced by Szegedy, which can be applied to arbitrary graphs, has been applied for tasks such as classification and optimization. The aim of this thesis is the development of novel quantum walk algorithms, from two differentiate research lines: phase extensions and the semiclassical framework.

In the first results part of the thesis we extend the standard Szegedy quantum walk to incorporate complex phases, both at the level of edges with link phases, and at the level of nodes with arbitrary phase rotations (APR), giving rise to the graph-phased Szegedy walk. This phase-extended framework establishes a broader equivalence between coined and Szegedy walks, and provides new possibilities for quantum search algorithms on graphs without needing to modify the underlying structure. Furthermore, we analyze its application in a problem of interest such as the quantum PageRank, where we demonstrate that the introduction APR can highlight secondary hubs while restoring the degeneracy among residual nodes, and retains the quantum stability of the standard algorithm.

In the second results part, we introduce the semiclassical framework for quantum walks, giving rise to the semiclassical walks. These walks can be understood as classical walks where the transition matrix encodes the quantum evolution. We explore the Szegedy walk in this context. Among our results, we find that this walk is able to break the symmetry of weighted graphs when they are inhomogeneous, which is useful for the problem of ranking nodes in symmetric graphs, where the classical PageRank fails. Furthermore, we have demonstrated experimentally that the semiclassical walks can be applied on real quantum computers using the platform IBM Quantum.

As an important application of the semiclassical Szegedy walk, we explore its use in the context of the quantum SearchRank algorithm. This algorithm aims to search for marked nodes on graphs at the same time that provides a classification of them. However, its usefulness is lost as soon as the size of the graph increases. After replacing the underlying Szegedy quantum walk with a semiclassical walk, this issue is solved. Furthermore, we propose a simplification, denoted as randomized SearchRank, which corresponds to a quantum walk acting on a randomized mixed state, and maintains the same time complexity as the quantum SearchRank algorithm. As a further application, based on the randomized SearchRank, we propose the Quantum Signature Validation Algorithm (QSVA), which is a protocol for detecting fraudulent activity in blockchain networks with a quantum speedup. Our simulations results show that this algorithm is able to detect all the tampered transaction in a real Bitcoin dataset.

Finally, apart from quantum walk algorithms, in the third results part we delve with the classical simulation of the Szegedy quantum walk, which is important for research since currently there are no fault-tolerant quantum computer. We propose a memory-saving algorithm that scales as $\mathcal{O}(N^2)$ with the size N of the graph, and provide additional procedures for simulating the phase extensions and the semiclassical Szegedy walk. Moreover, we have built a classical simulator in Python called SQUWALS. We show that our simulator scales as $\mathcal{O}(N^2)$ in both time and memory resources. This package provides some high-level applications for algorithms based on the Szegedy quantum walk, as for example the quantum PageRank.

Resumen

La computación cuántica es un paradigma emergente que aprovecha los principios de la mecánica cuántica para procesar información, con el fin de proporcionar una aceleración en comparación con los ordenadores clásicos. Entre los diferentes tipos de algoritmos cuánticos que existen hoy en día, los caminos cuánticos, que son cadenas de Markov cuantizadas, han mostrado un buen rendimiento en una amplia variedad de dominios. En particular, un camino cuántico introducido por Szegedy, que puede aplicarse a grafos arbitrarios, ha sido utilizado para tareas como clasificación y optimización. El objetivo de esta tesis es el desarrollo de nuevos algoritmos de caminos cuánticos, desde dos líneas de investigación diferentes: extensiones de fases y el método semiclásico.

En la primera parte de resultados de la tesis extendemos el camino cuántico de Szegedy estándar para incorporar fases complejas, tanto a nivel de aristas mediante fases de enlace, como a nivel de nodos mediante rotaciones de fase arbitrarias (APR en inglés), dando lugar al camino de Szegedy con fases de grafo. Este marco extendido con fases establece una equivalencia más amplia entre los caminos con moneda y los caminos de Szegedy, y proporciona nuevas posibilidades para algoritmos de búsqueda cuántica en grafos sin necesidad de modificar la estructura subyacente. Además, analizamos su aplicación en un problema de interés como el PageRank cuántico, donde demostramos que la introducción de APR puede resaltar hubs secundarios mientras restablece la degeneración entre nodos residuales, y mantiene la estabilidad cuántica del algoritmo estándar.

En la segunda parte de resultados, introducimos el método semiclásico para los caminos cuánticos, dando lugar a los caminos semiclásicos. Estos caminos pueden entenderse como caminos clásicos donde la matriz de transición codifica la evolución cuántica. Exploramos el camino de Szegedy en este contexto. Entre nuestros resultados, encontramos que este camino es capaz de romper la simetría de grafos ponderados cuando son inhomogéneos, lo cual es útil para el problema de clasificación de nodos en grafos simétricos, donde el PageRank clásico falla. Además, hemos demostrado experimentalmente que los caminos semiclásicos pueden aplicarse en ordenadores cuánticos reales utilizando la plataforma IBM Quantum.

Como una aplicación importante del camino de Szegedy semiclásico, exploramos su uso en el contexto del SearchRank cuántico. Este algoritmo busca nodos marcados en grafos al mismo tiempo que proporciona una clasificación de los mismos. Sin embargo, su utilidad se pierde en cuanto el tamaño del grafo aumenta. Tras reemplazar el camino cuántico de Szegedy por un camino semiclásico, este problema se resuelve. Además, proponemos una simplificación, denominada SearchRank aleatorizado, que corresponde a un camino cuántico que actúa sobre un estado mixto aleatorizado, y mantiene la misma complejidad temporal que el SearchRank cuántico. Como una aplicación adicional, basada en el SearchRank aleatorizado, proponemos el Algoritmo Cuántico de Validación de Firmas (QSVA en inglés), que es un protocolo para detectar actividad fraudulenta en redes blockchain

con una aceleración cuántica. Nuestros resultados de simulación muestran que este algoritmo es capaz de detectar todas las transacciones manipuladas en un conjunto de datos real de Bitcoin.

Finalmente, además de los algoritmos de caminos cuánticos, en la tercera parte de resultados abordamos la simulación clásica del camino cuántico de Szegedy, lo cual es importante para la investigación dado que actualmente no existen ordenadores cuánticos tolerantes a fallos. Proponemos un algoritmo que ahorra memoria y escala como $\mathcal{O}(N^2)$ con el tamaño N del grafo, y proporcionamos procedimientos adicionales para simular las extensiones de fases y el camino semiclásico de Szegedy. Además, hemos construido un simulador clásico en Python llamado SQUWALS. Mostramos que nuestro simulador escala como $\mathcal{O}(N^2)$ tanto en tiempo como en recursos de memoria. Este paquete proporciona algunas aplicaciones de alto nivel para algoritmos basados en el camino cuántico de Szegedy, como por ejemplo el PageRank cuántico.

List of Publications

[1] S. A. Ortega and M. A. Martin-Delgado. Generalized Quantum PageRank Algorithm with Arbitrary Phase Rotations. Physical Review Research, 5:013061, 2023.

[2] S. A. Ortega and M. A. Martin-Delgado. Discrete-Time Semiclassical Szegedy Quantum Walks. Physica A, 625:129021, 2023.

[3] S. A. Ortega and M. A. Martin-Delgado. SQUWALS: A Szegedy QUantum WALks Simulator. Advanced Quantum Technologies, 7:2400022, 2024.

[4] S. A. Ortega and M. A. Martin-Delgado. Randomized SearchRank: A Semiclassical Approach to a Quantum Search Engine. Physical Review Research, 6:043014, 2024.

[5] S. A. Ortega and M. A. Martin-Delgado. Complex-Phase Extensions of the Szegedy Quantum Walk on Graphs. Physical Review A, 111:032216, 2025.

[6] J. Torres, S. A. Ortega and M. A. Martin-Delgado. A Quantum Signature Validation Algorithm for Efficient Detection of Tampered Transactions in Blockchain. arXiv:2502.15023, 2025.

Additionally, there is an article not covered in this thesis, on the topic of quantum homomorphic encryption:

[7] S. A. Ortega, P. Fernández, and M. A. Martin-Delgado. Implementing Semiclassical Szegedy Walks in Classical-Quantum Circuits for Homomorphic Encryption. arXiv:2412.01966, 2024.

Conference Contributions

- Quantum Matter 2023 (May 23-25th, 2023): Poster presentation entitled "Generalized Quantum PageRank Algorithm with Arbitrary Phase Rotations".
- 17th Granada Seminar: Machine Learning and Physics: Quantum, Classical and Applications (September 12-15th, 2023): Poster presentation entitled "Discrete-Time Semiclassical Szegedy Quantum Walks".
- First technical meeting of the Complementary Quantum Communications Plan (September 19-21st, 2023): Poster presentation entitled "Discrete-Time Semiclassical Szegedy Quantum Walks".
- VI International Workshop on Information Geometry, Quantum Mechanics and Applications (February 20-22nd, 2024): Oral talk entitled "Semiclassical Walks and Their Application to a Quantum Search Engine".
- Second technical meeting of the MadQuantum-CM project (June 10th, 2024): Oral talk entitled "Classical Simulation of Quantum Walks and Its Applications in Network Analysis".
- Attendance to the Qiskit Global Summer School on Quantum Simulations (July 18th - August 3rd, 2022) - Quantum Excellence.
- Attendance to the CISM-UniUD Joint Advanced School on Quantum Machine Learning: from Fundamentals to Applications (September 12-16th, 2022).
- Attendance to the Qiskit Global Summer School on Theory to Implementation (July 17-28th, 2023) - Quantum Excellence.

Conventions and Notation

Along this thesis we use the following conventions and notation:

- Although linear operators can be represented by matrices, both mathematical objects are strictly different. Nevertheless, in the context of quantum computing we may use both terms interchangeably.
- For network nodes, and matrix and sum subindexes, we start counting from 0. Therefore, sums of N elements go from index 0 to index $N-1$.
- In quantum circuits we start counting the qubits from 1, and we use the big-endian convention.
- Tensor product symbols may be avoided when unnecessary. Therefore, we may denote $|a\rangle \otimes |b\rangle$ simply as $|a\rangle\,|b\rangle$.
- We indicate matrix elements using subindexes. For example, A_{10} corresponds to the element $(1,0)$ of matrix A. If the matrix name has a subscript, we may use brackets. For example, $(A_2)_{10}$ is the element $(1,0)$ of matrix A_2.
- We denote the identity operator as $\mathbb{1}$ for whatever space. If needed, we indicate with a subscript the dimensions of the subspace it acts on. For example, $\mathbb{1}_2$ corresponds to a 2×2 identity matrix.
- To distinguish from the general subindex i and the complex unit, we use the character i for subindexes and the character i for the complex unit, such that $\mathrm{i} = \sqrt{-1}$.
- We use an asterisk for complex conjugation. Therefore, the complex conjugated of $z = x + \mathrm{i}y$ is $z^* = x - \mathrm{i}y$.
- For complex phases, depending on the context we may denote as phase the complex number $e^{\mathrm{i}\theta}$, or directly the angle θ.
- For Markov chains we chose the transition matrix as column-stochastic, so that the columns are normalized to add up to 1. We denote it as G, and the element G_{ji} is the probability of transitioning from state i to state j.

Data Availability and Artwork Statement

During this thesis, a Python package has been developed:

- SQUWALS: A Szegedy QUantum WALks Simulator.
  https://github.com/OrtegaSA/SQUWALS-repo.

In this repository there are example codes of the numerical simulations performed in this thesis.

All the figures displayed in this thesis are original and have been developed during this thesis using the following software packages:

- Data plots have been represented using Matplotlib [8].
- 3D histograms have been represented using Origin [9].
- Graphs have been represented using NetworkX [10].
- Quantum circuits have been represented using Quantikz [11].
- Schemes have been represented using PowerPoint [12].

Jaime Torres Arranz also authors the figures of Chapter 8.

Introduction

Chapter 1

Preface

Quantum computing is an emerging computing paradigm which started four decades ago. In a naive sense, it consists of leveraging quantum mechanics to build more powerful computers. However, it is not as simple as this, and to understand truly what quantum computing is, let us review the historical relationship between quantum mechanics and computer since. First computers were huge machines able to process the information using vacuum tubes [13], and storage the information in punched tapes [14]. The development of quantum mechanics allowed the construction of smaller and faster processors based on transistors [15,16]. It has also played an important role in the development of smaller and faster storage devices, as hard drives based on the magnetic spin, or the recent solid state drives, which use quantum tunneling [17].

In a rough sense, the more transistors a computer has, the faster it is. Moore's law, which states that the number of transistors double every two years thanks to miniaturization, has held approximately true during the 1960s [18,19]. However, the same ally that enables the miniaturization of computers, is becoming an enemy that sets a limit for Moore's law [20]. There is a point from which quantum effects interfere with the functioning of electronic devices as they are small enough. Therefore, it is expected that in a near future we will not be able to construct more powerful computers based on this technology.

The solution for the limit set by quantum mechanics, comes also from the hand of quantum mechanics, although this time not applied to the computer devices themselves, but to the information they process. Despite the fact that current computers are in some sense quantum, the information that they process is classical, and therefore they are refereed to as classical computers. A bit, which is the basic unit of information, can be in state 0 or 1, but not both at the same time. In quantum information, the quantum analog of a bit, the quantum bit, or qubit [21], can be in either each of these states, or in a quantum superposition. Therefore, a machine able to process quantum information can leverage the effects of superposition, coherence and entanglement to obtain results in a more efficient manner than classical processors [19], paving the way to the desired quantum speedup. Now, we can state that the adjective "quantum" in "quantum computing" actually refers to the kind of information is being processed, rather than to the physical processors. Of course, a quantum computer must have at its core a quantum technology in order to work with quantum information.

A precursor idea by Feynman was to use quantum computers just for simulating quantum physical

systems in a more natural way [22, 23]. Nevertheless, there is also the understandable aim of using quantum computers for solving faster usual problems of computer science, which are not related with quantum mechanics. The form a quantum computer is more powerful than a classical one does not consist of doing the same operations quicker. Instead, they use different algorithms, which based on the effects of quantum mechanics, need less operations than classical ones. To understand how this works, let us see an example for a quadratic speedup. Suppose that, on the one hand, we have a classical algorithm requiring N operations, where N is a parameter related with the size of the problem to solve. On the other hand, we have a quantum algorithm that needs $10\sqrt{N}$ operations for solving the same problem. For $N = 1$ the classical algorithm is faster, since it only needs a single operations, whereas the quantum computer needs ten. However, as the size of the problem increases, there is a point from which the quantum computer needs less operations, becoming faster than the classical computer. In this case it occurs from $N = 100$. This is so because the classical algorithm scales as $\mathcal{O}(N)$ and the quantum one as $\mathcal{O}(\sqrt{N})$, so that in the asymptotic limit the quantum algorithms requires less operations. The prefactor, which was 1 for the classical algorithm and 10 for the quantum one, plays no role in this limit. Therefore, independently of the prefactors, there is always a point from which the quantum algorithm requires less operations as long as it has a better scaling. Different examples are shown in Figure 1.1, were we can see that a square root scaling always ends up requiring less operations than a linear one for different prefactors. Note that the actual running time of the algorithm is obtained multiplying the number of operations by the time needed by the computer per operation. This only affects to the prefactor, so even if the quantum computer operates at a slower rate, there is also always a size of the problem from which it needs less time than a classical computer.

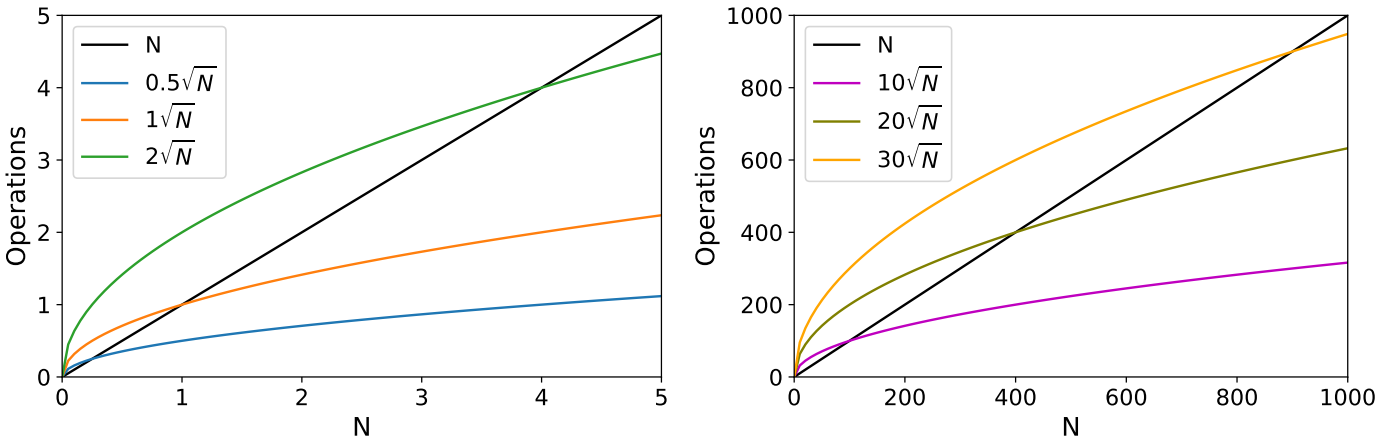


Figure 1.1: *Comparison of a linear scaling with different square root scalings. Independently of the prefactor, the square root function always ends up below the linear one.*

The first quantum algorithm that showed a quantum speedup was developed by Deutsch in 1985 [24], and was followed by the Deutsch-Jozsa algorithm in 1992 [25]. However, they were restricted to a very specific problem with a difficult real application. Two of the most important algorithms with practical applications are the Grover algorithm (1996) [26, 27], which provides a quadratic speedup in search problems, and the Shor algorithm (1994) [28, 29], which provides an exponential speedup for the discrete logarithm and integer factoring problems. This last algorithm represents a potential

break of security in modern cryptography based on the RSA protocol [30].

Although quantum algorithms are mostly devised with the aim of a quantum speedup, there are also algorithms that points to the quality of the results rather than to the speed. An example is the quantum PageRank algorithm [31], which not providing any speedup with respect to the classical algorithm, resolves better the structure of networks [32].

So far, we have only talked about quantum algorithms, but what about the physical quantum computers? Despite the development of quantum computing services as IBM Quantum in 2016 [33], and the claimed quantum supremacy demonstration by Google [34] on a very artificial and applicationless example, quantum computers are only a promising technology for the future. Nowadays, quantum computers are at a research level. Current quantum computers are very error prone, so that it is unfeasible to run quantum algorithms with real applications. Some algorithms are devised for computers allowing a certain amount of error, in what is called the noisy intermediate-scale quantum (NISQ) era [35]. Nevertheless, actual quantum advantages are expected in a more distant future when fault-tolerant quantum computers become available [36], using quantum error correction techniques. [37, 38].

It is obvious that there is still much research to do with quantum hardware. However, even if a full-fledge quantum computer becomes available suddenly, its applications would be limited since the set of quantum algorithms showing advantages is too reduced. Research in the field of quantum algorithms is very necessary, and for this reason we focus this thesis on this area. In particular, we work with quantum walks algorithms [39, 40], which are quantizations of Markov chains [41] with plenty of applications such us triangle finding [42], element distinctness [43] and quantum search [44]. Among all the possible quantizations, we focus on the one introduced by Szegedy in 2004 [45], which has applications in problems of optimization [46–50], network ranking [31, 32], graph completeness [51], and machine learning [52].

The main objectives of this thesis are the development of new quantum walk algorithms at a fundamental level, and their application to problems of interest. We can split them into two separate research lines:

1) We explore the introduction of complex-phase extensions as new degrees of freedom in the Szegedy quantum walk, which gives rise to a new family of quantum walk algorithms [5]. We analyze their properties and applications in different areas. As an important example, we use them to develop new quantum PageRank algorithm with improved properties [1].

2) We develop a new walk framework combining classical and quantum features, giving rise to what we denote as semiclassical walks [2]. We analyze their properties and apply them to cases of interest, as the quantum SearchRank algorithm [4]. This has provided an algorithm for search and ranking in networks, which we also use for detecting frauds in blockchain technologies [6].

A problem we faced during the development of this thesis is how to simulate the Szegedy quantum walk on a classical computer for big networks. To solve it, we needed to devise a novel simulation algorithm. Although not an initial objective, it has provided by itself an extra result of this thesis:

3) We devise a novel classical simulation algorithm for the Szegedy quantum walk, which has given rise to a Python library called SQUWALS implementing it [3]. This algorithm is not only useful

for the numerical simulations of this thesis, but also serves as a tool for the quantum computing community, either for the Szegedy walk simulation at a research level, or for implementing quantum-inspired classical algorithms based on it as the quantum PageRank.

The structure of this thesis is as follows. An introductory part provides all the theoretical framework needed to understand the results of this thesis:

- Chapter 2 provides an introduction to quantum computing, and sets the mathematical formulation that we use in this thesis.
- Chapter 3 provides an introduction to quantum walks, focusing on the Szegedy quantum walk.

A first part of results relates to the first objective of the thesis:

- Chapter 4 introduces the phase extensions of the Szegedy quantum walk, and show some applications.
- Chapter 5 shows a further application of the phase extensions in the context of the quantum PageRank algorithm.

A second part of results relates to the second objective of the thesis:

- Chapter 6 introduces the semiclassical framework for quantum walks, and presents the semiclassical Szegedy walk, along with applications.
- Chapter 7 shows a further application of the semiclassical Szegedy walk in the context of the quantum SearchRank.
- Chapter 8 presents a quantum algorithm for detecting fraud in blockchain based on the SearchRank algorithm within the semiclassical framework.

A third part of results relates to the third objective of the thesis:

- Chapter 9 presents SQUWALS, an efficient simulator of the Szegedy quantum walk.

Finally, we summarize and conclude in Chapter 10. Additionally, Appendix A provides some calculations about the spectral properties of the Szegedy quantum walk.

Chapter 2

Quantum Computing

In this chapter, we provide all the theoretical background about quantum computing, going from the most fundamental concepts to elaborated quantum algorithms.

As stated in the previous chapter, quantum computing is the discipline that study the use of computers for processing quantum information. There are different quantum computing models that we can chose for defining our computing framework. Examples are the adiabatic model [53, 54] and the quantum circuit model [55, 56], which depend on how the quantum system and its evolution is defined. Therefore, before choosing one, we start with a brief introduction to quantum mechanics in Section 2.1. We show how quantum systems are defined and how they evolve in an abstract sense, explaining the three main features that make quantum computing so powerful: superposition, interference and entanglement.

In Section 2.2 we chose the quantum circuit model, which is the most widely extended model, and it is used in quantum computers based on qubits. There, we define the elementary operations, or quantum gates, that compose the quantum circuits and are responsible of the evolution of the system. At a higher logic level, before constructing functional quantum algorithms, in Section 2.3 we show some complex quantum operators whose action provides a better intuition for a quantum programmer than the elementary quantum gates.

Finally, we can construct quantum algorithms from the previous operators. In particular, in Section 2.4 we show the Grover algorithm [26, 27], which is the basis of the quantum walk search algorithms studied in this thesis.

2.1. Introduction to Quantum Mechanics

In this section, we make an introduction to quantum mechanics by means of the four postulates [19, 40], so that we can define the mathematical formalism that we use in this thesis to develop the analysis on quantum computing.

2.1.1. Quantum systems and the computational basis

The first postulate of quantum mechanics makes reference to how the state of a quantum system is represented, and is related with the property of superposition.

Postulate 2.1. *Associated to any isolated physical system there is a complex vector space with inner product, i.e., a Hilbert space, known as the state space of the system $\mathcal{H}$. The system is completely described by a unit state vector $|\psi\rangle$ in its state space.*

In the bra-ket notation introduced by Dirac [57], a state vector is denoted by a ket as $|\psi\rangle$. In order to work with vectors, we need to represent them in a basis of the state space. Suppose we have a classical memory register allowing the representation of N different integer numbers from 0 to $N-1$. In quantum computing we usually work with an equivalent N-dimensional quantum state space, where we define the computational basis $\mathcal{C} := \{|i\rangle\,,\ i = 0, \ldots, N-1\}$ [19]. In contrast to a classical register, where only a number at a time can be stored, a quantum register can be in superposition, so that a state can be represented as a linear combination of this basis as:

$$|\psi\rangle = \sum_{i=0}^{N-1} a_i \,|i\rangle\,, \tag{2.1}$$

where the coefficients a_i are complex numbers usually denoted as amplitudes. Moreover, a ket can be represented in matricial form as a column vector formed by the amplitudes a_i.

Associated to any ket $|\psi\rangle$ there is a bra $\langle\psi|$, so that $\langle\psi| = |\psi\rangle^\dagger$. The $\dagger$ symbol denotes a linear operation known as the adjoint. Therefore, a bra state can be expanded in the basis of bras as:

$$\langle\psi| = \sum_{i=0}^{N-1} a_i^* \,\langle i|\,, \tag{2.2}$$

where the asterisk represents complex conjugation. In matricial representation the adjoint is obtained transposing and conjugating, so that any bra corresponds to a row vector with the conjugated amplitudes a_i^*.

The inner product between two vectors $|\psi_1\rangle$ and $|\psi_2\rangle$ is just the product between the bra of the first vector and the ket of the second one, so it is expressed as $\langle\psi_1\,|\psi_2\rangle$, and provides an scalar number. This product is not symmetric, since it satisfies that $\langle\psi_1\,|\psi_2\rangle = (\langle\psi_2\,|\psi_1\rangle)^*$. With it, we can define the square of the norm of a vector as $\||\psi\rangle\|^2 := \langle\psi\,|\psi\rangle$.

The computational basis is orthonormal, so that all the vectors are perpendicular to each other and their norm is 1. Therefore, the inner product between them is given by $\langle i\,|j\rangle = \delta_{ij}$, where

$$\delta_{ij} := \begin{cases} 1 & \text{if } i = j, \\ \\ 0 & \text{if } i \neq j. \end{cases} \tag{2.3}$$

Since the basis is orthonormal, the amplitudes a_i of any state $|\psi\rangle$ can be obtained by inner products as:

$$a_i = \langle i\,|\psi\rangle\,, \tag{2.4}$$

and the squared norm of the vector as:

$$|||\psi\rangle||^2 = \langle\psi\,|\psi\rangle = \sum_{i=0}^{N-1} |a_i|^2. \tag{2.5}$$

For a vector to represent a valid quantum state, its norm must be 1.

The second postulate explains how a quantum system evolves and it is the one related with interference.

Postulate 2.2. *The evolution of a closed quantum system is described by a unitary transformation. This means that the state $|\psi(t_0)\rangle$ of the system at time t_0 is related to the state $|\psi(t_1)\rangle$ of the system at time t_1 by a unitary operator U, which depends only on the times t_0 and t_1, so that $|\psi(t_0)\rangle = U\,|\psi(t_1)\rangle$.*

The time evolution of any closed quantum system is governed by the Schrödinger equation [57,58]:

$$i\hbar\frac{d\,|\psi(t)\rangle}{dt} = H\,|\psi(t)\rangle\,, \tag{2.6}$$

where $\hbar$ is the reduced Plank constant, and H is the Hamiltonian of the system describing its energy. This operator is Hermitian, so that it is equal to its adjoint: $H^\dagger = H$. The solution of this linear equation is given by a unitary operator, which is the responsible of the interference phenomena in a quantum system. For simplicity, let us suppose that H does not depend on time. The evolution is given by the unitary operator

$$U = e^{-\frac{\mathrm{i}}{\hbar}H(t_1-t_0)}. \tag{2.7}$$

A unitary operator satisfies that $U^\dagger U = UU^\dagger = \mathbb{1}$, where $\mathbb{1}$ is the identity operator. Then, $U^\dagger = U^{-1}$. Since H is Hermitian, the adjoint of U is $U^\dagger = e^{+\frac{\mathrm{i}}{\hbar}H^\dagger(t_1-t_0)} = e^{+\frac{\mathrm{i}}{\hbar}H(t_1-t_0)}$, and then it is trivially the inverse of U. In order to physically implement an arbitrary unitary transformation, we would need to find a Hamiltonian such that evolving for a fixed quantity of time, such unitary U is applied. Nevertheless, a quantum computer usually provides a set of primitive unitary operators from which any other can be constructed, as we will see in the following section.

The same as vectors, we can represent operators in matricial form as square matrices expanding in the outer products of the computational basis:

$$U = \sum_{i,j=0}^{N-1} U_{ij}\,|i\rangle\,\langle j|\,, \tag{2.8}$$

where the matrix elements U_{ij} in the computational basis can be obtained as:

$$U_{ij} = \langle i|\,U\,|j\rangle\,. \tag{2.9}$$

The third postulate makes reference to how a quantum system is measured in order to obtain information about it.

Postulate 2.3. *A quantum projective measurement is described by a Hermitian observable operator O acting on the state space of the system, whose spectral decomposition is*

$$O = \sum_{\lambda} \lambda\Pi_\lambda, \tag{2.10}$$

where Π_λ is the orthogonal projector on the eigenspace of O with eigenvalue λ. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement, then the probability that result λ occurs is given by $p_\lambda = \langle\psi| \Pi_\lambda |\psi\rangle$, and the state of the system after the measurement is

$$\frac{\Pi_\lambda |\psi\rangle}{\sqrt{p_\lambda}}. \tag{2.11}$$

The physical information that we obtain from the system is determined by the probability distribution of the possible measurement outcomes. Moreover, if the quantum state is in a superposition with components in different eigenspaces of the observable, this superposition is broken by the measurement and the result is the normalized projection onto the subspace associated with the result. Therefore, posterior measurements would always produce the same result. For this reason, we cannot obtain all the information about a state just measuring a single copy of it, and several copies are usually required.

Note that the probabilities p_λ for two quantum states differing in a global phase $e^{\mathrm{i}\theta}$ are the same. For example if we measure the state $|\psi'\rangle = e^{\mathrm{i}\theta} |\psi\rangle$ we have:

$$\langle\psi'| \Pi_\lambda |\psi'\rangle = e^{-\mathrm{i}\theta} e^{\mathrm{i}\theta} \langle\psi| \Pi_\lambda |\psi\rangle = \langle\psi| \Pi_\lambda |\psi\rangle . \tag{2.12}$$

This occurs for whatever observable O, and the global phase is unaffected by a unitary evolution. Thus, both states are physically indistinguishable and equivalent. For this reason, in quantum mechanics global phases play no role, and two states or unitary operations differing in a global phase are considered to be equal.

Orthogonal projectors are Hermitian, so that $\Pi_\lambda^\dagger = \Pi_\lambda$, and idempotent, so that $\Pi_\lambda^2 = \Pi_\lambda$. Moreover, since for Hermitian operators the subspaces of different eigenvalues are orthogonal, we have that $\Pi_\lambda \Pi_{\lambda'} = \delta_{\lambda\lambda'} \Pi_\lambda$, and they satisfy the following completeness relationship of the identity:

$$\mathbb{1} = \sum_\lambda \Pi_\lambda. \tag{2.13}$$

In this thesis we only deal with orthogonal projectors, and thus we omit the adjective and refer to them directly as projectors.

In quantum computing we are interested in measuring in the computational basis. The projector associated to each computational basis state $|i\rangle$ is $\Pi_i = |i\rangle \langle i|$. Thus, a diagonal observable in the computational basis is needed:

$$O = \sum_i \lambda_i |i\rangle \langle i| . \tag{2.14}$$

The measurement result λ_i is a physical quantity that depends on the particular experimental set. In quantum computing we are interested only in the index i of the computational basis. Thus, in a logic sense, we can say that the result of the measurement is directly the number i. The probability in this case can be obtained simply making the inner product with the corresponding computational basis state and taking the squared modulus:

$$p_i = \langle\psi| \Pi_i |\psi\rangle = \langle\psi |i\rangle \langle i |\psi\rangle = \langle i |\psi\rangle (\langle i |\psi\rangle)^* = |\langle i |\psi\rangle|^2 , \tag{2.15}$$

and we can also calculate it from the amplitudes as $p_i = |a_i|^2$. Furthermore, note that all eigenvalues must be different in order to project onto a particular computational basis state. If it is not possible

to construct an observable that is not degenerate, we need a complete set of commuting observables (CSCO) sharing the computational basis states as eigenbasis, so that the set of eigenvalues for each computational basis state is different [59]. An example can be seen in the context of qubits in Section 2.2.3.

In some quantum algorithms, instead of measuring the computational basis index, the aim is to determine the expected value of an observable. Examples are quantum chemistry algorithms for estimating the energy of molecules [60]. The expected value of an observable O for a system in state $|\psi\rangle$ is defined as:

$$\langle O \rangle_\psi := \langle \psi | O | \psi \rangle = \sum_\lambda p_\lambda \lambda, \tag{2.16}$$

so that it is the weighted average of the eigenvalues, where the weights are the probabilities of measuring each eigenvalue. This value is sampled experimentally measuring the observable many times and taking the average, which requires preparing the same quantum state for each sample.

2.1.2. Density operator

Before going on with the fourth postulate, it is useful to introduce the density matrix formalism [61].

The state of a quantum system represented by the vector $|\psi\rangle$ can also be represented by a density operator $\rho = |\psi\rangle \langle\psi|$. In this case we say that it is a pure state, since it corresponds to a particular vector state. However, the density operator formalism allows states that do not correspond to a particular vector state, but to an ensemble of them. This allows the introduction of classical uncertainty about the particular quantum state. Suppose we have prepared a quantum state, but we do not know what state. We only know that there is a probability p_1 of having prepared $|\psi_1\rangle$ and p_2 of having prepared $|\psi_2\rangle$. Of course, $p_1 + p_2 = 1$. Then, we can represent the state as an ensemble of both possibilities with their corresponding probabilities as:

$$\rho = p_1 |\psi_1\rangle \langle\psi_1| + p_2 |\psi_2\rangle \langle\psi_2| . \tag{2.17}$$

Although the state is actually one of both, the density matrix takes this classical uncertainty into account at the time of obtaining the final statistics, even after a quantum evolution. It is just as if we averaged the statistics of each independent pure state, weighted by their probabilities.

The same as quantum state vectors must be normalized so that their norm is 1, density operators have a normalization condition to represent valid quantum states. In this case the trace must be normalized so that it is 1. Since the trace of a matrix is the sum of its diagonal elements, using equation (2.9) it is calculated with the computational basis as:

$$\mathrm{Tr}[\rho] = \sum_{i=0}^{N-1} \langle i | \rho | i \rangle = \sum_{i=0}^{N-1} \rho_{ii} = 1. \tag{2.18}$$

Moreover, they must be positive operators. Note that the outer product $|\psi\rangle \langle\psi|$ is the same regardless of the global phase of the state. Since $(e^{i\theta} |\psi\rangle)^\dagger = e^{-i\theta} \langle\psi|$, the global phases are canceled out between the bra and the ket. Thus, in the density matrix formalism there is no notion of global phase, and the density operator ρ is unique for each state.

The quantum evolution for a density operator is also given by a unitary transformation as:

$$\rho(t_1) = U\rho(t_0)U^\dagger. \tag{2.19}$$

The probability of obtaining each of the outcomes of an observable O in the form of (2.10) is calculated in this case as:

$$p_\lambda = \mathrm{Tr}[\Pi_\lambda \rho], \tag{2.20}$$

and the state of the system after the measurement is

$$\frac{\Pi_\lambda \rho \Pi_\lambda}{p_\lambda}. \tag{2.21}$$

In the case of measuring in the computational basis we have:

$$p_i = \mathrm{Tr}[\Pi_i \rho] = \mathrm{Tr}[|i\rangle \langle i| \rho] = \sum_{k=0}^{N-1} \langle k | i\rangle \langle i| \rho |k\rangle = \sum_{k=0}^{N-1} \delta_{ik} \langle i| \rho |k\rangle = \langle i| \rho |i\rangle = \rho_{ii}, \tag{2.22}$$

so the probability is given by the corresponding diagonal element in the computational basis.

2.1.3. Example about superposition and interference

Now we can see a quite simple example to gain some intuition about the usefulness of superposition and interference with respect to a classical system. Suppose we have a two-level system, with computational basis $\mathcal{C} = \{|0\rangle, |1\rangle\}$. Let us define two superposition states:

$$|+\rangle := \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle), \tag{2.23}$$

$$|-\rangle := \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle). \tag{2.24}$$

In both states, when measuring in the computational basis, we have a probability of one half of measuring either 0 or 1. This statistic is the same as for the mixed state

$$\rho = \frac{1}{2} |0\rangle \langle 0| + \frac{1}{2} |1\rangle \langle 1|, \tag{2.25}$$

which denotes that the state is actually in $|0\rangle$ or $|1\rangle$ but we have a classical ignorance about it. One could think that the same happens for the quantum superposition states since they have the same statistics. Nevertheless, thanks to interference we can proof that this is not the case. Let us define a unitary operator U such that $U|0\rangle = |+\rangle$ and $U|1\rangle = |-\rangle$, and apply it to our superposition states:

$$U|+\rangle = \frac{1}{\sqrt{2}} (U|0\rangle + U|1\rangle) = \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle + |0\rangle - |1\rangle) = |0\rangle, \tag{2.26}$$

$$U|-\rangle = \frac{1}{\sqrt{2}} (U|0\rangle - U|1\rangle) = \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle - |0\rangle + |1\rangle) = |1\rangle. \tag{2.27}$$

Let us also apply it to the mixed state:

$$U\rho U^\dagger = \frac{1}{2}\left(U\left|0\right\rangle\left\langle 0\right|U^\dagger + U\left|1\right\rangle\left\langle 1\right|U^\dagger\right) = \frac{1}{2}\left(\left|+\right\rangle\left\langle +\right| + \left|-\right\rangle\left\langle -\right|\right)$$
$$= \frac{1}{2}\left(\left|0\right\rangle\left\langle 0\right| + \left|0\right\rangle\left\langle 1\right| + \left|1\right\rangle\left\langle 0\right| + \left|1\right\rangle\left\langle 1\right|\right) + \frac{1}{2}\left(\left|0\right\rangle\left\langle 0\right| - \left|0\right\rangle\left\langle 1\right| - \left|1\right\rangle\left\langle 0\right| + \left|1\right\rangle\left\langle 1\right|\right) = \rho. \tag{2.28}$$

Since the states $|+\rangle$ and $|-\rangle$ are in superposition, the unitary evolution produces an interference between the computational basis states. In the case of $|+\rangle$, there is a destructive interference for $|1\rangle$ and constructive for $|0\rangle$, and the other way around for state $|-\rangle$. If we measure the result in the computational basis we can now distinguish them with certainty. However, for the mixed state, since there is no coherence between the computational basis states, there is no interference and the same mixed state is obtained after the evolution, so that the same uniform statistic is obtained. This example shows that interference needs superposition, which is not possible for classical computers. Moreover, a quantum superposition is not only the fact of being in several states simultaneously. For example, applying U to states $|+\rangle$ and $|-\rangle$ makes them distinguishable despite being superpositions of the same states with the same probability. The relative phase of -1 in the state $|1\rangle$ plays an important role, so that these superpositions are new entities different to the computational basis states. Of course, they do not have classical counterpart.

2.1.4. Composite systems

The fourth and last postulate makes reference to composite systems, and it is the one related with entanglement. Furthermore, it is crucial for constructing complex quantum systems from simpler ones.

Postulate 2.4. *The state space of a composite physical system is the tensor product of the state spaces of the component physical subsystems. Moreover, if we number them from* 1 *through* n*, and subsystem* i *is prepared in the state* $|\psi_i\rangle$*, then the joint state of the total system is* $|\psi_1\rangle_1 \otimes |\psi_2\rangle_2 \otimes \cdots \otimes |\psi_n\rangle_n$.

The symbol $\otimes$ represents the tensor product and the subindex in each ket of the tensor product represents the particular register. Sometimes, when it is clear, we may remove either of them.

The computational basis of the composite system is obtained taking all the tensor product combinations of the computational basis states of each system. Therefore, the dimension of the composite Hilbert space is $\prod_i^n N_i$, where N_i is the dimension of subsystem i. In order to measure in the total computational basis, we just have to measure each subsystem independently, so that all of them turn into a particular state, and the total system turns into their tensor product.

Instead of measuring the total system, we can also perform a partial measurement, so that only a part of the system is measured. Without loss of generality we can consider a bipartite system, where we only measure the first register. The computational basis is

$$\mathcal{C} = \{\left|i\right\rangle_1\left|j\right\rangle_2,\ i = 0, \ldots, N_1 - 1,\ j = 0, \ldots, N_2 - 1\}. \tag{2.29}$$

The probability $(p_1)_i$ of measuring the computational basis state i of the first register is the sum of the probabilities of measuring each state $\left|i\right\rangle_1\left|j\right\rangle_2$ for all j. First, let us expand the state in the

composite computational basis as:

$$|\psi\rangle = \sum_{\substack{i=0 \\ j=0}}^{\substack{i=N_1-1 \\ j=N_2-1}} a_{ij} \, |i\rangle_1 \, |j\rangle_2 \, . \tag{2.30}$$

Therefore:

$$(p_1)_i = \sum_{j=0}^{N_2-1} |\, {}_1\langle i| \; {}_2\langle j \; |\psi\rangle|^2 = \sum_{j=0}^{N_2-1} |a_{ij}|^2 \, . \tag{2.31}$$

Note that if we take a partial inner product between the computational basis state of the first register and the composite state, the result is a vector living in the second register:

$${}_1\langle i \, |\psi\rangle = \sum_{\substack{k=0 \\ j=0}}^{\substack{k=N_1-1 \\ j=N_2-1}} a_{kj} \;\; {}_1\langle i \, |k\rangle_1 \, |j\rangle_2 = \sum_{\substack{k=0 \\ j=0}}^{\substack{k=N_1-1 \\ j=N_2-1}} a_{kj}\delta_{ik} \, |j\rangle_2 = \sum_{j=0}^{N_2-1} a_{ij} \, |j\rangle_2 \, , \tag{2.32}$$

whose norm is precisely the probability $(p_1)_i$ in (2.31). For this reason, in this thesis we also calculate the probabilities of partial measurements taking an inner product with the corresponding computational basis state. Nevertheless, in this case we take the squared norm of the result, rather than the squared modulus of a complex number since the result is actually a vector. Then:

$$(p_1)_i = || \, {}_1\langle i \, |\psi\rangle \, ||^2 \, . \tag{2.33}$$

In the case of working with a density operator ρ, instead of taking an inner product we take the sandwich with the corresponding computational basis state of the measured register. Again, the result is not a number, but a density operator living in the unmeasured register, and the probability is obtained taking the trace of the result:

$$(p_1)_i = \mathrm{Tr}[\, {}_1\langle i| \, \rho \, |i\rangle_1]. \tag{2.34}$$

An important feature of quantum mechanics that arises when we have composite systems is entanglement. When a quantum state can be written as a tensor product of different states on each register, we say that the state is not entangled. However, we can have states that cannot be expressed that way. For example, let us consider two systems with dimension 2 each of them, and consider the state

$$|\psi\rangle = \frac{1}{\sqrt{2}} \, |0\rangle \, |0\rangle + \frac{1}{\sqrt{2}} \, |1\rangle \, |1\rangle \, . \tag{2.35}$$

This state cannot be expressed as a tensor product of two independent states, and therefore is entangled. When this happens, the measurement statistics on each register are not independent. For example, suppose we measure the second register. With a probability of one half the system results in $|1\rangle$ in the second register. However, the only possibility for that condition is that the composite system results in $|1\rangle$, so that the first register also turns into $|1\rangle \, |1\rangle$ even without having been measured. Therefore, a posterior measurement of the first register would yield 1 with certainty. This contrasts with classical computing, where classical registers are always independent.

With regard to the quantum evolution, a unitary operator that can be expressed as a tensor product of unitaries on each subsystem will apply each unitary independently on each register, so entanglement is neither created nor destroyed. However, there are unitary operators that cannot be expressed this way, and therefore are said to be entangling, because they change the quantity of entanglement of the system.

2.2. Quantum Circuits

The most widely extended quantum computing model is the quantum circuit model [19, 55, 56]. Quantum circuits are the quantum analogue of classical circuits formed by bits whose information is processed by logic gates. In this case the system is formed by qubits represented by wires, which are the quantum analogue of classical bits [21], and the quantum information is transformed by quantum gates, which are unitary operations. At the end of the circuit, the information of the qubits is obtained by measurements. An example is shown in Figure 2.1.

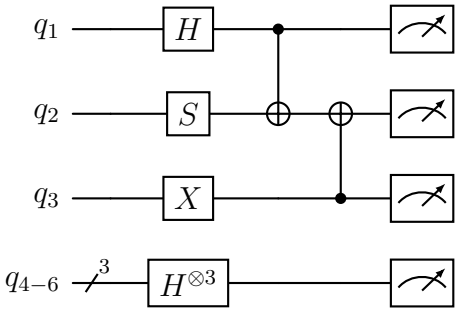


Figure 2.1: *Example of a quantum circuit with six qubits. The first three qubits are represented as independent wires, whereas the last three qubits are represented as a bundled wire. There are six single-qubit gates: four Hadamard gates applied to qubits* 1*,* 4*,* 5 *and* 6*, an* S *gate applied to qubit* 2*, and an* X *gate applied to qubit* 3*. There are also two multi-qubit gates: a CNOT between qubits* 1 *and* 2 *and other CNOT between qubits* 3 *and* 2*. The quantum information is processed from left to right applying the different quantum gates, and the qubits are measured at the end.*

We use the big-endian convention for the order of the qubits in the quantum circuit. Therefore, the first qubit of a composite system corresponds to the upper-most qubit in the circuit, and the last qubit is at the bottom [62]. With respect to the evolution time, it flows from left to right. This means that a gate at the left is applied before than one at the right. Note that this is in contrast with the application of operators in an algebraic sense. For example, if we had two gates, U_1 and U_2, and we wanted to apply $U = U_2U_1$, the gate U_1 would act first and would be at the left of the gate U_2 in the circuit, although they are the other way around in the algebraic product.

Any part of a quantum circuit composed only of quantum gates represents a unitary operator. Since for a unitary operator $U = U_2U_1$ the adjoint is obtained as $U^\dagger = U_1^\dagger U_2^\dagger$, the circuit representing the inverse U^{-1} is obtained reversing the order of the gates and substituting them by their inverses.

In the following subsections we explain deeper the three main elements of quantum circuits.

2.2.1. Qubits

A qubit is a two-level system with computational basis states $|0\rangle$ and $|1\rangle$, representing the two possible values, 0 and 1, of a classical bit [21]. Nevertheless, a qubit can take any state in superposition, going beyond the possibilities of a classical computer. There are several proposals for the physical implementation of qubits. For example, superconductor qubits [63,64], photons [65,66], ion traps [67]

and neutral atoms [68]. In Chapter 6 we use an IBM quantum computer based on superconductor qubits [33] for some experimental results.

The same as n bits can be concatenated in order to create a memory register with $N = 2^n$ different states, we can create a composite system of n qubits with an associated N-dimensional Hilbert space. If we consider the sequence of qubit states as a binary bitstring, we have $N = 2^n$ different possibilities. Therefore, each computational basis state correspond to the corresponding binary bitstring, and we usually represent them without a tensor product. For example, the fourth computational basis state for a system with $n = 3$ qubits is expressed as:

$$|3\rangle = |0\rangle_1 \otimes |1\rangle_2 \otimes |1\rangle_3 = |011\rangle \,. \tag{2.36}$$

2.2.2. Quantum gates

Quantum gates are the quantum analog of logic gates in classical circuits, although all the operations are reversible in contrast to classical gates. They are elementary operations that serve as building blocks for the construction of any arbitrary unitary operator [19, 69, 70]. In practice, a quantum computer is only able of implementing physically a reduced set of gates, known as universal set, from which any other gate can be composed. However, in a logic sense, we can consider as elementary gates a larger set of operations in order to construct quantum circuits with a greater human intuition, and let the quantum computer to compile them into the universal set. Moreover, the universal set can depend on the particular nature of the quantum computer. Therefore, in the following we are going to see a broad range of quantum gates, and where it is pertinent, show how they can be compiled into other gates.

2.2.2.1. Single-qubit gates

The first kind of gates that we are going to study are those that act on a single qubit. In classical computing the unique logic gate acting on a bit is the NOT gate, which flips the value of a bit from 0 to 1 or vice versa. In the case of a qubit, the Pauli matrix σ_x performs the change between the states $|0\rangle$ and $|1\rangle$. However, since a qubit is not restricted to these two states, but can take infinite possible states in superposition, we can also have in principle infinite single qubit gates. These gates are indeed those of the unitary group $U(2)$.

Let us start with the Pauli matrices. In quantum computing it is usual to denote them as X, Y and Z, avoiding the σ in the notation. Their mathematical expressions are the following:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \; Y = \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}, \; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.37}$$

The action of the X gate is known as bit-flip, in analogy to the classical NOT gate. However, the action of the other gates have no classical analog. The Z gate performs a phase-flip, because changes the state $|1\rangle$ to $-|1\rangle$, letting unchanged the state $|0\rangle$. The Y gate is just a combination of X and Z, since $Y = \mathrm{i}XZ$, so it is the product of both up to a global phase.

Other very important gates are the following three:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \; S = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{i} \end{pmatrix}, \; T = \begin{pmatrix} 1 & 0 \\ 0 & e^{\mathrm{i}\pi/4} \end{pmatrix}. \tag{2.38}$$

The H gate, known as Hadamard gate, is used to create superposition states. This gate transforms the state $|0\rangle$ into $|+\rangle$ and $|1\rangle$ into $|-\rangle$, where the states $|+\rangle$ and $|-\rangle$ were introduced in (2.23) and (2.24). Since $H^\dagger = H$, this gate is self-inverse, and therefore performs the transformation also the other way around. Moreover, since $|+\rangle$ and $|-\rangle$ are the eigenstates of the Pauli matrix X, the gates Z and X can be interconverted as $X = HZH$ and $Z = HXH$. The S and T gates are like the phase-flip gate Z, but applying a phase of $e^{i\pi/2}$ and $e^{i\pi/4}$, respectively, to state $|1\rangle$. Note that $T^2 = S$, and $S^2 = Z$. Therefore, the Z and S gates would not be necessary in a universal set containing the gate T. However, as we will explain below, the S gate is usually considered independently of T.

The next gates that we are going to introduce are parameterized gates, which depend on a parameter. Therefore, they can represent infinite gates. These gates are at the core of variational algorithms like the variational quantum eigensolver [71] or quantum neural networks [72], allowing the codification of both data and weights in the circuit. We can generate parameterized gates by exponentiation of the Pauli matrices. We define a rotation gate $R_\sigma(\theta) = \exp(-\mathrm{i}\theta\sigma/2)$ for the three Pauli matrices σ as follows:

$$R_X(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\mathrm{i}\sin\left(\frac{\theta}{2}\right) \\ -\mathrm{i}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \; R_Y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \; R_Z(\theta) = \begin{pmatrix} e^{-\mathrm{i}\theta/2} & 0 \\ 0 & e^{\mathrm{i}\theta/2} \end{pmatrix}. \tag{2.39}$$

These three matrices belonging to the $U(2)$ group are the equivalent of the three tridimensional rotations in $SO(3)$. Thus, two of them can be used to generate any single qubit gate U. For example, taking R_X and R_Z:

$$U = e^{i\alpha} R_X(\beta) R_Z(\gamma) R_X(\delta), \tag{2.40}$$

where the angles β, γ, and δ are known as Euler angles. Furthermore, since $X = HZH$, then $R_X(\theta) = HR_Z(\theta)H$, and only an arbitrary rotation gate is necessary. For example, IBM quantum computers have the $R_Z(\theta)$ gate in the universal set, along with others that allow the implementation of the Hadamard gate [33].

The last single qubit gate that we are going to introduce is the phase gate $P(\theta)$:

$$P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{\mathrm{i}\theta} \end{pmatrix}, \tag{2.41}$$

which applies an arbitrary phase $e^{\mathrm{i}\theta}$ to the state $|1\rangle$, generalizing the Z, S and T gates. This gate is actually the $R_Z(\theta)$ gate up to a global phase, since $P(\theta) = e^{\mathrm{i}\theta/2} R_Z(\theta)$. Nevertheless, it is useful to consider it as an independent gate due to its particular action.

In a circuit with several qubits we may want to indicate somehow what is the qubit on which the gate acts. For each gate acting on an individual qubit we can define a unitary operator on the composite system. For example, for a Z gate acting on the i-th qubit we define Z_i as:

$$Z_i = \mathbb{1}_2 \otimes \cdots \otimes Z \otimes \cdots \otimes \mathbb{1}_2, \tag{2.42}$$

where we have a single-qubit operator Z in the i-th position of the tensor product, and $n-1$ identities for the rest of the qubits. This means that the Z gate is only applied to the i-th qubit,

and the rest are idle. If we consider that each gate on the same column of a circuit are applied sequentially, this is enough to describe mathematically the action of the overall circuit. Nevertheless, we can suppose that all gates in the same column are applied at the same time and simplify the mathematical formulation. In the circuit of Figure 2.1 the six single qubit gates of the first column can be composed as a unique unitary operator as $H \otimes S \otimes X \otimes H \otimes H \otimes H$ directly. Moreover, if we have the same gate U acting on n different qubits we may denote it as $U^{\otimes n}$. For example, if we have three Hadamard gates on three qubits we denote it as:

$$H^{\otimes 3} = H \otimes H \otimes H. \tag{2.43}$$

In a real quantum computer whether different qubit gates can be applied simultaneously or not depends on the particular kind of hardware representing the qubits.

2.2.2.2. Single-controlled gates

Multi-qubit gates usually consists of what is called controlled gates. Let us star with gates controlled by a single qubit, and in particular, with the CNOT gate. This gate acts on two qubits. If the first qubit is the control and the second one the target, then the mathematical expression is

$$\text{CNOT} = |0\rangle\langle 0| \otimes \mathbb{1}_2 + |1\rangle\langle 1| \otimes X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.44}$$

This gate controls the application of an X gate on the target qubit. If the control qubit is in state $|0\rangle$, nothing happens. If it is in $|1\rangle$, the X gate is applied. Therefore, the action on an arbitrary computational basis state is

$$\text{CNOT}\,|x\rangle\,|y\rangle = |x\rangle\,|y \oplus x\rangle\,, \tag{2.45}$$

where $\oplus$ denotes the bitwise XOR operation or addition modulo 2. For a single bit, $0 \oplus 0 = 1 \oplus 1 = 0$ and $0 \oplus 1 = 1 \oplus 0 = 1$. Therefore, the CNOT gate is the quantum version of the classical XOR gate. This gate in a quantum circuit is represented as a black dot in the control qubit connected with the addition on the target, as shown in Figure 2.2(a). The CNOT gate is actually the only multi-qubit gate needed in a universal set containing all the single qubit gates to construct any arbitrary unitary operator [69]. Nevertheless, we are going to see other important controlled gates and how they can be decomposed back to the CNOT gate.

Instead of applying the X gate when the control qubit is in $|1\rangle$, we may want it the other way around and apply the bit-flip only if it is in state $|0\rangle$. This complementary CNOT gate is shown in Figure 2.2(b), where a white dot is used to denote that the control qubit must be in the state $|0\rangle$. A white dot can be turned into a black dot just surrounding it by X gates, and the same is true the other way around. Therefore, without loss of generality, in the following we will only use black dots unless a white dot is necessary.

Apart from an X gate, any arbitrary unitary operator can also be controlled, as shown in Figure 2.2(c). If the operator U is decomposed into simpler gates, we can consider that we have each gate being controlled by the control qubit. In this case we denote the general controlled gate with only a control qubit as a single-controlled-U gate.

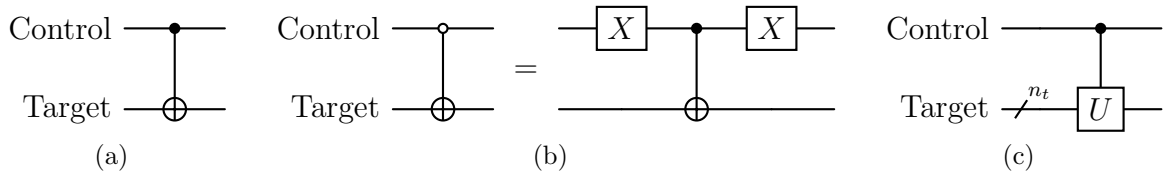


Figure 2.2: *a) Quantum circuit representation of a CNOT gate. b) Representation of a complementary CNOT gate and decomposition into a normal CNOT gate. c) Representation of a general single-controlled-U gate.*

Now suppose a single-qubit gate U is being controlled. The question that arises is how we can decompose it into single-qubit gates and CNOT gates. Let us suppose for the moment that U can be transformed into an X gate by a unitary transformation as $U = A^\dagger X A$ for some unitary gate A. Then, the A and $A^\dagger$ gates can be extracted from the target and be always applied surrounding a CNOT, as shown in Figure 2.3. If the X is applied, then U is actually applied on the target. If it is not applied, then $A^\dagger A = \mathbb{1}$ makes nothing.

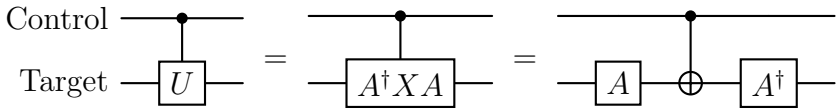


Figure 2.3: *Quantum circuit decomposition of a single-controlled-U gate acting on a single target qubit in the case that $U = A^\dagger X A$ for some arbitrary operator A.*

Now let us see what happens with the global phase of a gate when it is controlled. In this case the phase does not acts globally. For example, if we control $-X$ instead of X by a black dot, if the control qubit state is in state $|1\rangle$ then the target suffers a phase-flip apart from the bit-flip of the X gate. However, when the control is in state $|0\rangle$ no phase-flip is performed. A global phase from a unitary being controlled can be extracted as a phase gate acting on the control qubit when the dot is black, as shown in Figure 2.4.

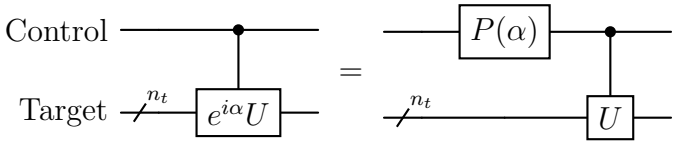


Figure 2.4: *Extraction of the global phase for an operator U being controlled by a single qubit as a phase gate acting on the control qubit.*

Finally, any single-qubit unitary U can be decomposed as $U = e^{i\alpha} AXBXC$ for some unitary operators satisfying $ABC = \mathbb{1}$ [19]. Therefore it can be decomposed into two CNOT gates, a phase gate and the three single-qubit gates of the transformation.

2.2.2.3. Multi-controlled gates

We can generalize the controlled gates to the case where we have more than one control qubits. In this case, to differentiate from the previous one, we denote as multi-controlled-U gate to a general gate U being controlled by several qubits.

The first example is the Toffoli gate, also denoted as CCNOT, since it is an X gate controlled by two qubits. Its representation is shown in Figure 2.5. In this case the X gate is only applied to the target when both control qubits are in state $|1\rangle$. This gate is special because it can be decomposed into CNOT, H and T gates, which are gates usually used as universal, and also allows the decomposition of any other multi-controlled gate [19].

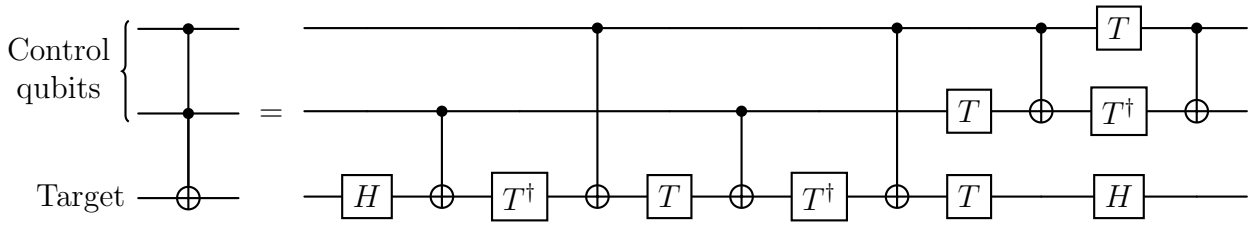


Figure 2.5: *Quantum circuit representation of a Toffoli gate and decomposition into CNOT and single-qubit gates.*

Let us consider a multi-controlled-U gate as the one showed in Figure 2.6. The U gate is applied to the target register only if all the control qubits are in state $|1\rangle$. We can decompose it with the help of ancilla qubits. Using a Toffoli gate we can check if a pair of two qubits satisfy the condition, and store the result in an ancilla qubit. Then, another Toffoli checks the condition between this ancilla and another control qubit, and stores the result in a new ancilla. The process is iterated until all the control qubits have been checked and the global result is stored in a final ancilla. This ancilla then controls the operation of the gate U on the target as a single-controlled-U gate. After that, all the ancilla qubits, which started in state $|0\rangle$, need to be uncomputed reversing the process with the Toffoli gates. For n_c control qubits this decomposition requires $2(n_c - 1)$ Toffoli gates and $n_c - 1$ ancilla qubits. These ancilla qubits must start in state $|0\rangle$, and they end up also in state $|0\rangle$. Thus, they can be reused for all the multi-controlled gates in the circuit, and the total number of ancilla qubits only depends on the larger multi-controlled gate, providing at most $n - 2$ ancilla qubits for a circuit with n qubits. Although this decomposition would be enough for any multi-controlled gate, it is important to mention that there can be more efficient schemes in the recent literature [73].

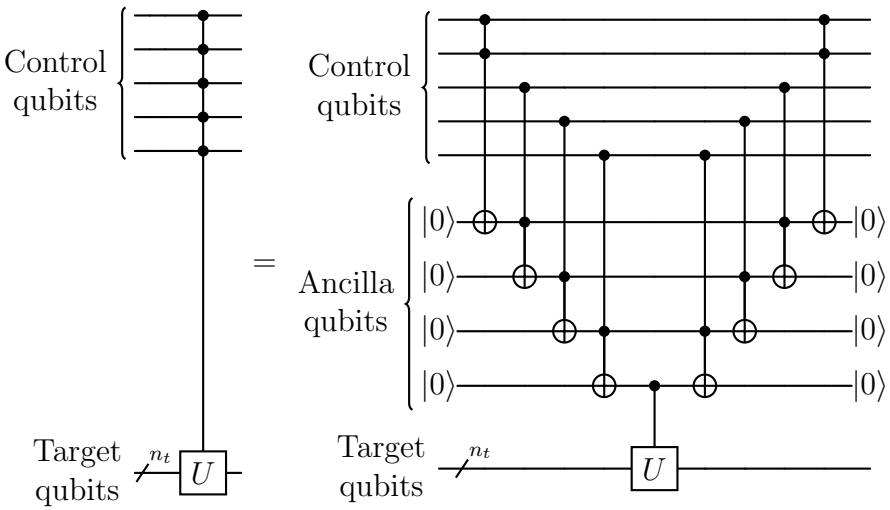


Figure 2.6: *Quantum circuit representation and decomposition of a multi-controlled-U operation that contains $n_c = 5$ control qubits into $2(n_c - 1) = 8$ Toffoli gates using $n_c - 1 = 4$ ancilla qubits.*

2.2.2.4. Controlled phase gate

The phase gate is special when is being controlled. In this case, the target qubit acquires a phase of $e^{i\theta}$ when it is in state $|1\rangle$. However, it cannot be attributed to this particular qubit, and it is attributed to the composite state considering all the qubits. Therefore, if we want to apply a relative phase on a particular computational basis state, we can use a controlled phase gate considering as target whatever qubit, as long as we take into account that if the target qubit must be in state $|0\rangle$, the phase gate must be surrounded by X gates. Thus, we can use a representation in which all the qubits represent the state to whom we want to apply the phase $e^{i\theta}$, and they control the phase gate on a ghost target qubit [74]. An example is shown in Figure 2.7 for the state $|110\rangle$ in a system with three qubits.

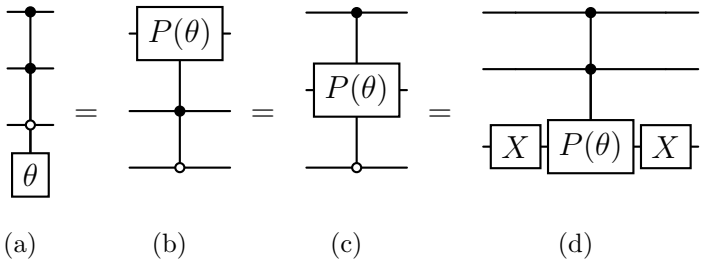


Figure 2.7: *a) Quantum circuit representation of a multi-controlled-$P(\theta)$ gate acting on a ghost qubit, which in this example applies a relative phase $e^{i\theta}$ to the state $|110\rangle$. The real target can be whatever qubit as shown in b)-d), as long as the target qubit is surrounded by X gates if it must be in state $|0\rangle$ for the application of the phase gate.*

2.2.2.5. Swap gate

A qubit-swap gate interchanges the states of two different qubits, therefore SWAP $|\alpha\rangle\,|\beta\rangle = |\beta\rangle\,|\alpha\rangle$ for two arbitrary qubits states. This gate can be decomposed into three CNOT gates [19] as shown in Figure 2.8.

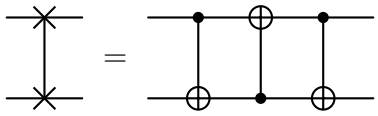


Figure 2.8: *Quantum circuit representation of a qubit-swap gate and decomposition into three CNOT gates.*

In an ideal quantum computer where we are able to apply a two-qubit gate between whatever two qubits, this gate can be implemented virtually at a free cost. We just have to rename the qubits and go on. However, actual quantum computers lack of a full connectivity between the qubits. Indeed, for these computers the swap gate is crucial for implementing two-qubit gates between non-connected qubits, performing swaps with the intermediate qubits of the chain, increasing the cost of the circuit [75]. For the sake of simplicity, we consider in this thesis that all qubits are connected between them.

2.2.2.6. Universal gates and the Clifford+T set

We have seen how multi-controlled gates can be decomposed into single-controlled gates, and these into CNOT gates. Indeed, any arbitrary unitary operator can be decomposed into CNOT gates and single-qubit gates without the help of ancilla qubits [69], although the decomposition that we have shown is more efficient requiring less gates [19]. Therefore, the CNOT gate, along with all the operators of the $U(2)$ group, form a universal set of gates.

Any single-qubit gate can be generated using the parameterized $R_Z(\theta)$ gate and other gates needed to convert it into a $R_X(\theta)$ gate. Although this would be enough for performing universal quantum computation, parameterized gates lack of a fault-tolerant implementation, since they actually represent infinite gates. In order to perform fault-tolerant quantum computing, a discrete and finite set of gates is required [36, 76]. It turns out that the H and T gates can approximate any single-qubit gate up to an arbitrarily small precision, as stated by Solovay-Kitaev theorem [77]. This error propagates linearly, so that the total error of the circuit is the sum of the errors of each individual gate. Moreover, recently an efficient algorithm for the decomposition of $R_z(\theta)$ gate was developed [78]. Despite the fact that $S = T^2$, the S gate is also introduced in the universal set of gates. The reason is that the H, S and CNOT gates generate the Clifford group, which is the group of gates that normalize the Pauli group [79, 80]. Whereas Clifford gates are easy to correct, T gates need a quite complicated quantum error-correction scheme. Indeed, it is usual to measure the cost of a quantum circuit in terms of T gates for fault-tolerant quantum computing [81, 82], after decomposing it into the Clifford+T set. Then, it is easier to work with an S gate instead of treating it as two T gates, although a quantum computer would apply the T gate twice in order to implement it in a physical sense.

In this thesis we consider that quantum computers are ideal, so we do not deal with quantum error correction. Therefore, we mostly express the circuits in terms of the gates we have defined in this section, considering that they could be decomposed into Clifford+T gates later.

2.2.2.7. Classically-controlled gates

In some quantum algorithms classical information also plays a role. Classical bits, which can carry the information of an intermediate measurement or of independent calculations, can be used to control the application of quantum gates. An example is shown in Figure 2.9, where a gate U is applied to the quantum register if the classical bit, represented as a double wire, is in state 1. This kind of gates are at the core of quantum error correction [36], and are crucial for the semiclassical walks [2] that we study in this thesis.

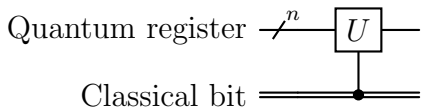


Figure 2.9: *Quantum circuit representation of a classically-controlled gate. The classical bit is represented as a double wire.*

2.2.3. Measurements

The computational basis of a single qubit corresponds to the two eigenvectors of the Pauli matrix Z, which has eigenvalue 1 for state $|0\rangle$ and -1 for $|1\rangle$. Therefore, in order to measure the state of a single qubit, this is the operator that is actually measured as observable. In the case of a system with n qubits, we have $N = 2^n$ computational basis states. However, we do not have an operator with N different eigenvalues to measure. For example, although the computational basis would be an eigenbasis of the operator $Z^{\otimes n}$, it would only have two different eigenvalues which would be degenerate. As mentioned in the previous section, in order to measure in the total computational basis we can use a CSCO. In this case we take the n different Z_i Pauli operators. Therefore, we need to perform n measurements in order to completely determinate a bitstring. Of course, this trivially corresponds to measuring each qubit independently.

There are cases where we may want to measure in a different basis. However, quantum computers usually only allow a physical measurement in the computational basis. Therefore, a different kind of measurement is implemented virtually transforming the state [19]. Let us suppose we want to measure in an orthonormal basis $\mathcal{B} = \{|\beta_i\rangle, i = 0, \ldots, N_1\}$, and we have a unitary operator $U_\mathcal{B}$ that transforms each computational basis state $|i\rangle$ into $|\beta_i\rangle$. We can obtain the statistics of measuring in the basis $\mathcal{B}$ measuring in the computational basis after the application of $U_\mathcal{B}^\dagger$, which rotates the basis $\mathcal{B}$ into the computational basis. Precisely, taking into account that $\langle\beta_i| = (U_\mathcal{B}|i\rangle)^\dagger = \langle i| U_\mathcal{B}^\dagger$, the probability $(p_\mathcal{B})_i$ of resulting in the state $|\beta_i\rangle$ after measuring an arbitrary state $|\psi\rangle$ is

$$(p_\mathcal{B})_i = |\langle\beta_i|\psi\rangle|^2 = \left|\langle i| U_\mathcal{B}^\dagger |\psi\rangle\right|^2. \tag{2.46}$$

Note that after the real measurement the state ends up in a computational basis state. If the measurement is at the end of the algorithm and we are only interested in the index i of the result, we are done. However, if it is an intermediate measurement and we want to continue with the state $|\beta_i\rangle$, we would have to apply $U_\mathcal{B}$ to the result of the measurement.

Examples of different basis of measurement are the eigenbasis of the other Pauli operators, mostly used in quantum chemistry for expressing Hamiltonian operators [60]. In the case of the Pauli X matrix the transformation is done with the Hadamard gate, so $U_\mathcal{B} = H$, and for the Y matrix it is done with $U_\mathcal{B} = SH$.

2.2.4. Classical simulation

In principle, using simple linear algebra we can calculate the evolution of any quantum state represented as a column vector just multiplying it by the unitary operator matrix. Therefore, the operations of a quantum computer could be simulated on a classical one, and we can obtain deterministically the final probability distribution. Note that in a real quantum computer we do not obtain the probability distribution, but a particular computational basis state with a certain probability. In the case that we want to make a stochastic simulation, therefore simulating how a real quantum computer would behave, we would just have to sample the resulting probability distribution using a random number generator. There are several software packages for simulating quantum circuits this way, as for example Cirq from Google [83] and Qiskit from IBM [84]. However, this simulation is only feasible for a relatively small number of qubits.

A composite system of n qubits can be in a superposition of the $N = 2^n$ computational basis states, and therefore we would have in general a quantum vector with 2^n different complex amplitudes. Even disregarding the error of approximating these numbers with a finite precision, in order to store them in a classical computer we would require a number of classical bits scaling exponentially with the number of qubits. For example, a computer with a RAM memory of 128 GB hardly can simulate a circuit with 32 qubits. Moreover, the classical simulation time grows with the size of the quantum vector, so it increases also exponentially with the number of qubits.

Note that this exponential scaling is due to the fact of entanglement between the qubits. If there would not be entanglement, we would just only need to store the individual state of each qubit, which would require $2n$ complex numbers. Therefore, entanglement is crucial for a quantum computer to exhibit advantage, because otherwise it could be simulated efficiently on a classical computer. Although necessary, it is not sufficient that the state is entangled to be hard to simulate, because Gottesman-Knill theorem states that quantum circuits composed solely of Clifford gates are efficient to simulate even with high entanglement [79].

Nevertheless, quantum algorithms that are efficiently simulated on a classical computer can also be of interest, giving rise to what is denoted as quantum-inspired classical algorithms. For example, algorithms that can be simulated using tensor networks [85], or the quantum PageRank algorithm [31,32], which we analyze further in Chapter 3.

2.3. Operators

So far, we have seen elementary gates performing quite simple operations at a very low level. The same as in classical computing, it is very difficult to program an algorithm just composing low level operations. In quantum computing we can define some operators by their specific action independently of the quantum computing model. Thus, they are at a higher level than the quantum gates we have previously seen. Although in the end these operators need to be decomposed into quantum gates if we want to implement them using quantum circuits, they perform operations that are more intuitive for a quantum programmer. Therefore, they can be used as building blocks for designing quantum algorithms.

2.3.1. Conditional operator

A conditional operator is the quantum equivalent of the *if-else* statement of classical programming languages [86,87]. Let us consider a composite system with two registers, so that $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$, where $\mathcal{H}_1$ corresponds to the conditional register and $\mathcal{H}_2$ to the target register. Let us also consider an orthonormal basis $\mathcal{B} = \{|\beta_i\rangle\,,\ i = 0, \dots, N_1\}$ of the Hilbert space of the first register. The conditional operator applies a unitary operator U_i to the target register if the conditional register is in state $|\beta_i\rangle$, so that it can be defined as:

$$C_{\mathcal{B}} := \sum_{i=0}^{N_1-1} |\beta_i\rangle \langle\beta_i| \otimes U_i. \tag{2.47}$$

In general, this operator corresponds to N_1 *if* statements. Nevertheless, these can be reduced if some U_i operators are the same for different states of the basis $\mathcal{B}$, where we would have the equivalent of

an *or* operation between two conditions for the same *if* statement. In the case we have $U_i = \mathbb{1}$, it would correspond to the lack of an *if* statement for the corresponding state $|\beta_i\rangle$.

In order to implement this operator with a quantum circuit, we need to perform a basis rotation to the computational basis. Let A be a unitary operator such that $A|i\rangle = |\beta_i\rangle$. Then:

$$C_{\mathcal{B}} = \sum_{i=0}^{N_1-1} A|i\rangle\langle i|A^\dagger \otimes U_i = (A\otimes\mathbb{1})\left(\sum_{i=0}^{N_1-1}|i\rangle\langle i|\otimes U_i\right)(A^\dagger\otimes\mathbb{1}) = (A\otimes\mathbb{1})C_C(A^\dagger\otimes\mathbb{1}), \tag{2.48}$$

where $C_{\mathcal{C}}$ is the conditional operator using the computational basis $\mathcal{C}$ for the quantum conditions. The implementation of $C_{\mathcal{C}}$ is done with what is called a uniformly controlled gate, which corresponds to a bunch of multi-controlled-U_i gates, for the N_1 different states of the computational basis of the conditional register. The circuit implementing the operator $C_{\mathcal{C}}$ is shown in Figure 2.10 for an example with $N_1 = 8$. This implementation is quite inefficient since it requires N_1 multi-controlled gates, and it is an open problem widely studied [88,89]. However, depending on the particular gates U_i, if some of them are equal, a more efficient implementation can be found [5], as we will see in Chapter 4.

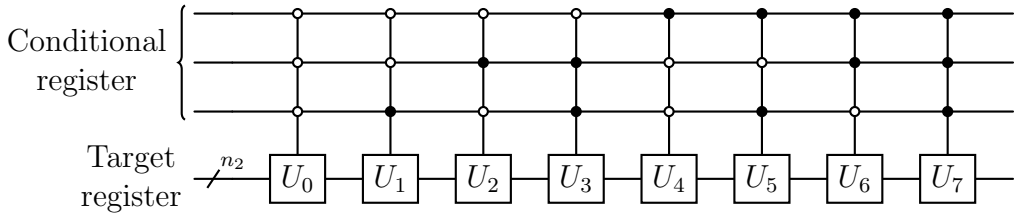


Figure 2.10: *Quantum circuit representation of an example of conditional operator in the computational basis for eight conditional states. There is a multi-controlled-U gate for each condition.*

The implementation of A depends on the particular basis $\mathcal{B}$, so that its compilation depends case to case and it is in general an open problem. Once A is compiled, the quantum circuit of $A^\dagger$ is obtained trivially reversing the order of the gates and substituting each elementary gate by its inverse. Luckily, in this thesis all conditional operators we deal with are in the computational basis, so that no basis rotation is needed.

2.3.2. Swap operator

A swap operator exchanges the quantum states between two registers of the same dimension [90]. Let N be the dimension of these registers. Then the swap operator is defined as:

$$S_w := \sum_{i,j=0}^{N-1} |i\rangle\langle j|\otimes|j\rangle\langle i|\,. \tag{2.49}$$

Note that this unitary operator is also Hermitian, so $S^2 = SS^\dagger = \mathbb{1}$.

The swap gate between two qubits is a particular instance of this operator. Moreover, the quantum circuit for a swap operator for two registers of n qubits is composed trivially by n swap gates [91], as shown in Figure 2.11.

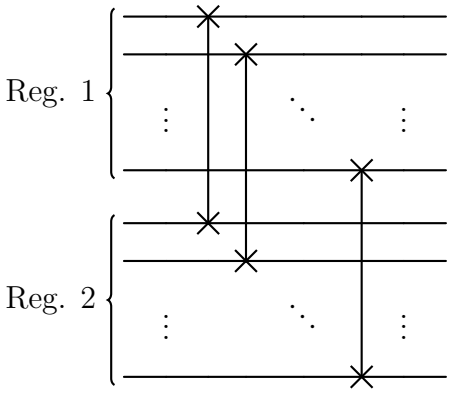


Figure 2.11: *Quantum circuit representation of the swap operator between two quantum registers. For each qubit of a register a qubit-swap gate is applied with the corresponding qubit of the other register.*

2.3.3. Permutation operator

A permutation operator adds some fixed quantity to each computational basis state. In this thesis we are interested in a permutation operator that acting on a N-dimensional system transforms the computational basis states $|x\rangle$ into $|x+1 \bmod N\rangle$. Therefore, it just adds 1 to each computational basis state [92]. We denote it as $\mathcal{P}^+$.

The quantum compilation of this operator is done recursively, so that for a register with $n+1$ qubits it is compiled adding a multi-controlled-X gate to the left of the operator for n qubits. For $n = 1$ the circuit is trivially composed by a single X gate. The general circuit is shown in Figure 2.12.

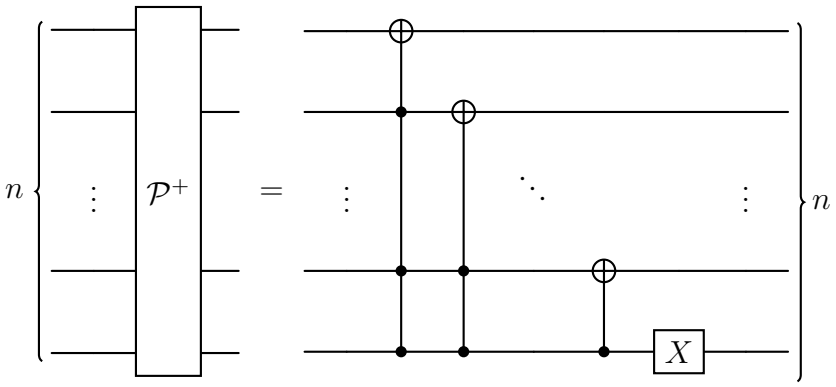


Figure 2.12: *Quantum circuit representation of a permutation operator that adds* 1 *to each computational basis state.*

A permutation operator $\mathcal{P}^-$ performing the inverse operation would be obtained by simply inverting the order of the gates. Nevertheless, it can also be compiled substituting all black dots by white dots in the circuit of $\mathcal{P}^+$.

2.3.4. Reflection operator

Let $\mathcal{H}_\mathcal{B}$ be a $N_\mathcal{B}$-dimensional subspace with an orthonormal basis $\mathcal{B} = \{|\beta_i\rangle\,,\ i = 0, \ldots, N_\mathcal{B}\}$. If we decompose the Hilbert space as $\mathcal{H} = \mathcal{H}_\mathcal{B} \oplus \mathcal{H}_\mathcal{B}^\perp$, then any state $|\psi\rangle$ can be decomposed into a parallel component and a perpendicular component to $\mathcal{H}_\mathcal{B}$:

$$|\psi\rangle = |\psi\rangle_\parallel + |\psi\rangle_\perp\,. \tag{2.50}$$

A reflection of $|\psi\rangle$ around the subspace $\mathcal{H}_\mathcal{B}$ changes the sign of the perpendicular component. Using the orthonormal basis we can obtain the projector operator onto the subspace $\mathcal{H}_\mathcal{B}$ as:

$$\Pi = \sum_{i=0}^{N_\mathcal{B}-1} |\beta_i\rangle\langle\beta_i|\,. \tag{2.51}$$

We have that $\Pi\,|\psi\rangle_\parallel = |\psi\rangle_\parallel$ and $\Pi\,|\psi\rangle_\perp = 0$. Therefore, an operator R performing the reflection around subspace $\mathcal{H}_\mathcal{B}$ can be defined using the projector Π as [26, 93]:

$$R := 2\Pi - \mathbb{1}. \tag{2.52}$$

In general we have that $N_\mathcal{B} << N$, where N is the dimension of the total Hilbert space $\mathcal{H}$. In order to implement the reflection in a quantum circuit, we actually implement the minus reflection $-R$, which flips the phase of the parallel component instead. Recall that this -1 is an unimportant global phase. Since $|\psi\rangle_\parallel \in \mathcal{H}_\mathcal{B}$, it can be expressed as a linear combination of the states $|\beta_i\rangle$. Therefore, flipping the sign of $|\psi\rangle_\parallel$ is equivalent to flipping the sign of all the states $|\beta_i\rangle$, letting unchanged any vector that is orthogonal to all of them. This is done with $N_\mathcal{B}$ minus reflections $-R_i$ around each state $|\beta_i\rangle$, i.e., $-R_i = -(2\,|\beta_i\rangle\langle\beta_i| - \mathbb{1})$. Indeed, it is easy to check that

$$-R = \prod_i^{N_\mathcal{B}-1} (\mathbb{1} - 2\,|\beta_i\rangle\langle\beta_i|) = \prod_i^{N_\mathcal{B}-1} (-R_i). \tag{2.53}$$

Now we need to find a compilation for a reflection R_i around a simple monodimensional subspace. We do it transforming the corresponding state into a computational basis state. Although any state is valid, in this thesis we transform it into state $|0\rangle$. Let A_i be a unitary operator such that $A_i\,|0\rangle = |\beta_i\rangle$. Then:

$$R_i = 2A_i\,|0\rangle\langle 0|\,A_i^\dagger - \mathbb{1} = A_i(2\,|0\rangle\langle 0| - \mathbb{1})A_i^\dagger, \tag{2.54}$$

so that each reflection around a single state $|\beta_i\rangle$ can be performed surrounding a reflection around the state $|0\rangle$ by A_i and $A_i^\dagger$.

Again, the difficult part is to find a quantum circuit for the operators A_i, and we will see that this is the great challenge for compiling the unitary evolution operator of quantum walks [91, 94]. The reflection around $|0\rangle$, actually the minus reflection, is implemented with a multi-controlled-$P(\pi)$ gate controlled by the state $|0\rangle$. As we saw in the previous section, this gate, shown in Figure 2.7, applies a phase of -1 to the controlling state letting the rest of the computational basis unchanged, thus effectively implementing the minus reflection.

2.3.5. Arbitrary phase rotation operator

The arbitrary phase rotation (APR) operator is a modification of the reflection R [95,96]. Instead of introducing a relative phase of -1 between the parallel and perpendicular components, this operator introduces an arbitrary complex phase $e^{i\theta}$. It is defined as:

$$R(\theta) := (1 - e^{i\theta})\Pi - \mathbb{1}. \tag{2.55}$$

A normal reflection is recovered for $\theta = \pi$. Again, in order to implement it in a quantum circuit we take $-R(\theta)$ instead. We have that for a general state decomposed as $|\psi\rangle = |\psi\rangle_{\parallel} + |\psi\rangle_{\perp}$:

$$-R(\theta)\,|\psi\rangle = e^{i\theta}\,|\psi\rangle_{\parallel} + |\psi\rangle_{\perp}\,, \tag{2.56}$$

so that it applies a phase of $e^{i\theta}$ to the parallel component. Following an analogue procedure to that of the reflection, we can decompose it as the product of multiple APR operators around monodimensional subspaces:

$$-R(\theta) = \prod_{i}^{N_{\mathcal{B}}-1} (\mathbb{1} - (1 - e^{i\theta})\,|\beta_i\rangle\,\langle\beta_i|) = \prod_{i}^{N_{\mathcal{B}}-1} (-R_i(\theta)), \tag{2.57}$$

and any single phase rotation can be transformed into the phase rotation around state $|0\rangle$ as $R_i(\theta) = A_i((1 - e^{i\theta})\,|0\rangle\,\langle 0| - \mathbb{1})A_i^{\dagger}$. Therefore, the only difference with the compilation of a reflection R is that instead of a phase of $-1 = e^{i\pi}$, we have $e^{i\theta}$, and therefore we use a general multi-controlled-$P(\theta)$ gate instead.

2.3.6. Oracle

Let $f(x) : \{0,1\}^n \to \{0,1\}^m$ be a classical function that takes as input a bitstring of n bits and outputs a bitstring of m bits. A quantum oracle O_f is a black-box operator that implements the function $f(x)$ taking the input in a quantum register, and outputting the result in another quantum register [19, 24, 25], such that

$$O_f\,|x\rangle\,|y\rangle := |x\rangle\,|y \oplus f(x)\rangle\,, \tag{2.58}$$

where $\oplus$ denotes the bitwise XOR operation. The first register is called data register, and the second one target register. When the target register is in state $|0\rangle$, it just stores the result of $f(x)$. Supposing a classical circuit for computing $f(x)$ is available, a quantum reversible circuit for the oracle can always be constructed with the same efficiency [19].

If we apply the oracle O_f to a quantum superposition in the first register, it computes the function $f(x)$ for all the computational basis states of the superposition simultaneously. This fact is known as quantum parallelism, since all the results are obtained using only once the oracle. However, since we have to measure the state in order to obtain information, we cannot obtain all the results of $f(x)$ so simply. Nevertheless, when combining the oracle parallelism with quantum interference, we can obtain some algorithms that outperform classical ones, for example the Deutsch-Jozsa algorithm [25].

A commonly used oracle is constructed by a kind of yes/no query function. Let $\mathcal{M}$ be a set of states satisfying some special conditions, so that the function is

$$f(x) = \begin{cases} 1 & \text{if } x \in \mathcal{M}, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{2.59}$$

The quantum oracle O_f can be used to store the result in the second register when it starts in $|0\rangle$. However, this oracle is normally used to construct an operator which marks the states for which $f(x) = 1$ flipping the sign of the state. To do so, the second register, which in this case is a single qubit, starts in state $|-\rangle$. Since the XOR operation adding 1 is equivalent to the application of a NOT operation, which corresponds to an X gate, and the state $|-\rangle$ is an eigenstate with eigenvalue -1, it turns the second register into $-|-\rangle$ when the state of the first register is marked. Therefore:

$$O_f |x\rangle |-\rangle = |x\rangle \left[(-1)^{f(x)} |-\rangle\right] = (-1)^{f(x)} |x\rangle |-\rangle . \tag{2.60}$$

The minus sign can be attributed to the composite state, rather than just only to the second register. The second register is unchanged after the application of the oracle, and can be considered as an ancilla qubit playing no role in the actual quantum evolution. Therefore, we can consider an operator acting only on the first register. We call this reduced operator the phase-flip oracle, and denote it as Q_f, such that

$$Q_f |x\rangle := \begin{cases} -|x\rangle & \text{if } x \in \mathcal{M}, \\ \\ |x\rangle & \text{otherwise.} \end{cases} \tag{2.61}$$

Although we treat this operator Q_f in a logical sense as an operator acting only on the first register, the actual quantum circuit compilation needs the corresponding oracle O_f and the ancilla qubit, as shown in Figure 2.13.

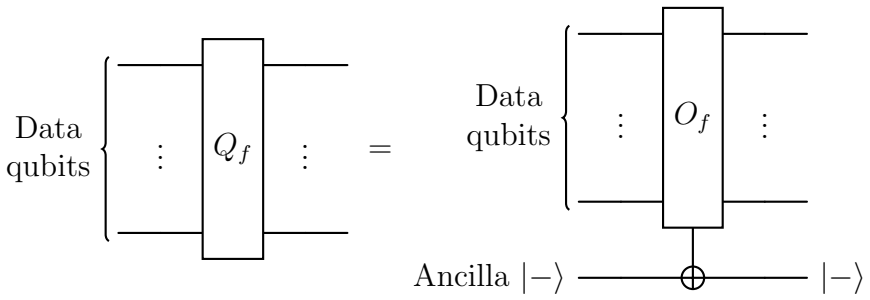


Figure 2.13: *Quantum circuit construction of the phase-flip oracle Q_f from the oracle O_f using an ancilla qubit. Since the ancilla is in state $|-\rangle$, a minus sign is obtained when the NOT gate is applied on it.*

The compilation of the oracle O_f depends on the particular problem, and is based on the classical arithmetic operations of the function $f(x)$. On the one hand, the input x of the function can be the actual instance of the problem, as for example the satisfiability problems where a boolean function is calculated applying logical gates between the bits [97]. On the other hand, the index x can be a reference to a database, so that $f(x)$ takes some data associated to that state in ancilla registers, and uses them to calculates a quite more complicated function. For example, in the context of the quantum SearchRank algorithm [4], we consider in Chapter 7 a function that takes a node index in the internet, reads the information about the page, and checks if it satisfies the search condition of

a user. For simulation purposes, in oracle-based algorithms it is usually considered that the states that are marked by the oracle are known beforehand, and a simple diagonal operator with the action given in equation (2.61) is constructed as:

$$Q_f = \mathbb{1} - 2\sum_{k\in\mathcal{M}} |k\rangle\langle k|\,, \tag{2.62}$$

rather than a complicated operator actually implementing the function $f(x)$. Therefore, the properties of the algorithm can be studied even if the actual implementation of the oracle remains an open problem.

2.3.7. Reset operator

In quantum computers based on qubits, apart from quantum gates, there is an operation denoted as qubit reset that erases the information of a qubit turning it back to state $|0\rangle$ [98]. In this thesis we abstract this idea to a reset operator acting on an arbitrary quantum register, independently of whether the physical system implementing it is based on qubits or not.

This reset operator takes a quantum register in any state and turns it into the computational basis state $|0\rangle$:

$$|\psi\rangle \xrightarrow{\text{Reset}} |0\rangle\,. \tag{2.63}$$

In contrast to the rest of the operators we have defined, this action is irreversible, and therefore is not unitary. It is key for the semiclassical algorithms that we develop in this thesis, which mix unitary and non-unitary operations [2].

Although the reset operation seems quite simple, it is highly nontrivial when a partial reset in a composite system is performed. Without loss of generality let us take a system with two quantum registers, so we can expand an arbitrary state in the computational basis of the first register as:

$$|\psi\rangle = \sum_{i=0}^{N_1-1} a_i\,|i\rangle_1\,|\alpha_i\rangle_2\,, \tag{2.64}$$

where the states $|\alpha_i\rangle$ are arbitrary linear combinations of the computational basis of the second register. The reset operation can be understood virtually as a measurement of the first register, obtaining whatever computational basis state, and a posterior unitary transformation into the state $|0\rangle$ [99, 100]. If all the $|\alpha_i\rangle$ states are equal to a state $|\alpha\rangle$, they can be factorized out of the sum so that the system is not entangled. Therefore, after the measurement of the first register we always get a state $|i\rangle_1\,|\alpha\rangle_2$ for whatever index i, and it is transformed into $|0\rangle_1\,|\alpha\rangle_2$. However, if the registers are entangled, the state of the second register after the reset depends on the result of the measurement of the first register. Although we are interpreting the reset operation this way, the physical operation may not imply an actual measurement, for example if the reset is implemented removing the quantum system of the first register and substituting it by a new one. Therefore, it is as if we performed the virtual measurement without knowing the result, and we had a classical ignorance about it. This results in a mixed state depending on the probabilities of each measurement result of the first register, and after transforming the first register into $|0\rangle$ the global action is as follows:

$$|\psi\rangle \xrightarrow{\text{Reset}} |0\rangle_1\langle 0| \otimes \sum_{i=0}^{N_1-1} |a_i|^2\,|\alpha_i\rangle_2\langle\alpha_i|\,. \tag{2.65}$$

In this thesis all the reset operations are performed on unentangled registers, so we do not have to deal with this issue about creation of mixed states.

2.4. Grover Search Algorithm

The first quantum algorithm that we want to consider in this thesis is the Grover search algorithm. This algorithm, devised in 1996 [26, 27, 95], is at the core of lots of quantum algorithms, as for example quantum walks search algorithms [40, 101] or the quantum SearchRank algorithm [102] which we examine further in Chapter 7.

Given an unsorted list of N elements, the aim of the Grover algorithm is to find an element satisfying some condition. Although the first version of the algorithm was devised for a single marked element, it was sooner modified to allow M different marked elements in the list [103]. Let us denote the set formed by them as $\mathcal{M}$. We are provided with a classical function $f(x)$ such that $f(x) = 1$ if the element x satisfy the condition, and $f(x) = 0$ otherwise. A classical search algorithm would need to apply the function $f(x)$ to each element one by one until it finds a solution. On average that would require $N/2$ calls to the functions if there is a single marked element, so that the complexity would scale as $\mathcal{O}(N)$. If there are M marked elements we would need $\mathcal{O}(N/M)$ iterations to find a marked element.

On a quantum computer we can build an oracle operator O_f that checks in an ancilla qubit if a given computational basis state $|x\rangle$ represents a marked element. Moreover, if we instead provide a quantum superposition of the computational basis, the result of $f(x)$ can be calculated simultaneously for all the list using quantum parallelism. Nevertheless, the probability of measuring a marked state would be the initial one in the superposition. In order to amplify it, so that we obtain a marked node with high probability when measuring, we need to make use of quantum interference with some unitary evolution.

The Grover algorithm starts with a uniform quantum superposition of the N computational basis states, which is created applying Hadamard gates to all the qubits. This state is denoted as $|s\rangle$, so that

$$|s\rangle := \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \,, \tag{2.66}$$

and its amplitudes are represented in Figure 2.14(a). Instead of using the oracle O_f to calculate $f(x)$ on an ancilla qubit, the phase-flip oracle Q_f is applied. Thus, the phase of the marked states is inverted, as shown in Figure 2.14(b). After it, the Grover diffusion operator G_D is applied, which is defined as

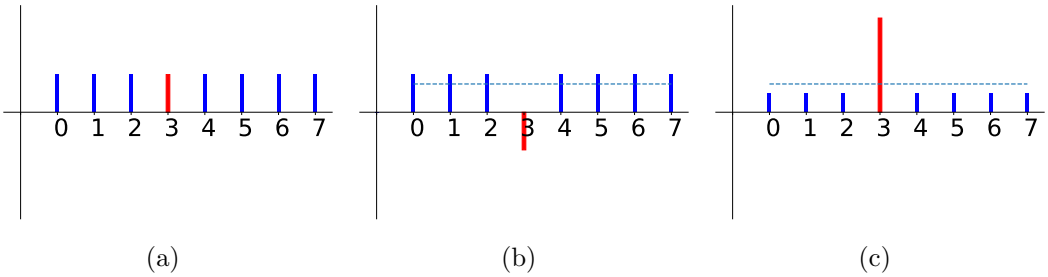
$$G_D := 2\,|s\rangle\langle s| - \mathbb{1}, \tag{2.67}$$

so that it is actually a reflection around the subspace spanned by the state $|s\rangle$. The action of this operator on an arbitrary superposition is:

$$G_D \sum_{i=0}^{N-1} a_i\,|i\rangle = \sum_{i=0}^{N-1} (2\,\langle a\rangle - a_i)\,|i\rangle \,, \tag{2.68}$$

where $\langle a\rangle := N^{-1}\sum_{i=0}^{N-1} a_i$ is the average of the amplitudes. Therefore, this operator performs an inversion about the average [26]. Since the marked elements have a negative amplitude due to the

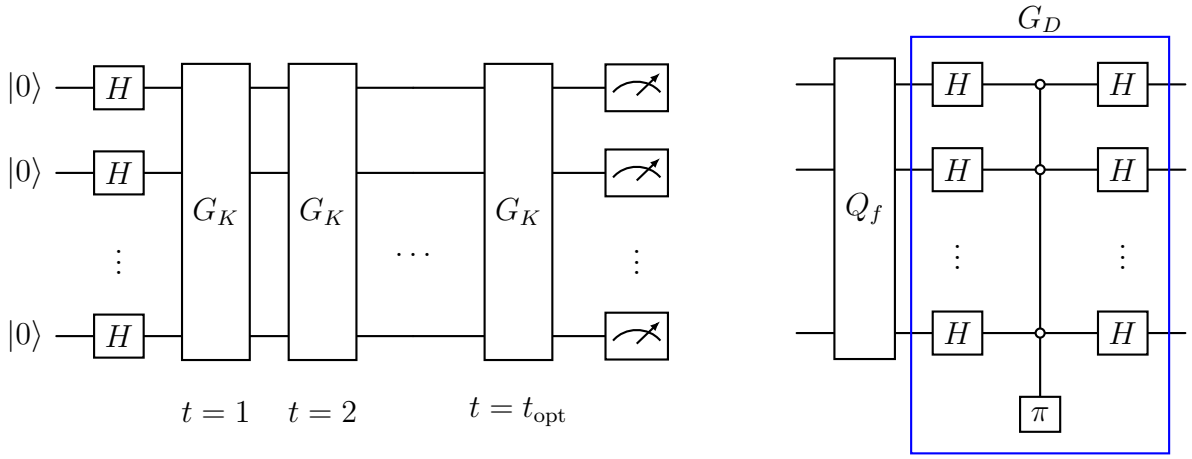
oracle Q_f, after the inversion the amplitude is positive again and it has been enhanced, as shown in Figure 2.14(c).



(a) (b) (c)

Figure 2.14: *Representation of the evolution of a quantum state in the Grover algorithm. (a) The initial state is an equiprobable quantum superposition with positive amplitudes. (b) After the application of the oracle the phases of the marked nodes (red) are inverted. (c) The diffusion operator produces and inversion about the average amplitude, represented with a dashed line. Therefore, the amplitudes of the marked nodes are amplified.*

Let us define the Grover kernel operator $G_K := G_D Q_f$. Then, the Grover algorithm consists of repeatedly applying the kernel G_K a number of time steps t_{opt} such that the probability of measuring a marked state is maximized. A quantum circuit [19] for the algorithm is shown in Figure 2.15.



(a) (b)

Figure 2.15: *(a) Quantum circuit for the Grover algorithm. The initial state $|s\rangle$ is created applying $H^{\otimes n}$ to the computational basis state $|0\rangle$. The kernel G_K is applied the optimal number of time steps and the system is measured. (b) Quantum circuit representation of the kernel G_K. The diffusion G_D corresponds to a reflection around $|s\rangle = H^{\otimes n}|0\rangle$, so it is compiled as a reflection around $|0\rangle$ surrounded by Hadamard gates. Note that this circuit actually implements the minus reflection, so the kernel is applied up to a global phase.*

Due to the unitary character of the kernel G_K, the evolution oscillates instead of converging. Thus, after the optimal time the probability of measuring a marked node starts to decrease. We need therefore to calculate what is the value of the optimal time step. To do so, we use the geometric interpretation of the Grover algorithm in a 2D space [104]. Let us define the vectors $|x_M\rangle$ and $|x_M^\perp\rangle$,

which are equal superposition of the marked and unmarked states, respectively:

$$|x_M\rangle := \frac{1}{\sqrt{M}} \sum_{i \in \mathcal{M}} |i\rangle \,, \tag{2.69}$$

$$\left|x_M^{\perp}\right\rangle := \frac{1}{\sqrt{N-M}} \sum_{i \notin \mathcal{M}} |i\rangle \,. \tag{2.70}$$

These two vectors are perpendicular and form a 2D plane where all the evolution takes place. Let us define a 2D basis as $\mathcal{B} = \{\left|x_M^{\perp}\right\rangle, |x_M\rangle\}$ for this subspace. Note that the superposition of the unmarked vectors is the first vector, and thus plays the role of the X axis, whereas the superposition of the marked vectors is the Y axis. On the one hand, the oracle Q_f flips the phase of $|x_M\rangle$ and lets $\left|x_M^{\perp}\right\rangle$ unchanged. Thus, in this 2D subspace the action is a reflection around the state $\left|x_M^{\perp}\right\rangle$ and can be written as $Q_f^{(2D)} = 2\left|x_M^{\perp}\right\rangle\left\langle x_M^{\perp}\right| - \mathbb{1}_2$. On the other hand, the state $|s\rangle$ lies in this plane, and can be expressed as:

$$|s\rangle = \cos\left(\frac{\theta}{2}\right)\left|x_M^{\perp}\right\rangle + \sin\left(\frac{\theta}{2}\right)|x_M\rangle \,, \tag{2.71}$$

where by convention we consider that it forms an angle $\theta/2$ with the X axis, with

$$\sin\left(\frac{\theta}{2}\right) = \sqrt{\frac{M}{N}}. \tag{2.72}$$

The plane is an invariant subspace of the diffusion operator, which is trivially written in this subspace as $G_D^{(2D)} = 2\,|s\rangle\langle s| - \mathbb{1}_2$.

Taking into account the matricial representation in this reduced basis of $\left|x_M^{\perp}\right\rangle = (1,0)^T$ and $|s\rangle = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\right)^T$, the Grover kernel is expressed as:

$$G_K^{(2D)} = (2\,|s\rangle\langle s| - \mathbb{1}_2)(2\left|x_M^{\perp}\right\rangle\left\langle x_M^{\perp}\right| - \mathbb{1}_2) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \tag{2.73}$$

so it corresponds to a rotation in the plane with angle θ. This is so because the product of two reflections is a rotation. How both reflections produce this rotation is shown in Figure 2.16.

After each iteration, the quantum state is closer to the superposition of marked states $|x_M\rangle$. Since after each iteration the angle formed by the system state and the X axis is increased θ, after t time steps the angle of the system is $\theta/2 + t\theta$. The optimal time for measuring occurs when the system state is perpendicular to the X axis, so that the total angle is $\pi/2$. This occurs for

$$t = \frac{\pi}{2\theta} - \frac{1}{2} = \frac{\pi}{4\arcsin\left(\sqrt{M/N}\right)} - \frac{1}{2} \quad \xrightarrow{N/M>>1} \quad \frac{\pi}{4}\sqrt{\frac{N}{M}}. \tag{2.74}$$

This quantity is in general a non-integer number. Therefore, the optimal number of time steps t_{opt} is the closest integer. In the asymptotic limit, where $N >> M$, we can check that the optimal time grows as $\mathcal{O}(\sqrt{N/M})$, so that the algorithm is able to find a marked element with a quadratic speedup with respect to the classical search. Moreover, there exists a theorem that states that any algorithm based on arbitrary unitary operators interleaved with the oracle cannot provide a better query complexity than $\mathcal{O}(\sqrt{N/M})$, and therefore the Grover algorithm is optimal [105].

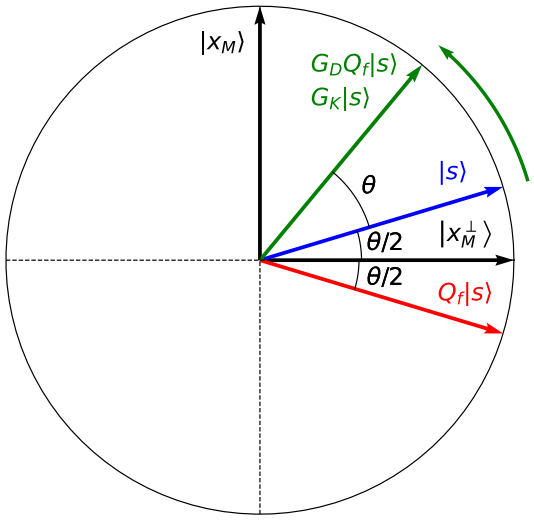


Figure 2.16: *Representation of the first iteration of the Grover algorithm as a rotation. The initial state is $|s\rangle$, forming an angle $\theta/2$ with the X axis. The application of the oracle Q_f produces a reflection around $|x_M^\perp\rangle$, and the posterior application of the diffusion produces a reflection around $|s\rangle$. The final state forms an angle $\theta/2 + \theta$ with the X axis, so that it has been rotated an angle θ.*

The probability of measuring a marked state is given by the amplitude corresponding to the state $|x_M\rangle$ [40]:

$$p_M = \left| \langle x_M | \, G_K^{t_{opt}} \, |s\rangle \right|^2 \geq 1 - \frac{M}{N}. \tag{2.75}$$

Since in general the system state after t_{opt} iterations is not totally parallel to the superposition of the marked states, the probability of measuring them is not exactly 1. Nevertheless, it is always above 1/2 for $M < N/2$, so that repeating the algorithm many times the probability of measuring at least once a marked element is drastically increased. Moreover, in the asymptotic limit $N >> M$ the probability tends to 1. In order to check if a measurement result corresponds to a marked state, we just have to apply the oracle O_f that calculates $f(x)$ on an ancilla qubit, or the classical function $f(x)$ on a classical computer if we dispose of it. This checking post-procedure does not alter the computational complexity of the algorithm since the oracle is only applied once for each run of the algorithm, independently of the size N of the list.

Note that the quadratic speedup of the quantum search algorithm has been obtained considering only the number of steps of the algorithm. This is called the query complexity, since it depends on the number of calls to the function $f(x)$ or its corresponding quantum oracle O_f [106]. It is a kind of complexity intrinsic to the algorithm. Nevertheless, there is an extrinsic part that must be taken into account, which depends on the particular implementation of each time step of the algorithm.

In the classical search, the extrinsic complexity depends on the implementation of the function $f(x)$. Usually, we can assume that the cost is polynomial in the number of bits n, and thus scales polylogarithmically with the size $N = 2^n$ of the list. In the case of the quantum search, the oracle O_f is constructed based on the classical arithmetic operations of the function $f(x)$, and thus the number of quantum gates is also expected to increase polylogarithmically with N. The quantum implementation also depends on the diffusion operator G_D. This requires $2n$ Hadamard gates, and the reflection operator around the state $|0\rangle$. As we saw in Section 2.2.2.3, the multi-controlled gate

can be decomposed with a linear cost in the number of qubits. Therefore, the cost of the diffusion operator is $\mathcal{O}(\log_2(N))$. The total complexity of an algorithm is the product of the intrinsic and the extrinsic complexities, and since both the classical and quantum extrinsic complexities scale polylogarithmically, the quadratic speedup is maintained.

The Grover algorithm has been widely studied in the literature, and there are lots of modifications to improve it. An important modification is the introduction of arbitrary phase rotations, replacing the reflections by APR operators [96, 107, 108]. The complex phases can be matched in a way that the Grover algorithm becomes deterministic, and thus the probability of measuring a marked element becomes 1 at an integer measurement time [74, 109]. Moreover, there are other algorithms that employs different phases in a manner such that for a wide range of different ratios N/M the probability of measuring a marked node is above a threshold using a fixed number of time steps. Thus, it allows a certain amount of unknownledge about the number of marked states M [110–112].

2.4.1. Quantum counting

Sometimes, we are interested in determining the number M of solutions to $f(x) = 1$ given an oracle O_f, for example in order to calculate the optimal number of time steps of the Grover algorithm. To do so, there is a quantum algorithm called quantum counting that determines M leveraging the theory of the Grover algorithm [103, 113–115]. This algorithm employs quantum phase estimation (QPE) [116], which returns the phase of the eigenvalue of a unitary operator when its corresponding eigenvector is given. The deeper details of this subroutine are beyond the scope of this thesis. However, we sketch how quantum counting is performed for completeness.

We perform QPE for the Grover kernel operator G_K, which is formed by the problem oracle O_f. As input state we provide the uniform superposition of all the computational basis states $|s\rangle$. This state is not an eigenvector of the kernel. However, since G_K corresponds to a rotation in a 2D plane as shown in (2.73), and $|s\rangle$ lies in this plane, it is a linear combination of only two eigenvectors with eigenvalues $e^{\mathrm{i}\theta}$ and $e^{-\mathrm{i}\theta}$. Therefore, QPE returns a value $\widetilde{\theta}$ when measuring, which can be either θ or $2\pi - \theta$. The greater value of θ is π when $M = N$. Therefore, we can always know if we are measuring θ or $2\pi - \theta$. Anyway, since $\sin^2\left(\frac{\theta}{2}\right) = \sin^2\left(\frac{2\pi-\theta}{2}\right)$, we can use the direct measured result $\widetilde{\theta}$ to obtain M from (2.72) as:

$$M = N \sin^2\left(\frac{\widetilde{\theta}}{2}\right). \tag{2.76}$$

A quantum circuit for the quantum counting algorithm [19] is shown in Figure 2.17. It employs a subroutine known as quantum Fourier transform (QFT) [24, 28], whose circuit is also shown but the precise details are beyond the scope of this thesis. The time complexity of quantum counting scales as $\mathcal{O}(\sqrt{N})$ [19]. Since the time complexity of two independent algorithms is added rather than multiplied, the total scaling of Grover algorithm considering a previous quantum counting holds the quadratic speedup.

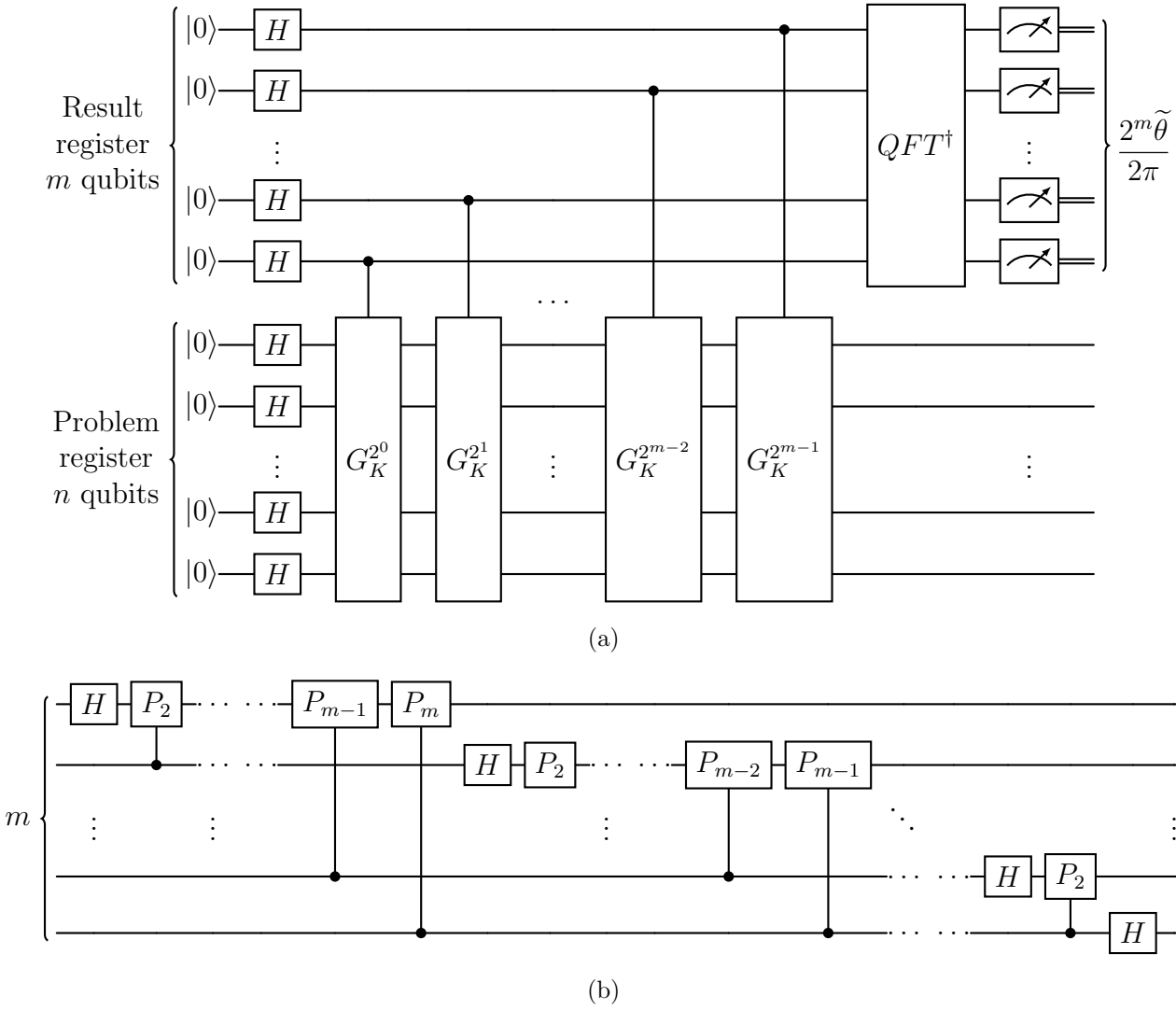


Figure 2.17: *a) Quantum circuit for the quantum counting algorithm. The input state $|s\rangle$ is created in the problem register with Hadamard gates. b) Quantum circuit for the quantum Fourier transform subroutine. The gate P_k corresponds to a phase gate with angle $2\pi/2^k$.*

Chapter 3

Quantum Walks

Random walks are stochastic processes representing Markov chains on the nodes of a graph [41,117]. They have resulted a useful tool for constructing algorithms with lots of applications. For example, simulation of Brownian motion [118], sampling distributions [119, 120], and optimization problems [121]. In the era of quantum computing, a quantization of these random walks is expected to provide a new and wide set of algorithms outperforming the classical ones. This idea has given raise to the concept of quantum walks or quantum Markov chains [40]. They were first proposed in the discrete time version [39], and later using a continuous time [122]. However, precursor ideas can be attributed to Feynman [123].

An advantage of quantum walks is that they can explore a state space polynomially faster than their classical counterparts, for example, spreading quadratically faster [40,124]. Moreover, there are examples with exponential speedup in some graphs as the hypercube [125]. Due to their properties, they have been used to develop a wide variety of algorithms for problems such us triangle finding [42], element distinctness [43] and quantum search [44]. Moreover, quantum walks can simulate a lot of physical systems [40], and can be used as a universal quantum computing paradigm [126, 127].

Quantum walks in discrete time usually require an inner degree of freedom, leading to what is called the coined quantum walk model [128]. These walks act on undirected graphs, and have difficulties quantizing arbitrary Markov chains. To solve this issue, Szegedy introduced a coinless quantum walk on bipartite graphs based on precursor algorithms of Ambainis [43] and Watrous [129], quantizing a general Markov chain on weighted graphs by a duplication process [45]. Soon after Szegedy's work, an alternative version of the Szegedy quantum walk avoiding the need to duplicate the graph [90] was developed. This formulation produces more general results for classical weighted graphs, and moreover, allows establishing an equivalence between the Szegedy quantum walk and the coined model [130–132].

The Szegedy quantum walk was first used for detecting the presence of marked elements on a graph with a quadratic speedup [45,133], and later it was generalized to the search problem [134,135]. Since its development, there has been much research on its applications, resulting useful for optimization [46–50], testing graph completeness [51], classification [31, 32], quantum search [40, 101, 102, 117] and machine learning [52]. Moreover, there has also been research in implementing this algorithm in quantum circuits [91, 94].

This chapter is devoted to provide an introduction to the topic of quantum walks, setting the formalism that we use in this thesis. We start in Section 3.1 by introducing Markov chains and random walks, as well as the mathematical formulation of graphs. Then, we introduce quantum walks in Section 3.2. We do it defining the computational space in a general manner, independently of the quantum computing model chosen to represent it. Te first quantum walk that we study is the coined quantum walk in Section 3.3, followed by the Szegedy quantum walk in Section 3.4. We analyze the equivalence between both models in Section 3.5, and show the implementation with quantum circuits in Section 3.6. In Section 3.7 we define properly the computational complexity of the quantum walk algorithms. In Section 3.8 we briefly show other quantum walks. In Section 3.9 we show how the Grover search algorithm can be obtained from the quantum walk perspective. Finally, in Section 3.10 we show the PageRank algorithm and its quantization, which we use to test our new walk algorithms in this thesis.

3.1. Markov Chains, Graphs and Random Walks

A Markov chain [41, 136] is a stochastic process that assumes values in a discrete set $\mathcal{X}$, so that a sequence of states obeys that the next state depends only on the current state. On the one hand, in this thesis we only consider discrete-time Markov chains, so that the chain is formed by a discrete set of states. On the other hand, although the size of the set can be in principle infinite, for simplicity we can consider, without loss of generality, a finite size with N possible states. Let x_t be the state of the chain at time step t, and consider a chain of states $x_t, x_{t-1}, \ldots, x_0$. Then, the probability of being at time step t in the state $x_t = j$ conditioned on the past states being $x_{t-1} = i, \ldots, x_0 = k$ is actually conditioned only by the immediate past state x_{t-1}:

$$\mathbb{P}(x_t = j | x_{t-1} = i, \ldots, x_0 = k) = \mathbb{P}(x_t = j | x_{t-1} = i). \tag{3.1}$$

Therefore, it is as if the chain would not have memory about all the history, and it only matters the current state.

When these probabilities do not depend on the time step, so that $\mathbb{P}(x_t = j | x_{t-1} = i) = \mathbb{P}(x_{t'} = j | x_{t'-1} = i)$ for all pairs t, t', the chain is said to be time homogeneous, being the case for all the chains considered in this thesis. To characterize a Markov chain further let us introduce the following properties [136, 137]:

- Irreducibility: A Markov chain is irreducible if for each pair of states x_1 and x_2, the system can form a chain from x_1 to x_2 with an arbitrary number t of time steps. Therefore, each state is reachable from any other.
- Aperiodicity: The period of a state is defined as the greatest common divisor of the set of times when it is possible for the chain to return to the starting position x. A Markov chain is aperiodic if all the states have period 1.
- Ergodicity: A Markov chain is ergodic if it is both irreducible and aperiodic.

Associated to any Markov chain there is a transition matrix G, whose elements G_{ji} are the probabilities of transitioning from state $x_{t-1} = i$ to $x_t = j$. By construction this matrix is column-

stochastic[1], so that all the columns add up to 1. Thus:

$$\sum_{j=0}^{N-1} G_{ji} = 1. \tag{3.2}$$

This matrix allows the simulation of the Markov chain as a random walker jumping stochastically in the nodes of a graph. Before going on with random walks [117], let us introduce the fundamental concepts of graph theory needed in this thesis.

A graph $\Gamma(\mathcal{V}, E)$ consists of a non-empty finite set $\mathcal{V}$ of elements called nodes or vertices, and a finite set E of edges connecting the nodes [138, 139]. Depending on the nature of the edges we can define two types of graphs:

Definition 3.1 (Undirected graph)**.** *It is a graph where each edge connects two nodes in a symmetric way. The edges are called undirected edges. The degree of a node is the number of edges connecting it.*

Definition 3.2 (Directed graph or digraph)**.** *It is a graph where each edge connects two nodes in an asymmetric way, so that they have an arrow indicating a direction. These edges are called directed edges, and sometimes they are also referred to as arcs or links. The indegree of a node is the number of arcs pointing to it, whereas the outdegree is the number of arcs outgoing from it.*

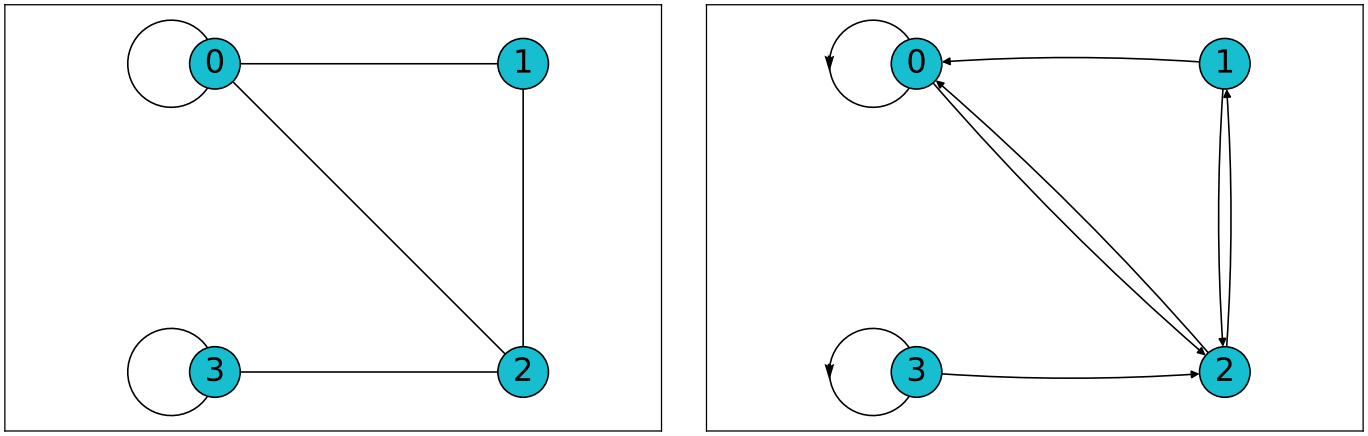
Examples of these two kind of graphs with four nodes are given in Figure 3.1. In this thesis we consider simple graphs, so that there are no repeated undirected or directed edges. Therefore, whereas for the undirected graph two distinct nodes can only be joined by a simple line, in directed graphs we can have two arcs between each pair of nodes, since there are two possible directions. When for each directed edge (i, j) in a digraph connecting from node i to node j, there exists the inverse arc (j, i) from node j to node i, the digraph is said to be symmetric. Moreover, an undirected graph can be understood as a symmetric digraph, so that for each undirected edge between two different nodes we have the two possible arcs. In the case of an undirected self-loop, it has only an associated directed edge, since it goes from a node to itself.

Associated to each graph there is a boolean matrix known as the adjacency matrix A, satisfying that $A_{ji} = 1$ if and only if there is an arc connecting from node i to node j. In the case of an undirected graph this matrix is necessarily symmetric. The adjacency matrices A_u and A_d for the undirected and directed graphs of Figure 3.1, respectively, are:

$$A_u = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad A_d = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.3}$$

Although a directed graph with N nodes represents an appropriate place where a walker could represent a Markov chain with N possible states as a random walk, this graph is only a kind of backbone with no information about the transition probabilities. We need to define another kind of graph taking into account all the Markov chain information [140]:

[1] In the literature it can be found a different convention so that the transition matrix is row-stochastic [40], so that it corresponds to the transposed matrix of our convention.



(a) (b)

Figure 3.1: *(a) Undirected graph with $N = 4$ nodes. (b) Directed graph with $N = 4$ nodes. Te adjacency matrices are given in (3.3). Note that whereas the directed graph is asymmetric, the underlying undirected graph is symmetric. For example, in the directed graph there is a directed edge from node 1 to node 0, but there is no directed edge from 0 to 1. This asymmetry is not reflected in the undirected graph, where a symmetric edge is present.*

Definition 3.3 (Weighted graph)**.** *It is a directed graph where each directed edge or arc has associated a real number. This real number can be understood as a probability, so that all the arcs outgoing from the same node add up to* 1*.*

An example of weighted graph is shown in Figure 3.2. Instead of associating a boolean adjacency matrix, we associate to weighted graphs directly the transition matrix G of a Markov chain. Therefore, the graph represents properly all the information for performing a random walk representing the Markov chain. The transition matrix for this example weighted graph is

$$G = \begin{pmatrix} 0.7 & 0.3 & 0.4 & 0 \\ 0 & 0 & 0.6 & 0 \\ 0.3 & 0.7 & 0 & 0.7 \\ 0 & 0 & 0 & 0.3 \end{pmatrix}. \tag{3.4}$$

Although a random walk occurs necessarily on a weighted graph, sometimes we are given a random walk on an undirected or directed unweighted graph. In those cases we consider that they are actually weighted graphs and that the transition matrix G is obtained by normalizing the adjacency matrix A, so that we divide each column by the corresponding outdegree of the node. Therefore, a walker in each node has the same probability of jumping to any of the nodes it points to.

From a functional point of view, a random walk algorithm is a stochastic process, so that it can be implemented with a Monte Carlo simulation. At each time step the walker is at only one node of the graph, it tosses a (biased) coin[2] to decide what path to follow given the transition probabilities,

[2]Whereas the term "coin" is appropriated for unidimensional graphs, where there are only two possible directions, for general graphs a more appropriated term would be "dice". However, the term "coin" is fixed in the literature and is implicitly understood as a dice.

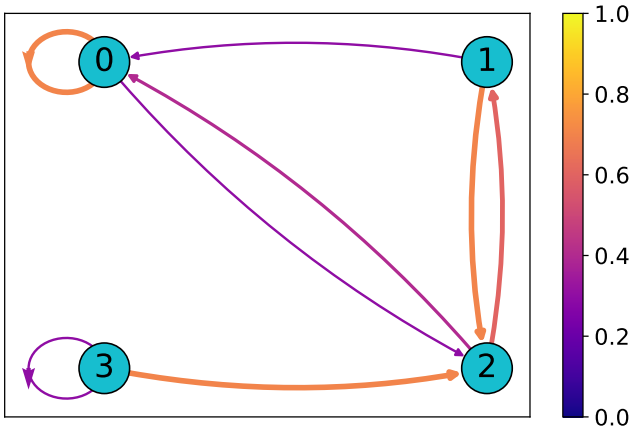


Figure 3.2: *Weighted graph with $N = 4$ nodes whose transition matrix is given in (3.4). The weights of the directed edges are represented by the colormap, and are proportional to the width of the edges. Missing edges have a null transition probability.*

and then jumps to a neighbor node. Since this process is stochastic, each time we repeat the random walk the trajectory followed by the walker is different. Averaging over the different trajectories we could obtain a probability distribution for the walker being at each node at each time step.

From other point of view, if we knew beforehand all the elements of the transition matrix G, then we could simulate deterministically the probability distribution of the walker being at each node. Let $p(t)$ be a column vector whose elements $p_i(t)$ are the probabilities of the walker being at node i at time t. Then, given an initial probability distribution $p(0)$, the probability distribution can be simulated deterministically as:

$$p(t) = G^t p(0). \tag{3.5}$$

Both points of view for random walks have applications depending on whether we want to obtain a specific position of the walker or the entire probability distribution. For example, the stochastic simulation is used for optimization algorithms like simulated annealing with Metropolis-Hastings, where it is wanted to obtain a single node as an optimal solution [121]. The search space can be so big that it is unfeasible to calculate the entire transition matrix. Thus, only the probabilities of the current node at each time step are calculated, reducing the computational cost of the algorithm. Other example is the PageRank algorithm, where the objective is the limiting probability distribution of the walker for classifying the nodes of the graph [141–144], so that a deterministic simulation using the complete transition matrix is performed.

3.2. Quantum Walks

In this section, we introduce the concept of quantum walks [39, 40], which are the quantization of random walks. Since quantum algorithms are indeed also random, in order to clearly differentiate between quantum and classical algorithms, from now on we denote random walks as classical walks.

In contrast to classical walks, where the walker is at a single node at a time, in a quantum walk

the walker can be in a superposition of the nodes of the graph, so that the position at each time step is represented by a quantum state $|\psi(t)\rangle$. Although the quantum evolution can be in discrete or continuous time, in this thesis we mostly deal with discrete-time quantum walks. The evolution of the system is therefore given by a unitary matrix U applied a discrete number t of times, so that

$$|\psi(t)\rangle = U^t |\psi(0)\rangle . \tag{3.6}$$

The position of the walker is obtained measuring the state after the quantum evolution has finished. The Hilbert space is usually more complex than just the one formed by the computational basis associated to the node indexes, and depends on the quantum walk model. However, for simplicity, so far let us consider that the state is a simple linear combination of the computational basis $\mathcal{C} = \{|i\rangle , i = 0, \ldots, N-1\}$. Then, the probability that the quantum walker can be measured at each node at time step t is

$$p_i(t) = |\langle i|\psi(t)\rangle|^2 . \tag{3.7}$$

On the one hand, in principle quantum walks are though to be implemented on real quantum hardware. In this real scenario, after measuring, the quantum state results in a state of the computational basis with a certain probability, being analog to the stochastic simulation of the classical walk. This is useful for example in the case of the quantum Metropolis algorithm for optimization problems [46–50]. If we wanted the probability distribution, we would have to repeat the walk several times and average the results. Moreover, since the quantum superposition is broken after the measurement, we cannot measure at intermediate steps of the walk and resume it. If we wanted the probability distribution at each time step, we would have to perform a different quantum walk for each final time. On the other hand, we could perform a deterministic simulation on a classical computer provided that we can construct and store the unitary matrix U, or just simulate its action on a quantum state. This would be useful for algorithms where the result is the entire probability distribution, as for example the quantum version of the PageRank algorithm [31, 32]. However, in some cases it is very costly to classically simulate the quantum walk (see Section 3.7).

3.3. Coined Quantum Walk

The first quantization model we consider in this thesis is the coined quantum walk [39, 128]. This algorithm is inspired in the coin tossing of a classical walker, so that it is performed in superposition in the quantum context by a unitary operator.

In contrast to classical walks, in order to perform a coined quantum walk we need an undirected graph or symmetric digraph. Although self-loops are not usually considered, without loss of generality we consider them in this thesis. Thus, we provide a construction of coined quantum walks on arbitrary graphs [40] considering also self-loops. This construction uses the arc notation [145], so that the basis of the Hilbert space is formed by states representing directed edges of the graph. Therefore, the Hilbert space where the coined quantum walk takes place is

$$\mathcal{H}_C := \text{span}\{|(i,j)\rangle \ : \ (A_u)_{ji} = 1\}, \tag{3.8}$$

where (i,j) represents the directed edge pointing from node i to node j. The dimension is $2|E_U|+|E_L|$, where E_U is the set of undirected edges between different nodes, and E_L the set of self-loops. For

the example of Figure 3.1(a), $E_U = 4$ and $E_L = 2$. Note that due to the symmetry of the undirected graph, if $|(i,j)\rangle$ is in the Hilbert space, then $|(j,i)\rangle$ is too.

For the quantum evolution we need a coin operator C that performs the quantum coin tossing, and a shift operator that updates the position of the quantum walker depending of what node is directed to. For a general coined quantum walk on an undirected graph, the flip-flop shift operator S_f is usually used. Its action is defined by [146]

$$S_f \left|(i,j)\right\rangle := \left|(j,i)\right\rangle, \tag{3.9}$$

so that the walker ends up in the node the arc points to, and after that the walker turns around pointing back to the previous node. The unitary evolution operator U_c of the coined quantum walk is then defined as follows:

$$U_c := S_f C. \tag{3.10}$$

The coin operator C is expressed as:

$$C = \bigoplus_{i=0}^{N-1} C_i, \tag{3.11}$$

so that is a block-diagonal matrix, and in general there is a different coin block C_i associated to each of the N nodes of the graph. Thus, depending on the node where the walker is, which is represented by the tail of the directed edge, the coin acts differently. Moreover, in general each C_i matrix has a different dimension. Each C_i operator is a d_i-dimensional unitary operator, where d_i is the degree of node i, and acts in the subspace

$$\mathcal{H}_C^i := \text{span}\{|(i,k)\rangle \ : \ (A_u)_{ki} = 1\}, \tag{3.12}$$

which is formed by the d_i directed edges departing from node i. Usually, the Grover diffusion operator G_D [26], which is the most distant from the identity, is used as coin [40, 147], so that the matrix elements are

$$(C_i)_{ab} = \frac{2}{d_i} - \delta_{ab}. \tag{3.13}$$

Note that in general each node connects to a different set of nodes, so that each coin block C_i is different despite the fact of all them being a Grover coin. If we want to quantize a walk on an asymmetric digraph or a weighted graph, the state space must also contain all the edges considering the undirected graph due to the flip-flop operator, and it is the coin who has to take into account somehow the lack of an arc in one of the two directions of the edge, or the different transition probabilities. As we will see in Section 3.4, Szegedy's quantization provides a natural recipe for constructing such a coin, being indeed able to consider arbitrary weights.

The probability of measuring the walker at each node is not straightforwardly obtained by projecting into the computational basis of the nodes given the construction of the quantum states as arcs. In this case the graph position associated to each arc is the node it departs from, whereas the arriving node is an additional inner degree of freedom. Therefore, we have to sum over all the arcs departing from a particular node. The probability of being at node i is

$$p_i = \sum_j \left|\langle (i,j)|\psi\rangle\right|^2. \tag{3.14}$$

In the case we want to construct a quantum circuit, it depends on the particular structure of the graph and the coin operators C_i, so that it is in general an open problem. Some quantum circuits for different kinds of graphs can be found in the literature [92]. In order to build a circuit, we usually need to express the basis of arc states as tensor products of a position state and a coin state indicating the possible direction [40]:

$$|(i,j)\rangle = |i\rangle_P \, |j\rangle_C \,, \tag{3.15}$$

where the first register is the position register and the second one is the coin register. For doing so, we need that each node has the same degree d, so that the Hilbert subspace of the register associated to the coin inner degree of freedom has the same dimension d for all the nodes. This is the case of d-regular lattice graphs. However, for a general graph it is not straightforward since each node has a different degree. A solution would be augmenting the Hilbert space $\mathcal{H}_C$ in (3.8) into an N^2-dimensional space, considering also the edges (i,j) that are not present in the undirected graph. In this augmented space each coin register would be N-dimensional, and then we can express each basis state as a tensor product. The details about how the unitary evolution operator U_c is augmented, so that the original subspace $\mathcal{H}_C$ is invariant under its action, will be shown in Section 3.5, where we establish an equivalence with the Szegedy quantum walk. Moreover, we provide some guidelines for constructing a circuit for the Szegedy quantum walk in Section 3.6, which can serve for a general coined quantum walk when both models are equivalent.

3.3.1. Example on the 1D line

As an example of a quantum walk we show a quantization on the infinite one-dimensional line. The corresponding undirected graph is shown in Figure 3.3. Since it is a regular lattice, each node i connects only to nodes $i \pm 1$, and the Hilbert space can expressed as a tensor product where the coin register has dimension 2. Let us define the computational basis of the second register as[3] $\{|R\rangle, |L\rangle\}$, where R (L) indicates that an arc departing from node i is directed to the right (left), i.e., to node $i+1$ ($i-1$). Therefore, the Hilbert space is

$$\mathcal{H}_C = \text{span}\{|(i,i+1)\rangle, |(i,i-1)\rangle \;:\; i \in \mathbb{Z}\} = \text{span}\{|i\rangle_P \, |R\rangle_C, |i\rangle_P \, |L\rangle_C \;:\; i \in \mathbb{Z}\}. \tag{3.16}$$

The first quantum walk on the line was done using the Hadamard gate in (2.38) as the coin operator for all the nodes [124], so that the coin operator C is

$$C = \mathbb{1} \otimes H. \tag{3.17}$$

The original shift operator for this walk was the moving shift operator, defined as:

$$S_m := \mathcal{P}^+ \otimes |R\rangle\langle R| + \mathcal{P}^- \otimes |L\rangle\langle L| \,. \tag{3.18}$$

The permutation operator $\mathcal{P}^+$ ($\mathcal{P}^-$) is applied when the coin is in the state $|R\rangle$ ($|L\rangle$), so that the position is displaced one position to the right (left), letting the coin state unchanged.

[3] In the case of representing the coin register with a qubit, we would have $|R\rangle = |0\rangle$ and $|L\rangle = |1\rangle$. However, since it can create confusion with the node indexes 0 and 1 in the arc notation, we take the special names R and L in this particular context.

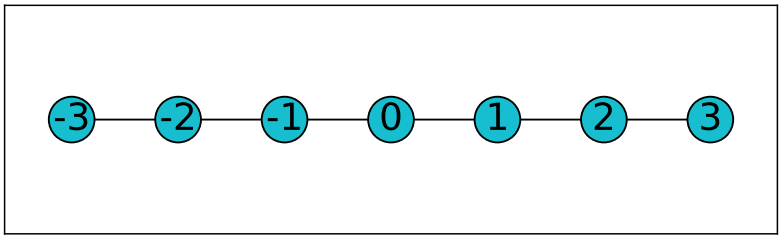


Figure 3.3: *Undirected graph for an infinite 1D line. For the sake of simplicity only seven nodes have been represented.*

In this thesis we are mostly interested in quantum walks on general graphs, using the flip-flop shift operator instead. In this Hilbert space it can be expressed as [44]:

$$S_f = \mathcal{P}^+ \otimes |L\rangle \langle R| + \mathcal{P}^- \otimes |R\rangle \langle L| , \tag{3.19}$$

so that the same displacement is performed according to the coin state. However, the state of the coin is flipped so that the walker points back to the previous node, reproducing the effect of the flip-flop shift operator in (3.9). Note that both shift operators can be easily interconverted with an X Pauli gate, since $(\mathbb{1} \otimes X)S_m = S_f$. The X gate can be reabsorbed into the coin operator C of the next step, so that a walk using a shift operator can be converted into the other by a redefinition of the coin and applying a previous X gate to the initial state. Therefore, both quantizations are in some sense equivalent and we can deal directly with the flip-flop shift. Moreover, for the Hadamard coin the results are quite similar even without the redefinition of the coin.

The reason behind the use of the Hadamard coin is that it can put the coin register in a superposition state. For example, let us suppose that the walker starts at node 0 in the initial state

$$|\psi(0)\rangle = |0\rangle_P |R\rangle_C . \tag{3.20}$$

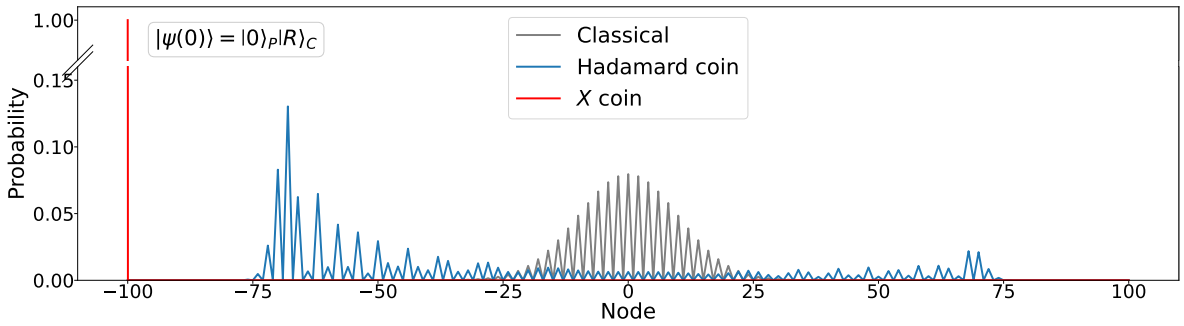
After the application of the Hadamard coin, it is converted to

$$(\mathbb{1} \otimes H) |\psi(0)\rangle = |0\rangle_P (|R\rangle_C + |L\rangle_C)/\sqrt{2}. \tag{3.21}$$

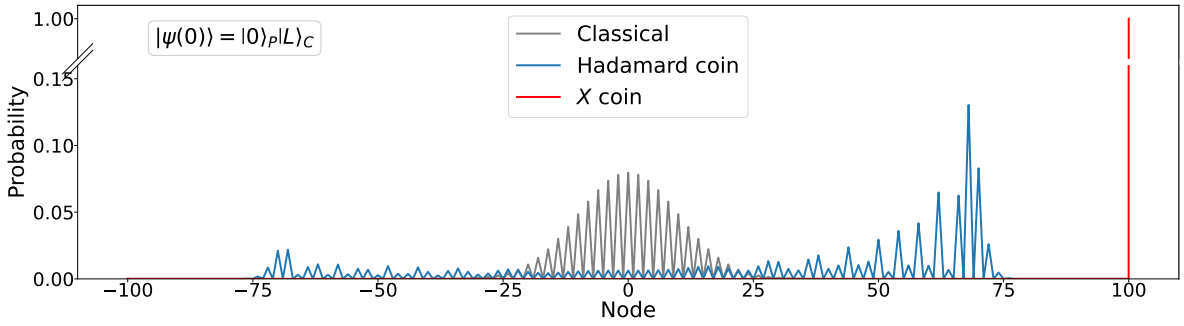
Therefore, the walker points to both directions with an equal probability, being equivalent to a coin tossing of the classical walker on the undirected 1D line. After the shift operator S_f we obtain the complete action of the unitary evolution operator:

$$U_c |\psi(0)\rangle = (|1\rangle_P |L\rangle_C + |-1\rangle_P |R\rangle_C)/\sqrt{2}. \tag{3.22}$$

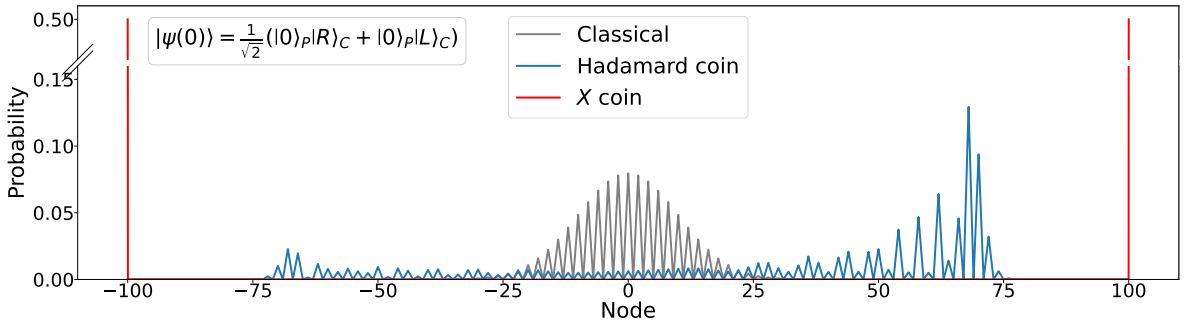
Thus, there is a probability of one half of measuring the position either at node 1 or node -1, as in the classical walk. Although so far the quantum walk reproduces the classical one, this only occurs for the first step. After this, since in a quantum walk we are not measuring until the end, the state remains in superposition and there are interference effects in the successive steps, so that the distribution obtained at the final time step is completely different. We have chosen the initial coin state this way to reproduce the first classical step. However, in quantum walks we have freedom to choose the initial coin state for the same equivalent classical initial distribution. For example, the coin could start in a superposition state, so that the first Hadamard coin let the walker pointing only in one direction. Therefore, different initial coin states give rise to different quantum walks.



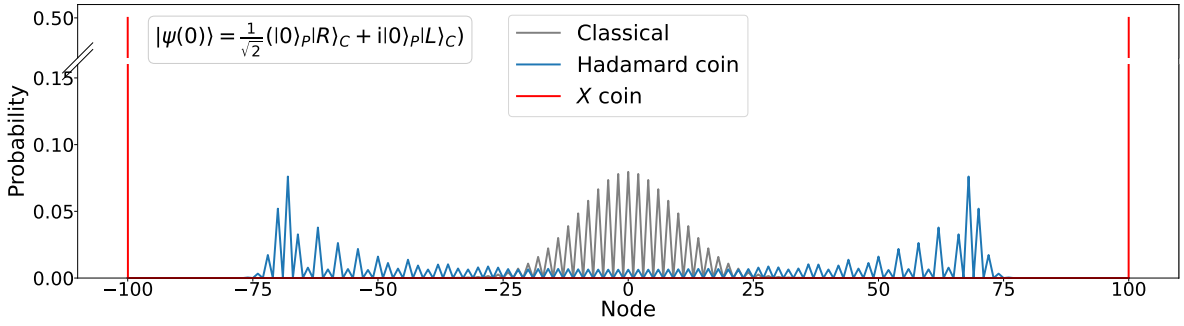
(a)



(b)



(c)

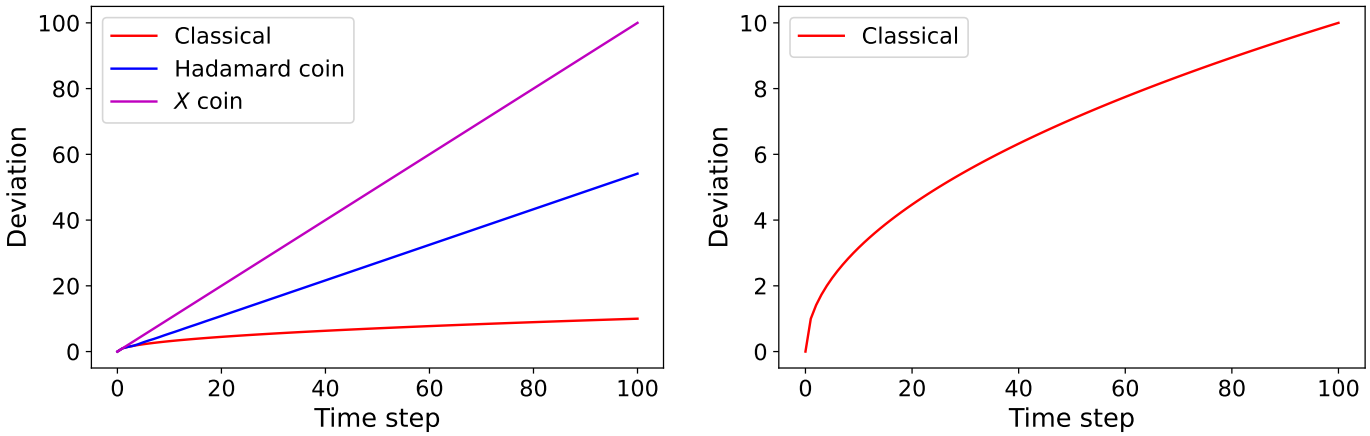


(d)

Figure 3.4: *Probability distribution results after* 100 *time steps of the coined quantum walk on a 1D line using the X coin (red) and the Hadamard coin (blue) for different initial coin states at node* 0*, compared with the classical walk (gray).*

Examples of the probability distributions obtained for simulations starting with different initial coin states during 100 time steps are shown in Figure 3.4. In Figure 3.4(a) we observe that we do not obtain a symmetric pattern as in the classical walk, since it is biased to the left. This is curious since the initial state pointed to the right, and it is due to the fact of being using the flip-flop shift instead of the moving shift operator. When the initial state points to the left, the mirrored distribution is obtained as shown in Figure 3.4(b). It is also curious that when the initial coin state is the superposition $(|R\rangle_C + |L\rangle_C)/\sqrt{2}$, the quantum walk keeps biased to the right, as shown in Figure 3.4(c). This is due to interference effects, which can be broken introducing a relative complex phase in the superposition, obtaining a symmetric distribution as shown in Figure 3.4(d).

We have seen previously that a usual coin for general graphs is the Grover coin. For a two-dimensional Hilbert space it corresponds to the X gate. The results for the simulations using this coin are also shown in Figure 3.4. In this case the walker moves completely ballistically, so that there are no probabilities at intermediate nodes, and the walker reach nodes 100 and / or -100. A notable difference with the Hadamard walk is that for the superposition initial coin the distribution is unbiased, so that it moves equally to both directions. We will analyze what coin provides more sensible results later in the context of the Szegedy quantum walk.



(a) (b)

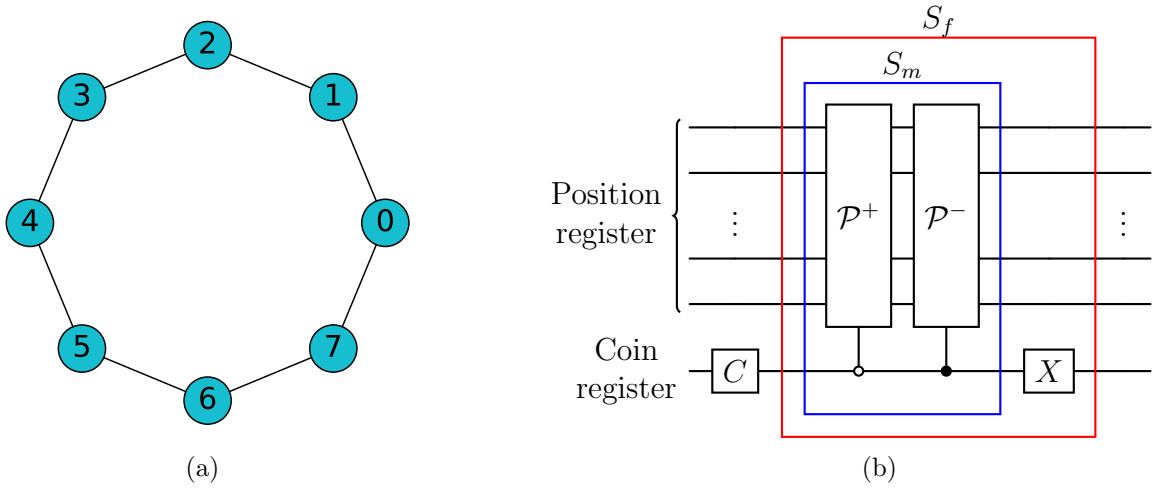
Figure 3.5: *(a) Standard deviation of the probability distribution from the origin at different time steps of walks in the infinite 1D line. (b) Zoom for the classical walk line. The classical walk spreads square-root-like, whereas the coined quantum walk spreads linearly, being faster with the Grover X coin.*

As we mentioned at the beginning of this chapter, quantum walks provide advantage with respect to classical algorithms spreading quadratically faster in the space. This quantum walk is a good example of that fact [40,124]. As we can see, the classical walk produces a more centered distribution, spreading slower. We can calculate the deviation of the probability distribution from the origin for each time step as:

$$\sigma(t) = \sqrt{\sum_{i\in\mathbb{Z}} i^2 p_i(t)}. \tag{3.23}$$

It is represented in Figure 3.5 for the different walks. We find that for the classical walk it grows as $\sqrt{t}$, whereas for the quantum walk it grows linearly, being approximately $0.54t$ for the Hadamard coin, and exactly t for the Grover coin. Therefore, for the X coin the spreading is the faster one.

Finally, let us construct a quantum circuit for this quantum walk. A quantum walk in the infinite 1D line can be simulated in a finite 1D cycle, as long as the time steps do not surpass a threshold from which the wavefront interferes with itself when it completes a turn on the cycle. An example of a cycle graph is shown in Figure 3.6(a). For a graph with $N = 2^n$ nodes, we need n qubits for the position register, and an additional qubit for the coin register. The quantum circuit is shown in Figure 3.6(b). Since the flip-flop shift operator can be obtained from the moving shift just adding an X gate, we can leverage the circuit of the moving shift walk [92]. First, we apply the coin to the coin register, with no controls since it is the same for all the nodes. After that, we apply the moving shift operator S_m, which is composed of the permutation operators controlled by the coin register. Finally, we convert it into the flip-flop shift operator S_f with an X gate in the coin register.



(a) (b)

Figure 3.6: *(a) Cycle graph with $N = 8$ nodes. (b) Quantum circuit for the coined quantum walk on the cycle graph for a generic coin C. The flip-flop shift operator S_f is compiled adding an X gate in the coin register to the circuit of the moving shift operator S_m. The black dot represents a control by the state $|L\rangle$ and the white dot by the state $|R\rangle$.*

3.4. Szegedy Quantum Walk

Whereas usual quantum walks in discrete time are based on the additional inner degree of freedom associated to a coin, Szegedy proposed a coinless quantum walk on bipartite graphs, which can be straightforwardly used to quantize any Markov chain on a weighted graph [45, 133]. However, soon after an alternative formulation avoiding the need of a bipartite graph was proposed, which resembles again a coined quantum walk while quantizing a weighted graph in a natural form [90]. In this thesis, we only deal with the alternative formulation, which provides more general results. However, for completeness, we first review the original formulation to later introduce the coined form.

3.4.1. Original formulation

In order to perform a Szegedy quantum walk we need a weighted bipartite graph [45]. This graph is formed by two set of nodes, $\mathcal{V}_1$ and $\mathcal{V}_2$, of sizes N_1 and N_2, respectively. At the same time there are two sets of directed edges, E_1 and E_2. The edges from E_1 depart from nodes in the first set and arrive to nodes in the second set. For the set E_2 the edges behave the other way around. Therefore, there are no edges connecting nodes inside the same set. An example of weighted bipartite graph is shown in Figure 3.7. Associated to the sets of edges E_1 and E_2 there are two transition matrices: G_1 and G_2. On the one hand, G_1 is an $N_2 \times N_1$ matrix whose elements $(G_1)_{ji}$ are the probabilities of jumping from node i in the first set to node j in the second set. On the other hand, G_2 is an $N_1 \times N_2$ matrix whose elements $(G_2)_{ji}$ are the probabilities of jumping from node i in the second set to node j in the first set.

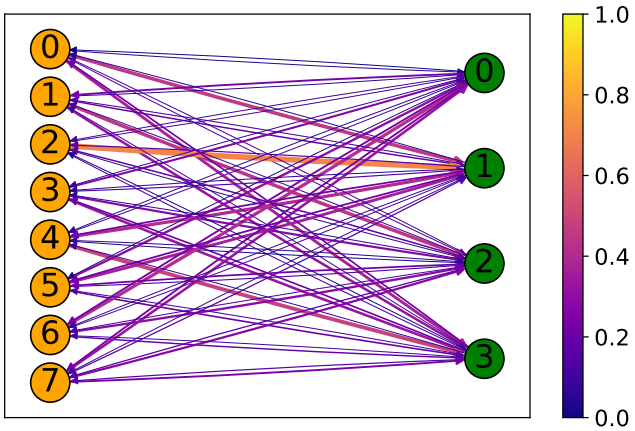


Figure 3.7: *Weighted bipartite graph with a first set of $N_1 = 8$ nodes and a second set of $N_2 = 4$ nodes. The weights of the directed edges are represented by the colormap, and are proportional to the width of the edges.*

The walk occurs on a bipartite graph, so that each computational basis state represents a walker with a location in the first set of nodes and also a location in the second set, also interpreted as an edge of the bipartite graph. Therefore, the states are formed by a tensor product of two registers, one for each set of nodes. The Hilbert space is

$$\mathcal{H}_S := \text{span}\{|i\rangle_1 |j\rangle_2 , \ i = 0, 1, \ldots, N_1 - 1, \ j = 0, 1, \ldots, N_2 - 1\}, \tag{3.24}$$

where the states with indexes 1 and 2 refer to the nodes in the two sets of the graph. Note that unlike in the coined quantum walk, where the total number of states depends on the underlying undirected graph, here the Hilbert space is defined with all the possible $N_1 \times N_2$ combinations even if the probabilities associated to the arcs are null. This is so for convenience, although we will see later that null edges can be removed under some circumstances.

We define the vectors

$$|\alpha_i\rangle := |i\rangle_1 \otimes \sum_{k=0}^{N_2-1} \sqrt{(G_1)_{ki}} \, |k\rangle_2 , \tag{3.25}$$

which are a superposition of the vectors representing the edges connecting to the i-th vertex of the first set, whose coefficients are given by the square root of the i-th column of the matrix G_1.

Analogously, we define the vectors

$$|\beta_i\rangle := \sum_{k=0}^{N_1-1} \sqrt{(G_2)_{ki}}\, |k\rangle_1 \otimes |i\rangle_2 \,, \tag{3.26}$$

which are a superposition of the vectors representing the edges connecting to the i-th vertex of the second set, and whose coefficients are given by the square root of the i-th column of the matrix G_2. We define two rotations, each of them around the subspace generated by each of these vector sets:

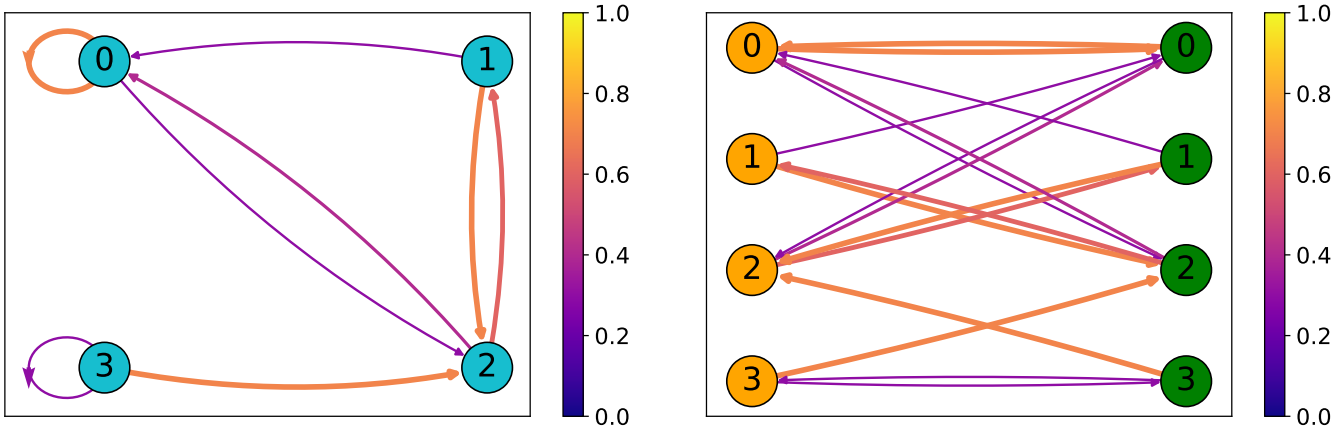
$$R_A := 2 \sum_{i=0}^{N_1-1} |\alpha_i\rangle \langle\alpha_i| - \mathbb{1}, \tag{3.27}$$

$$R_B := 2 \sum_{i=0}^{N_2-1} |\beta_i\rangle \langle\beta_i| - \mathbb{1}. \tag{3.28}$$

The quantum walk evolution operator W_s of the Szegedy quantum walk is defined as the product of the two reflections:

$$W_s := R_B R_A. \tag{3.29}$$

So far, the quantum walk has been defined on a general bipartite graph. In order to quantize a classical Markov chain with this quantum walk, we need a bipartite graph representing the associated transition matrix G. This is done with a symmetric bipartite graph obtained by duplicating the original graph of the Markov chain, as shown in Figure 3.8. The transition matrices of the resulting bipartite graph are therefore equal, so that $G_1 = G_2 = G$. Therefore, the quantum walk occurs on a bipartite graph, but represents the structure of the original graph.



(a) (b)

Figure 3.8: *(a) Weighted graph with $N = 4$ nodes whose transition matrix is given in (3.4). (b) Weighted symmetric bipartite graph obtained from a duplication process of the graph in (a). The weights of the directed edges are represented by the colormap, and are proportional to the width of the edges. Missing edges have a null transition probability.*

3.4.2. Alternative (coined) formulation

Whereas the Szegedy quantum walk was originally interpreted as a walk occurring on the edges of a bipartite graph, its motivation was actually quantizing a single Markov chain. Therefore, an alternative formulation based on the single graph is possible, avoiding the duplication process [90]. In this case, we interpret that the walk occurs on the nodes of the original graph, represented by the first register, and the second register is an inner degree of freedom as in the coined quantum walk.

For a graph with N nodes, the Hilbert space is now defined in a symmetric form as the span of all the vectors representing the $N \times N = N^2$ directed edges of the graph, i.e.,

$$\mathcal{H}_S := \text{span}\{|i\rangle_1 |j\rangle_2 , \ i, j = 0, 1, \ldots, N-1\}, \tag{3.30}$$

where the indexes 1 and 2 are inherited from the previous formulation. We define the vectors

$$|\psi_i\rangle := |i\rangle_1 \otimes \sum_{k=0}^{N-1} \sqrt{G_{ki}} \, |k\rangle_2 , \tag{3.31}$$

which are a superposition of the vectors representing the edges outgoing from the i-th vertex, whose coefficients are given by the square root of the i-th column of the matrix G. From these vectors we define the following projector operator:

$$\Pi := \sum_{i=0}^{N-1} |\psi_i\rangle \langle\psi_i| , \tag{3.32}$$

and use it to define a reflection operator around the subspace generated by the $|\psi_i\rangle$ states:

$$R := 2\Pi - \mathbb{1}. \tag{3.33}$$

The quantum walk evolution operator U_s is defined as:

$$U_s := S_w R, \tag{3.34}$$

where S_w is a swap operator (2.49) between the two registers. This form resembles the operator of a coined quantum walk, being the swap operator the equivalent of the flip-flop shift operator since in this case the system is expressed as the product of two registers with the same dimension.

If we look at equations (3.31) and (3.25), it is straightforward that the states $|\psi_i\rangle$ are equal to the states $|\alpha_i\rangle$. Thus, we have $R_A = R$. Moreover, the states $|\beta_i\rangle$ in (3.26) can be obtained applying the swap operator to the states $|\alpha_i\rangle$. Since the swap operator is Hermitian, we have $R_B = S_w R_A S_w = S_w R S_w$. Therefore, the original unitary operator $W_s = R_B R_A = S_w R S_w R$ corresponds to two steps of the alternative version $U_s = S_w R$, i.e., $W_s = U_s^2$. For this reason, we will refer to W_s as the double-step Szegedy operator, and to U_s as the single-step Szegedy operator. Unless otherwise stated, we consider the single-step operator U_s when we refer to the Szegedy quantum walk.

Finally, the initial state is usually constructed by a superposition of the $|\psi_i\rangle$ states, and the probability distribution of the walker after each time step of the quantum walk is usually obtained measuring the first register. However, there are also algorithms where the information of interest is obtained measuring the second register instead, as for example the quantum PageRank [1, 4, 31, 32, 102], which we review in Section 3.10.

3.5. Relation Between Coined and Szegedy Quantum Walks

It is known that given a Szegedy quantum walk, it can always be cast into the coined model, being the reflection R the coin of the walk. However, only a restricted set of coins can be cast into Szegedy's model [130]. In this section we review the equivalence between both models in order to establish the conditions that must be satisfied.

A priori we cannot establish a relationship between coined quantum walks and Szegedy's model because both Hilbert spaces have different dimensions. Moreover, the coined space $\mathcal{H}_C$ cannot be expressed as a tensor product since each node has in general a different degree. As we mentioned in Section 3.3, we can augment the coined space into an N^2-dimensional space considering also the edges (i,j) that are not present in the undirected graph. We call $\mathcal{H}_C^A$ to this augmented space, and $\mathcal{H}_\mathcal{C}$ is a subspace where the walk takes place. The augmented space now is isomorphic to the Hilbert space of the Szegedy quantum walk $\mathcal{H}_\mathcal{S}$ in (3.30). Let $\mathcal{F}$ be the isomorphism between both Hilbert spaces:

$$\mathcal{F} : \mathcal{H}_C^A \to \mathcal{H}_S. \tag{3.35}$$

Using it on $\mathcal{H}_\mathcal{C}$ we can find a reduced subspace of $\mathcal{H}_S$ where the equivalent Szegedy quantum walk should take place. Let us denote it as $\mathcal{H}_S^R$, so that $\mathcal{F} : \mathcal{H}_C \to \mathcal{H}_S^R$ and

$$\mathcal{H}_S^R := \operatorname{span}\{|i\rangle_1 |j\rangle_2 : (A_u)_{ji} = 1\}. \tag{3.36}$$

Now we need to define an augmented N^2-dimensional coined walk operator compatible with the augmented space. We define it directly acting on $\mathcal{H}_S$ as:

$$U_c^A = S_f^A C^A. \tag{3.37}$$

For this operator to be equivalent to the coined walk in $\mathcal{H}_\mathcal{C}$, it must have the same action as U_c in the reduced subspace $\mathcal{H}_S^R$ after the application of the isomorphism, and leaving it invariant. For the shift operator we can take $S_f^A = S_w$. From (3.9) it is trivial that the swap operator S_w acts equivalently to the flip-flop shift S_f in the reduced subspace. Moreover, $\mathcal{H}_S^R$ is trivially invariant under S_w due to the symmetry of the undirected adjacency matrix A_u.

With regard to the coin, we can define a coin operator in $\mathcal{H}_S$ as:

$$C^A := \sum_{i=0}^{N-1} |i\rangle_1\langle i| \otimes C_i^A, \tag{3.38}$$

where now C_i^A is a N-dimensional unitary operator that acts non-trivially in

$$\mathcal{H}_S^i := \operatorname{span}\{|k\rangle_2 \ : \ (A_u)_{ki} = 1\}, \tag{3.39}$$

conditioned by the first register being in the state $|i\rangle_1$. Therefore this augmented coin operator corresponds to a general conditional operator as shown in Section 2.3.1. In order to the coin C^A be equivalent to the coin C, the action in the reduced subspace must be provided by the isomorphism $\mathcal{F}$ as:

$$C^A |i\rangle_1 |j\rangle_2 := \begin{cases} \mathcal{F}(C\,|(i,j)\rangle) & \text{if } |i\rangle_1 |j\rangle_2 \in \mathcal{H}_S^R, \\ \\ -|i\rangle_1 |j\rangle_2 & \text{if } |i\rangle_1 |j\rangle_2 \in \left(\mathcal{H}_S^R\right)^\perp. \end{cases} \tag{3.40}$$

We need the reduced subspace to be invariant under the action of C^A. The action on the states that are perpendicular to $\mathcal{H}_S^R$ is irrelevant as long as it does not mix the subspaces. Thus, we have freedom to define the action on the orthogonal complement. For the sake of establishing an equivalence with the Szegedy quantum walk, we define this action as the $-\mathbb{1}$ operator.

Since we have defined $S_f^A = S_w$, the equivalence with the Szegedy quantum walk needs the operator R in (3.33) to be the coin operator C^A. For a state $|i\rangle_1 |j\rangle_2$ in the orthogonal complement of $\mathcal{H}_S^R$ we have $(A_u)_{ji} = 0$, and due to the symmetry, $(A_u)_{ij} = 0$, which implies that in the transition matrix $G_{ij} = G_{ji} = 0$. Thus, this state is perpendicular to all the $|\psi_i\rangle$ states in (3.31) and the action of R is just $-\mathbb{1}$. So, this subspace is invariant under R, and due to unitarity $\mathcal{H}_S^R$ also is. Therefore, we are closer to establish $C^A = R$, and $U_c^A = U_s$.

The last step is to find the expression of the individual coins C_i^A in (3.38). Let us rewrite the $|\psi_i\rangle$ states as:

$$|\psi_i\rangle = |i\rangle_1 \otimes |\omega_i\rangle_2 \,, \tag{3.41}$$

where

$$|\omega_i\rangle_2 := \sum_{k=0}^{N-1} \sqrt{G_{ki}}\, |k\rangle_2 \,. \tag{3.42}$$

Taking into account that $\mathbb{1}_{N^2} = \mathbb{1}_N \otimes \mathbb{1}_N$, the completeness relation of the identity for the first register, and substituting the expression for the $|\psi_i\rangle$ states in (3.33), we have

$$R = \sum_{i=0}^{N-1} |i\rangle_1\langle i| \otimes [2\,|\omega_i\rangle_2\langle\omega_i| - \mathbb{1}_N] \,. \tag{3.43}$$

Looking at the expression for the coin C^A in (3.38), we can identify the individual coins C_i^A with the reflections of the second register:

$$C_i^A = 2\,|\omega_i\rangle_2\langle\omega_i| - \mathbb{1}_N. \tag{3.44}$$

When the augmented coins can be expressed in this form, then the coined quantum walk is equivalent to a Szegedy quantum walk. In this sense, the coins codify the transition probabilities G_{ji}, and we have finally the equivalence between both quantum walks:

$$\mathcal{F}(U_c\,|(i,j)\rangle) = U_s\,|i\rangle_1\,|j\rangle_2 \,. \tag{3.45}$$

Note that the reduced subspace $\mathcal{H}_S^R$ is invariant under the Szegedy quantum walk, and it is actually where any Szegedy quantum walk takes place for a weighted graph, since it contains all the directed edges with non-null probability and their swapped versions, which appear due to the swap operator. Thus, the Szegedy quantum walk is indeed quantized also taking into account the underlying undirected graph. In the case that for an undirected edge one of the two arcs has a null transition probability, this is taken into account by the coin. Nevertheless, this ghost directed edge plays also a role in the quantum state and cannot be removed.

The question that remains is what are the conditions that must be satisfied so that a coin can be expressed as in (3.44). In the case that we are provided with a Szegedy quantum walk, we can always define the coins that way, so that all Szegedy quantum walk that come from the quantization of a classical Markov chain with transition matrix G can be cast into the coined model. In the case that the weighted graph is obtained normalizing the columns of the adjacency matrix of an undirected graph, the equivalent coin is the Grover coin [90, 131]. Thus, for arbitrary weighted graphs the

Szegedy quantum walk is a generalization of the Grover quantum walk [147], using as coin a general inversion about the weighted mean.

If, on the contrary, we are provided with a coined quantum walk, only a restricted set of coins satisfy equation (3.44). From it, we can formulate the following lemma about the conditions that must be satisfied by a coined walk to be cast into a Szegedy quantum walk.

Lemma 3.1. *Given a coined quantum walk U_c with a set of coin operators C_i of dimension d_i, there exists an equivalent Szegedy quantum walk U_s if and only if for each coin C_i there are $d_i - 1$ eigenvalues -1, and a single eigenvalue $+1$ whose eigenvector has real non-negative amplitudes.*

In this case the coin must be a reflection around the state $|\omega_i\rangle$, so that $|\omega_i\rangle$ is an eigenstate with eigenvalue $+1$, and the rest of eigenvalues are -1. Moreover, all the amplitudes of $|\omega_i\rangle$ are real positive or zero. This condition for the coins to be able to be cast into Szegedy's model is in concordance with the conditions found in the literature [130]. Note that, since eigenvectors that differ in a global phase are equivalent, the actual condition for the eigenvector is that there are no relative phases between the amplitudes.

This equivalence has been established between the coined walk and the single-step Szegedy operator U_s. If we can cast a coined walk into Szegedy in this case, then trivially it can also be cast into Szegedy's model considering the original double-step operator W_s, so that one step of the Szegedy quantum walk would be equivalent to two steps of the coined walk, being the equivalent operator U_c^2. Moreover, there can be cases where a coin cannot be cast into a Szegedy walk considering the single-step operator U_s, but it can be cast if we consider the double-step operator W_s instead. We will show an example for the $-\mathbb{1}$ coin in Section 4.4.1.

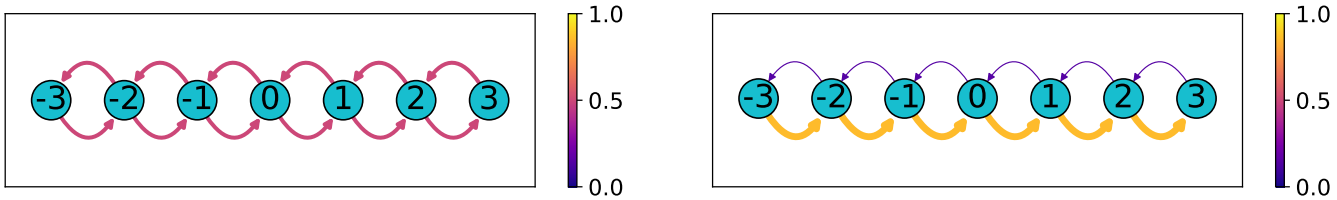
3.5.1. Example on the line

So far, we have obtained the conditions for establishing an equivalence between Szegedy's model and coined quantum walks. In this section we show how each model can be cast into the other using examples on the 1D line. The undirected graph that shows the backbone of the line is shown in Figure 3.3. The set of nodes ranges from $-\infty$ to ∞, although we only show seven nodes for the sake of simplicity.

Let us start with the Szegedy quantum walk on the undirected line. The naive transition matrix used for Szegedy's quantization is obtained by normalizing the adjacency matrix, so that a walker in node i has a probability of $1/d_i$ for jumping to each of the neighbor nodes. In this case the walker has a probability of one half for jumping either to the right or to the left. The associated weighted graph is shown in Figure 3.9(a), and the transition matrix is

$$(G_u)_{ji} = \frac{1}{2}\delta_{j-1,i} + \frac{1}{2}\delta_{j+1,i}, \tag{3.46}$$

where the subindex u makes reference to the fact that it is obtained from the undirected graph. Since the degree of each node is $d_i = 2$, all the coins C_i are going to be 2-dimensional matrices. Moreover, all the coins will be the same. By convention, the 2D basis is ordered so that the first element corresponds to the directed edge pointing to the right, and the second one pointing to the left. The coins correspond to reflections around the state $|\omega_i\rangle = (1,1)^T/\sqrt{2}$, which is obtained taking



(a) (b)

Figure 3.9: *(a) Weighted graph obtained by normalizing the adjacency matrix of the undirected graph for the infinite 1D line in Figure 3.3, so that all directed edges have the same transition probability. (b) Weighted graph associated to the coined walk with the Hadamard coin on the infinite 1D line, where the probability to the right is greater than to the left. The weights of the directed edges are represented by the colormap, and are proportional to the width of the edges. For the sake of simplicity only seven nodes have been represented.*

the square root of the columns of G_u and expressing it directly in the 2D basis of the coin. The rest of elements in the augmented space are always null. This is an eigenvector with eigenvalue $+1$, and the other eigenvalue must be -1. Thus, it trivially corresponds to the Pauli X operator, which is the Grover coin in 2D.

Now let us take the Hadamard coined quantum walk [124]. The spectrum of H is $\sigma(H) = \{+1, -1\}$, and the eigenvector for the eigenvalue $+1$ is $(h_R, h_L)^T$, where

$$h_R = \frac{1}{\sqrt{4 - 2\sqrt{2}}}, \tag{3.47}$$

$$h_L = \frac{\sqrt{2} - 1}{\sqrt{4 - 2\sqrt{2}}}. \tag{3.48}$$

Thus, by Lemma 3.1 we can cast it into Szegedy's model. The transition probabilities are obtained by the squared modulus of the amplitudes h_R and h_L, so that the walker has a probability of approximately 0.85 of jumping to the right and approximately 0.15 of jumping to the left. The transition matrix is therefore

$$(G_H)_{ji} = h_R^2 \delta_{j-1,i} + h_L^2 \delta_{j+1,i}. \tag{3.49}$$

Despite the fact that this coin puts in an equal superposition the computational basis of the coin register in 2D, so it was thought to be a sensible quantization of the classical walk on the undirected graph with G_u, when casting it into a Szegedy quantum walk we obtain a biased transition matrix which does not corresponds to the classical walk on the undirected line [130]. The associated weighted graph is shown in Figure 3.9(b), where we observe that the walk is biased to the right. This explains why the results of the simulation in Figure 3.4(c) show a distribution biased to the right when the initial coin state is in superposition, which is indeed the usual initial state of the Szegedy quantum walk.

Note that despite the fact that the Hadamard coined walk and the unbiased Szegedy walk are two quantizations based on the undirected graph, they are not equivalent. The equivalence between both the coined and Szegedy's models just means that given a transition matrix we can find a set of coins that reproduce that particular Szegedy walk. However, different coins produce different quantizations that may be equivalent to quite different weighted graphs, despite being devised from the same classical walk as in the case of the Hadamard coin.

3.6. Quantum Circuits for Szegedy Quantum Walks

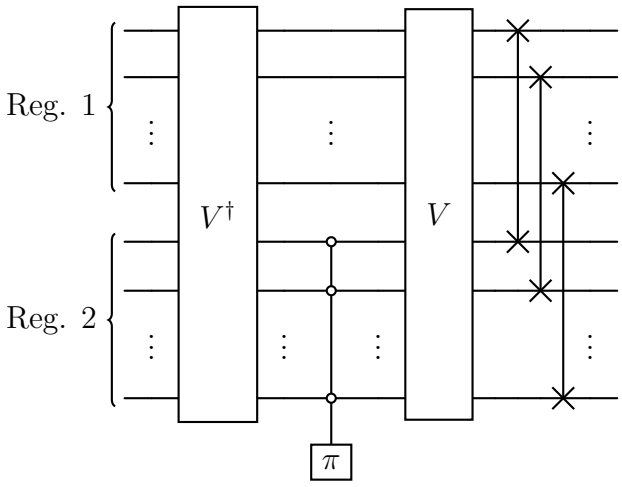


Figure 3.10: *Quantum circuit decomposition of the single-step Szegedy unitary evolution operator $U_s = S_w R$. Each register has n qubits for a graph with $N = 2^n$ nodes. The reflection $R = VDV^\dagger$, where D is a diagonal operator that implements a reflection around the state $|0\rangle_2$ in the second register.*

Constructing a quantum circuit for the Szegedy quantum walk is not a trivial task since it depends on the particular weighted graph where the walk takes place. However, there is a general structure for decomposing the circuit [91], which we review in this section.

Let us consider for simplicity that the number of nodes of the graph N can be expressed as a power of two, such that $N = 2^n$ for some integer n. Then, the quantum circuit needs two quantum registers of n qubits each. For $N \neq 2^n$ we could augment the graph in a manner such that the original walk occurs in an invariant subspace of the augmented graph.

The evolution operator in (3.34) is $U_s = S_w R$. The swap S_w is trivially a bunch of swap gates between the qubits of the first register and the qubits of the second register, as shown in Section 2.3.2. However, the reflection R is more complicated. In principle, it could be decomposed as the product of N reflections as shown in Section 2.3.4. However, this reflection operator is usually diagonalized in a way that it only requires a single reflection [91, 94]. With this purpose, we define the update operator V, which creates the $|\psi_i\rangle$ state if the first register is in the state $|i\rangle_1$, provided that the second register is in $|0\rangle_2$. Thus, its action is defined as:

$$V |i\rangle_1 |0\rangle_2 := |i\rangle_1 |\omega_i\rangle_2 = |\psi_i\rangle , \tag{3.50}$$

and the action on the rest of the computational basis is irrelevant as long as it is unitary. The update operator V diagonalizes the reflection operator R as $D = V^\dagger R V$, so that using (3.33) and (3.32) we obtain:

$$D = 2 \sum_{i=0}^{N-1} V_i^\dagger |\psi_i\rangle \langle\psi_i| V_i - \mathbb{1} = 2 \sum_{i=0}^{N-1} |i\rangle_1\langle i| \otimes |0\rangle_2\langle 0| - \mathbb{1}. \tag{3.51}$$

Factorizing the identity in both registers as $\mathbb{1}_{N^2} = \mathbb{1}_N \otimes \mathbb{1}_N$, and using the completeness relation of the identity in the first register, we obtain:

$$D = \mathbb{1}_N \otimes (|0\rangle_2\langle 0| - \mathbb{1}_N) . \tag{3.52}$$

This operator corresponds to a reflection around the state $|0\rangle_2$ in the second register. As shown in Section 2.3.4, it can be implemented with a controlled-$P(\pi)$ gate. Doing the inverse transformation we have $R = VDV^\dagger$. Thus, the circuit for the Szegedy quantum walk operator U_s can be decomposed in a general manner as shown in Figure 3.10. For the double operator W_s we would just apply twice this circuit.

The final problem is to provide a circuit for the update operator V. In general, we need to codify the N^2 transition probabilities of the transition matrix G, so that the complexity of the circuit would scale at least as $\mathcal{O}(N^2)$ for a general dense transition matrix. For sparse graphs, where there are few transition probabilities, this operator could be implemented more efficiently [94]. Moreover, even if the matrix is dense but the graph has some symmetry properties, this operator could also be implemented efficiently. Different implementations of the update operators with a polylogarithmic cost for graphs with symmetry can be found in the literature [91].

As an example, we take a cycle graph, as the one shown in Figure 3.6(a), and review the construction of the circuit of the update operator V shown in Figure 3.11.

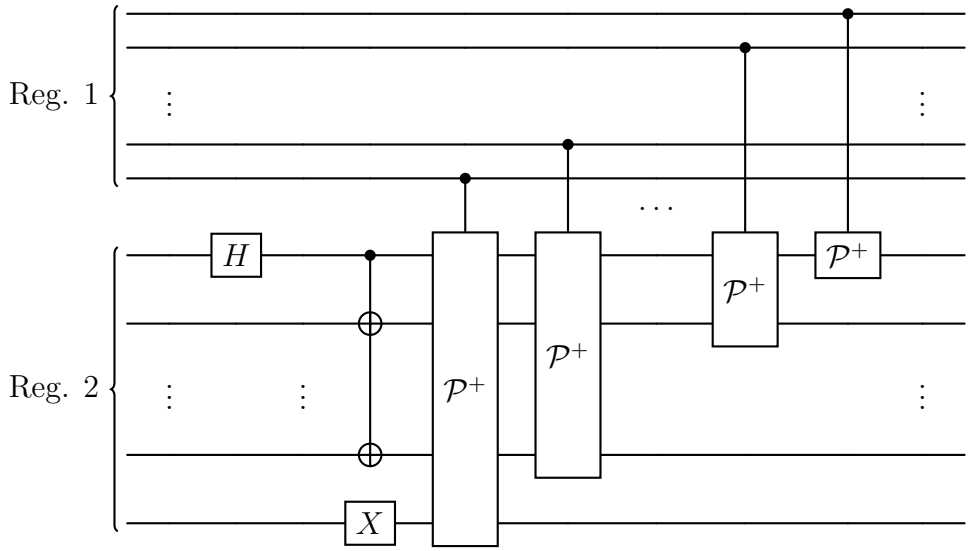


Figure 3.11: *Quantum circuit for the update operator V of the Szegedy walk on the undirected 1D cycle.*

Since we are considering an undirected cycle, the transition matrix G is the one given in (3.46). The update operator must act as $V|i\rangle_1|0\rangle_2 = |i\rangle_1|\omega_i\rangle_2$, where

$$|\omega_i\rangle_2 = \frac{1}{\sqrt{2}}|i+1\rangle_2 + \frac{1}{\sqrt{2}}|i-1\rangle_2 . \tag{3.53}$$

Since we are considering a finite graph, the additions and subtractions are performed modulo N.

The second register starts in the state $|0\rangle_2$. After the application of the H gate to the first qubit of the second register, it evolves to the state $(|0\rangle_2 + |1\rangle_2)/\sqrt{2}$. The following action of the CNOT gates and the last X gate let the second register in

$$\frac{1}{\sqrt{2}}|1\rangle_2 + \frac{1}{\sqrt{2}}|N-1\rangle_2 . \tag{3.54}$$

The permutation operators $\mathcal{P}^+$ were shown in Section 2.3.3. They can be implemented efficiently with multi-controlled-NOT gates. The join action of the $\mathcal{P}^+$ operators controlled by the first register this way corresponds to another permutation that transforms the computational basis states $|i\rangle_1 |x\rangle_2$ into $|i\rangle_1 |x + i \bmod N\rangle_2$ [91]. Therefore, when they are applied on the state in (3.54), the second register ends up in $|\omega_i\rangle_2$.

The Szegedy quantum walk on the undirected cycle corresponds to a coined quantum walk using a Grover coin for all the nodes. However, note that from the point of view of a Szegedy walk we cannot factorize the coin as a simple operator acting on the second register, as we did for the circuit based on the coined quantum walk in Figure 3.6(b). This is so because each Grover coin is different depending on which nodes are the neighbor of the particular node it is associated to. For example, for node 1 the Grover coin acts on the subspace formed by nodes 0 and 2, whereas for node 5 it acts on the subspace of nodes 4 and 6.

It is obvious that the quantum circuit for the Szegedy quantum walk is more inefficient that the one of the equivalent coined quantum walk, requiring more qubits and operations. The question that arises then is what the purpose of constructing such a circuit is. In this case we can obtain a simpler circuit from the perspective of a coined quantum walk since the cycle graph is a quite simple regular lattice, needing only a qubit for representing the coin state. However, for more general graphs, even from the point of view of a coined quantum walk, we would need a coin register of dimension N with n qubits as in the Szegedy quantum walk. The cycle graph is just a toy model that allows gaining intuition about how to construct circuits for Szegedy quantum walks, and its circuit can serve as a foundation for the construction of circuits for more complex graphs [91]. Moreover, the circuit of the cycle graph serves to test simulations of Szegedy quantum walks on a relatively simple graph, as for example has been done in the context of quantum homomorphic encryption [7].

3.7. Computational Complexity and Classical Simulation

In the context of the Grover search algorithm, in Section 2.4 we saw that the computational complexity depends on two factors. A query complexity, which is intrinsic to the algorithm and depends on the number of calls to the oracle, and an explicit factor that depends on the implementation of the algorithm. In the case of quantum walks we can make an analogy and differentiate between two kind of complexities [91], which in this thesis we define as follows:

Definition 3.4 (Walk complexity)**.** *It is the number of steps of the walk, i.e., that the unitary evolution operator is applied, and depends on the particular quantum walk algorithm.*

Definition 3.5 (Implementation complexity)**.** *It is the cost of implementing each step of the quantum walk, and depends on the computing model.*

It is frequent in the literature to find comparisons between classical and quantum walk algorithms based only on the walk complexity, showing generally a quadratic speedup [40,44,45,101]. This is due to the fact that implementing a quantum walk on a quantum computer is an open problem which depends on the particular graph, so that it is not always possible to provide an implementation complexity.

With regard to the efficiency of the implementation, along this introductory chapters we have talked about it in relative terms, explaining when a quantum circuit is more efficient than another one. One could think that a polynomial scaling might be enough to define something as efficient, since it would belong to the classical complexity class P or its quantum equivalent. Nevertheless, in practice a high exponent can make the algorithm very prohibitive [148]. Indeed, the adjective "efficient" depends on the context, and therefore is defined in a relative manner in due cases in this thesis. With regard to the Szegedy quantum walk, we have seen in Section 3.6 that the cost of a quantum circuit would scale as $\mathcal{O}(N^2)$ for a general transition matrix, although more efficient implementations are possible if the graph is sparse or has symmetry properties. The actual problem depends graph to graph, so we cannot give a particular implementation complexity until we deal with a particular problem. Since quantum walk algorithms usually provide a quadratic speedup with regard to classical ones in terms of the walk complexity, a polylogarithmic cost of the circuit would be desirable in order to maintain the speedup [91], as happens in the Grover algorithm. However, there are other algorithms where the aim is not an speedup in terms of steps, so that maybe a polynomial scaling is regarded as efficient depending on the exponent. For example, for the quantum PageRank algorithm we will see that an implementation complexity scaling as $\mathcal{O}(N^2)$ is acceptable as efficient.

Although the main idea is to implement quantum walks on quantum computers, some algorithms are also though to be run on classical computers, so that the implementation complexity depends on the classical simulation algorithm. An example is the quantum PageRank that we study in Section 3.10. For a classical simulation, we need to construct the matrix of the unitary evolution operator, and just apply it to the vector representing the state of the system, in a similar way to how we would simulate deterministically a classical walk.

For simplicity, let us consider that the $N \times N$ transition matrix G is dense, so that it has $\mathcal{O}(N^2)$ non-null elements. The matrix-vector multiplication requires N^2 operations, so that it scales as $\mathcal{O}(N^2)$ for the classical walk. In the case of the quantum walks in discrete time the inner coin degree of freedom makes that, in general, the unitary evolution operator is an $N^2 \times N^2$ matrix. Thus, the complexity of the classical simulation, both in time and memory terms, would scale as $\mathcal{O}(N^4)$. A more efficient implementation can be done using a sparse representation of the matrix, since actually only N^3 elements are non-null given the block-diagonal structure of the coin operator. Although the exponent is not so much big, a cubic scaling has resulted quite inefficient for the simulations performed in this thesis, allowing graphs up to 256 nodes. For that reason, in order to perform more complex simulations, we needed to develop a more efficient simulation algorithm scaling as $\mathcal{O}(N^2)$, which we present in Chapter 9. This scaling is optimal for dense transition matrices, since it is the minimum of information required to represent them, and therefore we denote it as efficient in the context of classical simulation.

Note that, in contrast to the quantum circuit simulation, where the complexity was measured in terms of the number of qubits $n = \log_2 N$, for quantum walk algorithms we measure it directly in terms of the number of nodes N. For this reason, we find their classical simulation scaling polynomially rather than exponentially, despite being quantum algorithms. Of course, in the case that we implement them on quantum computers based on qubits, we need exponentially less spatial resources than on a classical computer, although this has nothing to do with the running time complexity.

3.8. Other Quantum Walks

In this chapter we have mostly dealt with quantum walks in discrete time using an inner degree of freedom. It is important to mention that there exists other types of quantum walks lacking of the extra register, so that the Hilbert space is simply formed by the computational basis corresponding to the N nodes of the graph: $\mathcal{H} = \text{span}\{|i\rangle\,,\ i = 0, 1, \ldots, N-1\}$. Examples are the staggered quantum walk [149], which also occurs in discrete time, or the continuous-time quantum walk [122, 150]. The latter is important for comparing our discrete-time semiclassical walks [2] in Chapter 6.

A continuous-time quantum walk is obtained by the quantization of a classical Markov chain in continuous time, which is described by a differential equation. In the quantum case it is substituted by the Schrödinger equation, obtaining the unitary evolution operator by the exponentiation of a Hamiltonian H related to the graph structure:

$$U(t) = e^{-\mathrm{i}Ht}, \tag{3.55}$$

where now t is a continuous parameter. With regard to the Hamiltonian, it must be Hermitian, and since the parameters of the network are real, it is usually a real symmetric matrix. There are different definitions of it [40], but to keep it as simple as possible we consider a matrix proportional to the adjacency matrix A of the graph, which necessarily needs to be undirected.

3.9. Grover Search Algorithm as a Quantum Walk

In Section 2.4 we studied the Grover algorithm [26, 27, 95], that allowed finding marked elements with an oracle. A similar strategy can be used with quantum walks for finding marked nodes in a graph. Indeed, in this section we show that if we perform a coined quantum walk in the complete graph with loops using the Grover coin, and we add an oracle, we obtain the Grover algorithm [146]. Since we have established the equivalence between the coined model and the Szegedy quantum walk, we can use the latter for the proof.

A complete graph with loops is a graph where each node connects to each other and itself. Therefore, the probability of jumping elsewhere, including the current node, is $G_{ji} = 1/N$. From (3.31) it is straightforward that all the $|\psi_i\rangle$ states have in the second register an equal superposition of the computational basis, so that $|\psi_i\rangle = |i\rangle_1 |s\rangle_2$, with $|s\rangle$ given in (2.66). The equal superposition is created applying a Hadamard gate to all the qubits. Since this state is the same for all the $|\psi_i\rangle$ states, the update operator V is not controlled by the first register, and can be factorized as $V = \mathbb{1}_N \otimes H^{\otimes n}$. Moreover, the equivalent coin, the Grover coin, is exactly the same for all the nodes in this case, so that the reflection can also be factorized as $R = \mathbb{1}_N \otimes G_D$.

In order to construct the quantum walk search algorithm, the coin is modified introducing a phase-flip oracle acting on the first register. We define the operator $Q_1 := Q_f \otimes \mathbb{1}_N$, which applies this oracle. Therefore, the walk unitary evolution operator is

$$U_s = S_w R Q_1. \tag{3.56}$$

The initial state is taken as an equal superposition of all the $|\psi_i\rangle$ states, which is created applying a Hadamard gate to all the qubits of the first register, followed by the update operator V. In this

case, this corresponds to applying a Hadamard gate to all the qubits of the quantum circuit. Thus, the quantum circuit is initialized with the state $|s\rangle$ in both registers, which is the initial state of the Grover algorithm. Taking into account how the operators have been defined, in Figure 3.12 we show the circuit for two walk steps. Due to the swap operators, after the application of two steps of the walk we have an oracle followed by the diffusion operator in the first register, and the other way around in the second register. Therefore, two steps of the walk, $W_s = U_s^2$, correspond to a single step of the Grover algorithm, $G_K = G_D Q_f$, in the first register. Since it was initialized in the state $|s\rangle$, we effectively recover the Grover algorithm in the first register.

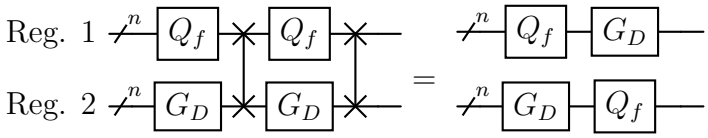


Figure 3.12: *Quantum circuit for two walk steps U_s of the Szegedy quantum walk with oracle on the undirected complete graph with loops. It is equivalent to the application of a Grover kernel $G_K = G_D Q_f$ in the first register, and a reversed kernel in the second register.*

With regard to the second register, we can observe that it is also occurring a Grover algorithm in parallel. In this case, two steps of the walk correspond to a reversed Grover kernel. However, if we concatenate steps, we can factorize some kernels, and after $2t$ walk steps we have applied the sequence of operators $Q_f G_K^{t-1} G_D$. Since the state $|s\rangle$ is an eigenstate of the diffusion operator, the first G_D plays no role. Therefore, we are performing $t-1$ Grover steps followed by a final oracle. Nevertheless, since we are measuring in the computational basis, the probabilities are not affected by the last phase-flip of the marked nodes. Therefore, it is as if we only applied the kernels, and we have in an effective manner another Grover algorithm in the second register, although delayed one time step with regard to the one of the first register.

It may seems quite inefficient to implement a Grover search as a quantum walk, since we are wasting resources in the second register. However, this serves as a basis for other quantum walk search algorithms on different graphs rather than the complete graph with loops, so that a non-trivial structure plays also a role in the evolution and can provide extra information. An example is the quantum SearchRank algorithm [4, 102] which we deal with in Chapter 7, which indeed leverages the process in the second register.

It is interesting to note that, from the point of view of a coined quantum walk, the addition of the oracle produces another coined quantum walk. Both the oracle Q_1 and the coin, which in this case is the reflection R, are block-diagonal operators. The coin has N blocks with the diffusion G_D, and the oracle Q_1 has the identity $\mathbb{1}_N$ in the blocks corresponding to unmarked nodes, and $-\mathbb{1}_N$ for the marked nodes. Therefore, the product RQ_1 provides also a block-diagonal operator with G_D for unmarked nodes and $-G_D$ for marked ones, which can be understood as a new coin operator. However, from the point of the Szegedy quantum walk, this algorithm does not corresponds to an instance of this model. The new coin RQ_1 is no more a reflection around a subspace of N states, and therefore there is no transition matrix G such that a Szegedy quantum walk is equivalent to this search algorithm. Therefore, in this case, it is just a Szegedy quantum walk plus an oracle, although by a slight abuse of notation we shall also call it simply a Szegedy quantum walk.

Apart from oracle-based quantum walk search algorithms, as different formulations of Szegedy quantum walks with oracles [101], there exist other proposals lacking of the need of oracles. For

example, a coined quantum walk with a different coin for marked and unmarked nodes [44], or Szegedy quantum walks on modified graphs with sinks [40, 45], which we review in Chapter 4 and indeed show that they are equivalent.

3.10. The PageRank Algorithm and Its Quantization

Before ending this chapter and diving into the main results of this thesis, we introduce the PageRank algorithm [141–144], a random walk algorithm that produced a revolution of search engines to surf the internet. Contrary to its competitors, whose ranking of pages was quite subjective, the PageRank algorithm classifies in an objective manner, taking into account the structure of links between the pages. Beyond its importance in the World Wide Web, it has a lot of applications in fields where networks play a central role. For example, in bibliometrics [151, 152], finances [153], metabolic networks [154], drug discovery [155], protein interaction networks [156], and social networks [157].

In the early era of quantum computing there has been great interest in the development of large-scale quantum networks with the perspective of a future quantum internet [158–160]. As happens with classical information, the quantum information of the quantum internet will need to be classified. In this sense, the classical PageRank could be used on a classical computer to classify the quantum information. However, it is sensible to think that a quantized version of the PageRank algorithm will classify the quantum information better, taking into account the effects of superposition and interference of quantum mechanics. With that purpose, in 2011 a quantization of the PageRank algorithm was proposed based on the Szegedy quantum walk [31, 32]. On the one hand, the results obtained with the quantum algorithm are expected to be more sensible for quantum networks. However, until larger quantum networks become available, this is still an open problem. On the other hand, if we consider that the information is classical, then the quantum PageRank shows features that can even enhance the classical algorithm in this context. Thus, the quantum PageRank is interesting not only for the future quantum networks, but also for the current classical ones.

Given the properties of this quantum algorithm, it represents a useful example of application of interest of the Szegedy quantum walk, and we mostly use it in this thesis as a baseline to test our new quantum walk algorithms.

3.10.1. PageRank formalism and example on a small graph

Given a set of pages P_i being nodes of a network, we define I_c as the vector whose entries are the classical importance or PageRank scores of every page. The naive definition of the PageRank distribution is the following:

$$I_c(P_i) := \sum_{P_j \in B_i} \frac{I_c(P_j)}{\text{outdeg}(P_j)}, \tag{3.57}$$

where B_i is the set of nodes linking to the node P_i and $\text{outdeg}(P_j)$ is the outdegree of the node P_j. This formula means that the importance of a node depends on the nodes that link to it. The more important a linking node is, the greater its contribution to the PageRank is. However, its

contribution is equally distributed between all the nodes it links to.

In order to compute the PageRank distribution we define the connectivity matrix H of the directed network of pages P_i:

$$H_{i,j} := \begin{cases} 1/\text{outdeg}(P_j) & \text{if } P_j \in B_i, \\ 0 & \text{otherwise,} \end{cases} \tag{3.58}$$

which is an $N \times N$ matrix for a network with N nodes. With this matrix, we can express equation (3.57) as $I_c = HI_c$, so that the PageRank vector is the eigenvector with eigenvalue 1 of the matrix H. Computing this vector by diagonalization is an unfeasible task for a network with millions of nodes as the World Wide Web. For that reason, a solution based on the power method was proposed. It consists of repeatedly applying the matrix H to an initial probability distribution vector until it converges to this eigenvector. Therefore, if H were an stochastic matrix, it would correspond to a random walk as shown in equation (3.5). However, given the form H is constructed, it can have null columns for pages no connecting any other node, and thus it is not stochastic. Therefore, for the algorithm to work, this matrix must be patched, taking care that the new eigenvector represents properly the original ranking of importance. In this case, all the columns where all the elements are zero are substituted with columns with all entries equal to $1/N$. This results in a column-stochastic matrix E, where all the columns add up to one, and could be used for the random walk. Although this ensures that E has an eigenvector with eigenvalue 1, the random walk may not converge. This occurs when there are other complex eigenvalues with a modulus equal to 1.

The Perron-Frobenius theorem asserts that for a primitive matrix, which corresponds to an irreducible and aperiodic matrix for a finite Markov chain, there exists only an eigenvector with modulus equal to 1 [137, 161, 162]. For an stochastic matrix this eigenvalue must be 1, and it suffices that the Markov chain is ergodic for the transition matrix to be primitive. In order to have such a matrix, we need to perform a second patch. The matrix E is mixed with another matrix $\mathbf{1}$ where all the entries are equal to 1, obtaining a primitive matrix called the Google matrix G:

$$G := \alpha E + \frac{(1-\alpha)}{N}\mathbf{1}. \tag{3.59}$$

The parameter α is called the damping parameter corresponding to the previous mixing, and its value lies in $[0, 1]$. It was found by Brin and Page that the optimal value is $\alpha = 0.85$. In this thesis, this value of the damping parameter is considered unless otherwise stated. This mixing can be interpreted as that the random walk is performed on the network of interest driven by E with probability α, and with probability $1 - \alpha$ the walker makes a random hopping driven by the matrix $\mathbf{1}$.

We perform the random walk with the patched matrix G, so we redefine the vector of PageRank scores satisfying $I_c = GI_c$. Thus, it is the eigenvector with eigenvalue 1 of the Google matrix G. Thanks to the mixing with the random hopping matrix, a random walk performed with the matrix G, starting with any probability distribution, always converges to this eigenvector. Then, now we can use the power method to obtain the PageRank distribution. We only have to take an initial probability distribution and repeatedly apply the matrix G until it converges. In this thesis, we choose the equal probability distribution as the starting point.

Since the classical PageRank corresponds to a random walk, its quantization is based on the quantum walk of Szegedy shown in Section 3.4, using as transition matrix the Google matrix G. Since the swap operator in the unitary evolution operator U_s in (3.34) changes the directedness of

the graph, which for the PageRank algorithm is crucial, the operator U_s must be applied an even number of times. Therefore, the actual time evolution operator is chosen as the double-step operator $W_s = U_s^2$.

The initial state of the system is chosen as an equal superposition of all the $|\psi_i\rangle$ states in (3.31):

$$\left|\Psi^{(0)}\right\rangle := \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\psi_i\rangle \,, \tag{3.60}$$

and the final state is obtained after the application of the operator W_s a number of time steps t. In the quantum PageRank the information is obtained by measuring the second register, since it represents where the links are pointing to. Therefore, the projection onto the computational basis of the second register gives us the quantum PageRank score for each node:

$$I_q(P_i, t) := \left|\left|\, {}_2\langle i| \, W_s^t \left|\Psi^{(0)}\right\rangle \right|\right|^2 . \tag{3.61}$$

For each time step of the algorithm we can obtain a different probability distribution. This quantum PageRank distribution depending on time is called the instantaneous quantum PageRank [31], and it fluctuates in time instead of converging. An analogous of the stationary distribution of a classical walk can be defined in the quantum context averaging all the instantaneous probability distributions [128]. In this case, the time-averaged quantum PageRank distribution is defined as:

$$I_q(P_i) := \frac{1}{T} \sum_{t=0}^{T} I_q(P_i, t). \tag{3.62}$$

This quantity converges for a sufficiently large value of T [32, 163]. Therefore, it is an objective measure of the importance, and in the following, when we refer to the quantum PageRank, we mean the time-averaged quantum PageRank, unless we mention explicitly the instantaneous PageRank.

As an example of how these algorithms work, we have taken the results on a small generic graph with seven nodes from the original paper [31], which is shown in Figure 3.13(a). For the classical PageRank, we show in Figure 3.13(c) the probability of the seven nodes for each time step of the random walk, where we can observe that the probability ends up converging. However, for the quantum PageRank, the instantaneous distributions oscillate as shown in Figure 3.13(d). In this case, we only show the two most important nodes and the least important one. Despite its clear difference in the classical PageRank, there are time steps of the quantum walk such that node 3 acquires a greater importance than nodes 4 and 6. This justifies the introduction of the time-averaged quantum PageRank, whose distribution ends up converging as shown in Figure 3.13(e). Finally, the classical and quantum PageRank distributions are compared in Figure 3.13(b). In both cases, node 6 is identified as the most important one, followed by node 4, and node 3 as the least important one. However, there are some violations of the order for intermediate nodes, which can be due to quantum fluctuations.

So far, we have only shown the initial example in a small network, showing intriguing properties such as a violation in the nodes ranking [31]. Nevertheless, it was also scaled to complex networks, showing further properties as a better resolving of the structure [32], which we review in Chapter 5 when comparing with our new algorithm introducing complex-phase extensions [1]. Moreover, the quantum PageRank algorithm has also been coupled to quantum search as a further step towards a quantum search engine [102], as we review in Chapter 7 in the context of semiclassical walks.

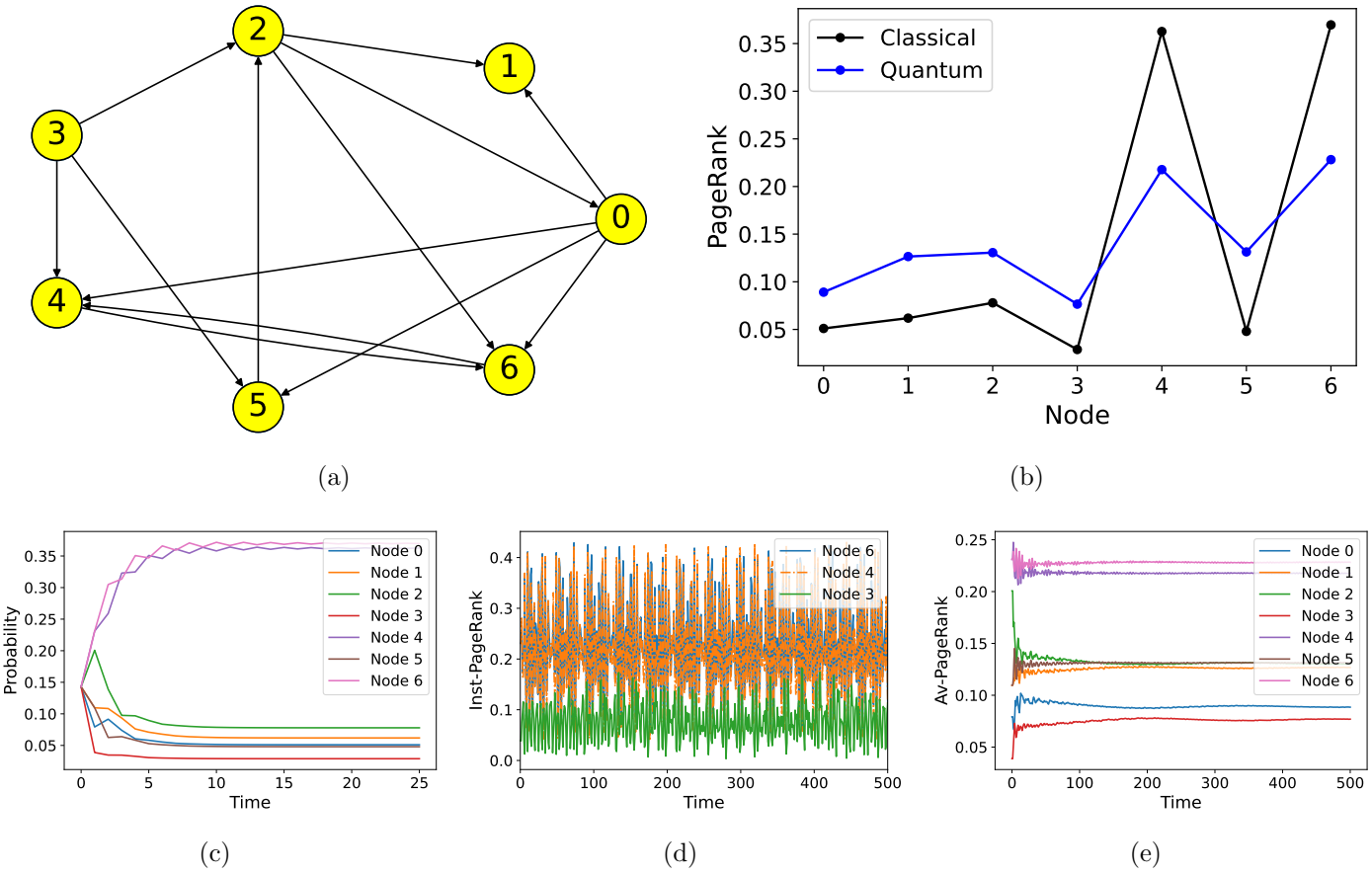


Figure 3.13: *(a) Small generic graph with $N = 7$ nodes. (b) Classical and quantum PageRank distributions for the small generic graph. (c) Probability of each node versus the time for the classical walk on the small generic graph, converging to the classical PageRank distribution. (d) Instantaneous quantum PageRank scores of nodes 6, 4 and 3 of the small generic graph versus the quantum walk time. (e) Time average of the instantaneous quantum PageRank scores for all the nodes of the small generic graph versus the quantum walk time, converging to the quantum PageRank distribution.*

3.10.2. Time and memory complexity

The classical PageRank algorithm consists of repeatedly applying the Google matrix G to an initial probability distribution vector until it converges to a stationary distribution. It turns out that the number of time steps grows very slowly with the size of the network, so that the walk complexity is $\mathcal{O}(\log N)$. Indeed, we can take it as $\mathcal{O}(1)$, since 50 time steps is enough for networks with millions of nodes [143]. Therefore, all the complexity depends on how the algorithm is implemented. As mentioned in Section 3.7, since the Google matrix G has no null elements, both the memory and time complexity of the matrix multiplication scales as $\mathcal{O}(N^2)$. However, if the network is sparse, so that each node connects to a few set of nodes, the implementation complexity can be reduced to $\mathcal{O}(N)$.

Suppose a sparse network, so that each node connect at most k nodes, with $k \ll N$ and approximately constant. In this case, the matrix H, and also the matrix E after the first patching, contains $\mathcal{O}(N)$ non-null elements. Since we can decompose G as in equation (3.59), we can also decompose

the application to a probability distribution vector as two matrix multiplications:

$$GI_c = \alpha E I_c + (1-\alpha)\frac{\mathbf{1}}{N} I_c. \tag{3.63}$$

On the one hand, the product of the matrix $\mathbf{1}/N$ applied to any probability distribution vector produces always the uniform distribution, so that there is no need to perform this calculus. On the other hand, for sparse networks the matrix E is also sparse, and therefore both the memory and time requirements of the product scale as $\mathcal{O}(N)$. After this product, the addition of the uniform vector, weighted by the damping parameter α, also scales as $\mathcal{O}(N)$, so that the implementation complexity of the classical PageRank ends up scaling as $\mathcal{O}(N)$ for sparse networks. Examples of sparse networks are the scale-free networks [164], which are good models of the World Wide Web and we use to test our algorithms in future chapters.

With regard to the quantum PageRank, it seems that the walk complexity is also constant, so that it does not grow with the size of the graph [165]. However, the number of time steps needed until the convergence of the averaged quantum PageRank is larger. In the simulations performed in this thesis we have taken 500 time steps to ensure the convergence, and we have checked that it does not vary with the size of the network. Thus, again the complexity of the algorithm relies ultimately on the implementation.

We discussed in Section 3.6 that, for a dense transition matrix as the Google matrix G, a quantum circuit would need at least $\mathcal{O}(N^2)$ gates to encode all the transition probabilities. If the network is sparse, it turns out that the mixing of the matrix E with the uniform matrix $\mathbf{1}$ does not break the symmetry properties of the graph, so that in principle, the quantum circuit for a quantum walk using as transition matrix E could be adapted for the Google matrix G [91]. Therefore, we could find also an implementation complexity scaling as $\mathcal{O}(N)$, or even more efficient if the graph posses special symmetry properties.

Another option is implementing the quantum PageRank as a quantum-inspired algorithm on a classical computer. As we mentioned in Section 3.7, we devised an algorithm that scales as $\mathcal{O}(N^2)$ for dense transition matrices as the Google matrix G, which we introduce in Chapter 9. Since the Google matrix enters in the quantum walk equations in a non-linear form, we cannot decompose the action between E and $\mathbf{1}$ as in the classical case, so that the classical simulation of the quantum PageRank scales as $\mathcal{O}(N^2)$ even for sparse networks. Nevertheless, this scaling is quite acceptable in efficiency terms, as it is optimal.

Although the number of walk steps is larger for the quantum PageRank, despite being constant, and the implementation complexity cannot be reduced for sparse networks in the classical simulation, it is important to remark that the aim of the quantum PageRank does not consist of providing a quantum speedup with respect to the classical algorithm. In this case the advantage comes from the quality of the results, which can be more sensible for quantum networks or resolve better the structure of classical ones [32]. Therefore, the increased running time is not actually an issue of the quantum algorithm we must worry so much about. Furthermore, we find an scaling as $\mathcal{O}(N^2)$ efficient enough.

Results I

Phase Extensions

Chapter 4

Graph-Phased Szegedy quantum walk

We begin the results parts of this thesis exploring new possibilities of the Szegedy quantum walk by means of complex-phase extensions. An extension of this walk was first done using a complex Hermitian adjacency matrix [166], and later using arbitrary complex weights in a model dubbed twisted Szegedy walk [167]. This model introduced complex phases in the evolution operator, which are associated to the edges of the graph, and were first used for studies of localization phenomena [167–169]. Moreover, they have been considered for problems of state transfer in quantum walks [170] and quantum search [171, 172].

In this thesis, we provide a further generalization of the phase extensions by means of arbitrary phase rotations (APR). This technique was introduced in the context of the Grover search algorithm [96, 107, 108], and has been used to improve its performance [110–112] and even make it deterministic [74, 109]. Combining them in the Szegedy quantum walk with the previous extensions, we obtain the graph-phased Szegedy quantum walk [5].

An important result we find for our graph-phased Szegedy model is that the set of coined quantum walks that can be cast into a Szegedy walk is increased considerably. We expand on the conditions that must satisfy the coin operators [130], shown in Section 3.5, providing new conditions considering also the phase extensions. We also show how the new extensions that we introduce in our work allow marking nodes in a graph using oracles. We use this extension to review the proof that the method of absorbing vertices usually used for marking [40, 45] is equivalent to the use of a different coin in the coined walk [44, 131]. Moreover, we study how given an efficient quantum circuit for the implementation of a Szegedy quantum walk [91, 94], we can modify it to include the complex-phase extensions.

This initial chapter of this first part of results presents the graph-phased walk in some general sense, and in Chapter 5 we explore further an application in the context of the quantum PageRank algorithm [1]. Moreover, this part connects with Chapter 9, where we show the efficient classical simulation algorithm of the Szegedy quantum walk [3] considering also the phase extensions.

With regard to this particular chapter, it is structured as follows. In Section 4.1 we review the current phase extensions of the Szegedy quantum walk, to later introduce the graph-phased model. In Section 4.2 we study the equivalence between the graph-phased Szegedy walk and the coined model, and show examples on the infinite line. In Section 4.3 we show the efficient construction of

quantum circuits with the phase extensions. In Section 4.4 we show how some phase extensions can be used for marking nodes in a graph. Finally, we summarize and conclude in Section 4.5.

4.1. Graph-Phased Szegedy Quantum Walk

In this section we review the current phase extensions given in the twisted Szegedy quantum walk [167], and introduce new extensions through arbitrary phase rotations. By combining them, we generalize the model and introduce the graph-phased Szegedy walk.

4.1.1. Twisted Szegedy walk

The Twisted Szegedy walk introduced complex phases by means of a kind of complex-valued weights [167,173]. In our notation, this corresponds to an extended Szegedy walk [149] where the $|\psi_i\rangle$ states in (3.31) are substituted by

$$|\psi_i(\varphi)\rangle := \sum_{k=0}^{N-1} e^{\mathrm{i}\varphi_{ik}} \sqrt{G_{ki}}\, |i\rangle_1 |k\rangle_2 \,, \tag{4.1}$$

where φ is an $N \times N$ matrix, and each element φ_{ij} is the phase associated to the edge state $|i\rangle_1 |j\rangle_2$. Thus, the reflection operator R in (3.33) is extended as another reflection $R(\varphi)$, which reflects around the subspace generated by the $|\psi_i(\varphi)\rangle$ states. When all the phases are 0, the standard Szegedy reflection is recovered. Note that due to the convention of the transition matrix G being column-stochastic, the weight probability associated to the edge state $|i\rangle_1 |j\rangle_2$ is G_{ji}. Thus, the real-valued weights in the transition matrix G are associated with the elements of φ^T, and vice versa.

The twisted Szegedy walk provided another extension with a complex phase for each edge, acting inside the swap operator [167]. In this case, the swap operator is substituted by

$$S_w(\Omega) := \sum_{i,j=0}^{N-1} e^{-\mathrm{i}\Omega_{ij}} |i\rangle_1\langle j| \otimes |j\rangle_2\langle i| \,, \tag{4.2}$$

which is denoted as the twisted swap. Moreover, the phases matrix Ω is asymmetric, so that $\Omega_{ij} = -\Omega_{ji}$. This preserves the Hermitian character of the swap, and therefore $S_w^2(\Omega) = \mathbb{1}$, as desired for a flip-flop shift operator. Note that the global action of this operator is just a swap S_w, followed by the application of a relative phase to each element of the quantum vector state in the computational basis.

4.1.2. Arbitrary phase rotations

In this thesis, we propose another modification of the Szegedy quantum walk changing the reflection R by an arbitrary phase rotation operator, as the one shown in Section 2.3.5. Whereas in the previous cases N^2 phases were introduced, in this case a single phase denoted as θ is used to define the phase rotation operator:

$$R(\theta) := (1 - e^{\mathrm{i}\theta})\Pi - \mathbb{1}. \tag{4.3}$$

The standard reflection is recovered for $\theta = \pi$.

The phase θ acts in a global manner in the graph. However, the same as there are N^2 phases φ_{ij} associated to the edges of the graph, we could think of N phases θ_i associated to each of the nodes of the graph. Let us expand the projector Π in (4.3) using (3.32), and introduce the constant factor $(1 - e^{\mathrm{i}\theta})$ in the sum:

$$R(\theta) = \sum_{i=0}^{N-1} (1 - e^{\mathrm{i}\theta}) \left|\psi_i\right\rangle \left\langle\psi_i\right| - \mathbb{1}. \tag{4.4}$$

Now, we can give θ a different value θ_i for each node i. Let us define $\vec{\theta}$ as a vector with the N different phases. The phase rotation operator becomes

$$R(\vec{\theta}) = \sum_{i=0}^{N-1} (1 - e^{\mathrm{i}\theta_i}) \left|\psi_i\right\rangle \left\langle\psi_i\right| - \mathbb{1}. \tag{4.5}$$

These phases act locally in the nodes of the graph, and the sum cannot be factorized with a projector operator.

In order to differentiate our extensions from the previous ones, we call link phases to the phases φ_{ij} associated to the edges of the graph through the reflection, shift phases to the phases Ω_{ij} associated to the edges through the swap, and APR phases to the phases θ_i related to the nodes of the graph. All these extensions are compatible and can be applied at the same time, giving rise to a generalized model.

4.1.3. Graph-phased Szegedy quantum walk

We define the graph-phased model joining all the phase extensions of the reflection operator R. Let us define the following operator:

$$\Sigma(\vec{\theta}, \varphi) := \frac{1}{2} \sum_{i=0}^{N-1} (1 - e^{\mathrm{i}\theta_i}) \left|\psi_i(\varphi)\right\rangle \left\langle\psi_i(\varphi)\right|. \tag{4.6}$$

This operator generalizes the projector Π in (3.32), which is recovered for $\theta_i = \pi$ and $\varphi_{ij} = 0$. It is a kind of pseudoprojector, since acting on an arbitrary state provides a state in the subspace spanned by the $|\psi_i(\varphi)\rangle$ states. However, it does not corresponds to the actual orthogonal projection.

The phase rotation operator associated to this walk is defined as:

$$R(\vec{\theta}, \varphi) := 2\Sigma(\vec{\theta}, \varphi) - \mathbb{1}, \tag{4.7}$$

thus resembling the original structure of the reflection R in (3.33), substituting the projector Π by the pseudoprojector $\Sigma(\vec{\theta}, \varphi)$. The unitary evolution operator is then defined substituting the reflection in the operator U_s in (3.34):

$$U_s(\vec{\theta}, \varphi) := S_w R(\vec{\theta}, \varphi). \tag{4.8}$$

Again, we can think of the double Szegedy operator W_s, using two times the operator $U_s(\vec{\theta}, \varphi)$. In general, the phases of both operators can be different, so that the most general double-step Szegedy operator is

$$W(\vec{\theta}_1, \varphi_1, \vec{\theta}_2, \varphi_2) := U(\vec{\theta}_2, \varphi_2) U(\vec{\theta}_1, \varphi_1). \tag{4.9}$$

Before going on, let us fix some definitions about the different models we deal with in this thesis:

Definition 4.1 (Standard Szegedy quantum walk)**.** *It is the Szegedy quantum walk without any phase extension.*

Definition 4.2 (Link-phased Szegedy quantum walk)**.** *It is the Szegedy quantum walk with link phases φ_{ij}, but without APR phases.*

Definition 4.3 (Szegedy quantum walk with global APR)**.** *It is the Szegedy quantum walk with the same APR phase θ for all the nodes, but without link phases.*

Definition 4.4 (Vertex-phased Szegedy quantum walk)**.** *It is the Szegedy quantum walk with multiple local APR phases θ_i, but without link phases.*

Definition 4.5 (Graph-phased Szegedy quantum walk)**.** *It is the Szegedy quantum walk with all the phase extensions of the reflection.*

None of these models consider the shift phases Ω_{ij}. Of course, they could be extended further if we added the twisted swap $S_w(\Omega)$ in the unitary evolution operator. Nevertheless, this is beyond the scope of this thesis. We are only interested in the extensions of the reflection operator R, which acts as the coin of the coined quantum walk, thus preserving the structure of a usual walk.

4.2. Equivalence with the Coined Quantum Walk

In Section 3.5 we reviewed the established equivalence between the coined quantum walk and the standard Szegedy model [130]. In this section we expand this analysis including the complex-phase extensions of the graph-phased Szegedy quantum walk, showing how it can host a wider set of equivalent coins.

The phase extensions that we are considering in this thesis enter at the level of the reflection operator. Thus, again the flip-flop shift operator S_f corresponds to the swap S_w, and the equivalence with the graph-phased Szegedy quantum walk needs the operator $R(\vec{\theta}, \varphi)$ in (4.7) to be the coin operator C^A. Recall that it was expressed as:

$$C^A := \sum_{i=0}^{N-1} |i\rangle_1\langle i| \otimes C_i^A, \tag{4.10}$$

where there is a N-dimensional unitary operator C_i^A for each node.

Following an analogous procedure to that described in Section 3.5, let us rewrite the $|\psi_i(\varphi)\rangle$ states as:

$$|\psi_i(\varphi)\rangle = |i\rangle_1 \otimes |\omega_i(\varphi)\rangle_2 , \tag{4.11}$$

where

$$|\omega_i(\varphi)\rangle_2 := \sum_{k=0}^{N-1} e^{\mathrm{i}\varphi_{ik}} \sqrt{G_{ki}}\, |k\rangle_2 . \tag{4.12}$$

Taking into account that $\mathbb{1}_{N^2} = \mathbb{1}_N \otimes \mathbb{1}_N$, the completeness relation of the identity for the first register, and substituting the expression for the $|\psi_i(\varphi)\rangle$ states in (4.7), we obtain:

$$R(\vec{\theta}, \varphi) = \sum_{i=0}^{N-1} |i\rangle_1\langle i| \otimes \left[(1 - e^{\mathrm{i}\theta_i})\, |\omega_i\rangle_2\langle\omega_i| - \mathbb{1}_N\right]. \tag{4.13}$$

Looking at the expression for the coin C^A in (4.10), we can identify the individual coins C_i^A with the phase rotations of the second register:

$$C_i^A = (1 - e^{\mathrm{i}\theta_i})\, |\omega_i\rangle_2\langle\omega_i| - \mathbb{1}_N. \tag{4.14}$$

When the augmented coins can be expressed in this form, then the coined quantum walk is equivalent to the graph-phased Szegedy quantum walk. The coins codify the transition probabilities G_{ij} and also the extended phases. Moreover, note that the reduced Szegedy subspace $\mathcal{H}_S^R$ defined in (3.36), which represents the original Hilbert space of the coined quantum walk $\mathcal{H}_C$, keeps being invariant after the introduction of the complex-phase extensions.

Now we can analyze the conditions that must be satisfied so that a coin can be expressed as in (4.14). Again, in the case that we are provided with a graph-phased Szegedy quantum walk, we can always define the coins that way, so that all Szegedy quantum walks that come from the quantization of a classical Markov chain with transition matrix G and extended phases $\vec{\theta}$ and φ can be cast into the coined model. Recall that in the case that the weighted graph is obtained normalizing the columns of the adjacency matrix of an undirected graph, the equivalent coin is the Grover coin [90, 131], although now can be extended with phases.

If, on the contrary, we are provided with a coined quantum walk, only a restricted set of coins satisfy equation (4.14). From it, we can formulate the following lemmas about the conditions that must be satisfied by a coined walk to be cast into the Szegedy quantum walk. Let us start reformulating Lemma 3.1 for the standard Szegedy model.

Lemma 4.1. *Given a coined quantum walk U_c with a set of coin operators C_i of dimension d_i, there exists an equivalent standard Szegedy quantum walk U_s if and only if for each coin C_i there are $d_i - 1$ eigenvalues -1, and a single eigenvalue $+1$, whose eigenvector has real non-negative amplitudes.*

In this case the coin must be a reflection around the state $|\omega_i\rangle$, so that $|\omega_i\rangle$ is an eigenstate with eigenvalue $+1$, and the rest of eigenvalues are -1. Moreover, all the amplitudes of $|\omega_i\rangle$ are real positive or zero [130]. Note that, since eigenvectors that differ in a global phase are equivalent, the actual condition for the eigenvector is that there are no relative phases between the amplitudes.

Lemma 4.2. *Given a coined quantum walk U_c with a set of coin operators C_i of dimension d_i, there exists an equivalent link-phased Szegedy quantum walk $U_s(\varphi)$ if and only if for each coin C_i there are $d_i - 1$ eigenvalues -1, and a single eigenvalue $+1$.*

In this case the coin must be a reflection around the state $|\omega_i(\varphi)\rangle$. Again, all the eigenvalues must be -1 except for the eigenstate $|\omega_i(\varphi)\rangle$, which is $+1$. However, thanks to the link phases this eigenstate can have any complex amplitudes. So that this model can host a bigger set of coins.

Lemma 4.3. *Given a coined quantum walk U_c with a set of coin operators C_i of dimension d_i, there exists an equivalent vertex-phased Szegedy quantum walk $U_s(\vec{\theta})$ if and only if for each coin C_i there are $d_i - 1$ eigenvalues -1, and a single eigenvalue $-e^{i\theta_i}$, whose eigenvector has real non-negative amplitudes.*

If we add APR phases, the coin becomes a phase rotation operator. Thus, $|\omega_i\rangle$ is an eigenstate with eigenvalue $-e^{\mathrm{i}\theta_i}$, and the rest of eigenvalues are -1. Note that if we would only have a model with global APR, all the coins should have the same eigenvalue different to -1. However, thanks to the local APR phases introduced in this thesis, each node can have a coin with different eigenvalues.

Lemma 4.4. *Given a coined quantum walk U_c with a set of coin operators C_i of dimension d_i, there exists an equivalent graph-phased Szegedy quantum walk $U_s(\vec{\theta}, \varphi)$ if and only if for each coin C_i there are $d_i - 1$ eigenvalues -1, and a single eigenvalue $-e^{i\theta_i}$.*

Considering both link phases and local APR phases, the eigenstate $|\omega_i(\varphi)\rangle$ can have any complex amplitude, at the same time that the eigenvalue is arbitrary. Therefore the set of coins that can be cast is maximal, and encompasses the previous cases.

Again, if we can cast a coined walk into Szegedy for the single-step operator U_s, then it can trivially also be cast into the Szegedy walk considering the original double operator W_s, so that one step of the Szegedy quantum walk would be equivalent to two steps of the coined walk, being the equivalent operator U_c^2. Moreover, as we already mentioned, there can be cases where a coin cannot be cast into Szegedy's model considering the single-step operator U_s, but it can be cast if we consider the double-step operator W_s instead. We study an example for the $-\mathbb{1}$ coin in Section 4.4.1.

4.2.1. Example on the line

In Section 3.5.1 we showed how the standard Szegedy and coined models can be cast into the other using examples on the 1D line, whose undirected graph is shown in Figure 3.3. Specifically, we observed that the standard Szegedy walk on the unbiased line of Figure 3.9(a) corresponds to the use of the Grover coin X in two dimensions, and the Hadamard coined walk corresponds to a Szegedy walk on a biased weighted graph, shown in Figure 3.9(b).

Now, let us present a different coin that cannot be cast into the standard Szegedy model. We have constructed the following coin:

$$\widetilde{N} = -\frac{1+\mathrm{i}}{2}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}. \tag{4.15}$$

The spectrum is $\sigma(\widetilde{N}) = \{-\mathrm{i}, -1\}$ and the eigenvector for the eigenvalue $-\mathrm{i}$ is $(1, \mathrm{i})^T/\sqrt{2}$. Thus, by Lemma 4.1 it cannot be cast into the standard model, but by Lemma 4.4 it can be cast into the graph-phased model. In this case the eigenvalue different to -1 is $-e^{\mathrm{i}\theta_i}$, so that the APR phase for this coin is $\theta_i = \pi/2$. The transition probabilities are obtained taking the squared modulus of the amplitudes of the eigenvector. In this case we obtain $1/2$ for both directions, so that the transition matrix $G_{\widetilde{N}}$ associated to this coin is the same as for the undirected graph G_u in (3.46). Since the amplitudes are complex numbers, we also need link phases. For a directed edge pointing to the right, the amplitude is a real positive number, and the link phase is $\varphi_{i,i+1} = 0$. However, for a directed edge pointing to the left, the amplitude is i, so the link phase is $\varphi_{i,i-1} = \pi/2$. Thus, the link phases matrix for this coin is

$$\left(\varphi_{\widetilde{N}}\right)_{ij} = \frac{\pi}{2}\delta_{i,j+1}. \tag{4.16}$$

To compare this quantum walk with the results of the H and X coins, we need to choose an initial state for the simulation. We take as initial state $|\psi_0\rangle = (|0\rangle_1 |1\rangle_2 + |0\rangle_1 |-1\rangle_2)/\sqrt{2}$, which

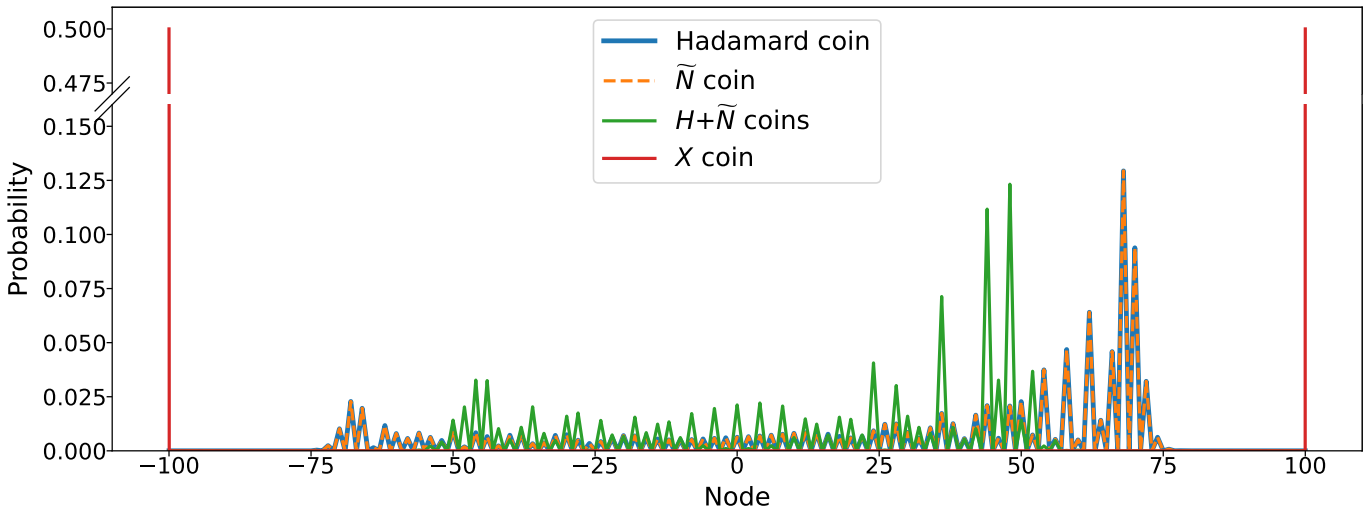


Figure 4.1: *Probability distribution results of the coined quantum walk simulations on a 1D line using the X coin (red), the Hadamard coin (blue), the $\widetilde{N}$ coin (orange), and the coined walk with the Hadamard coin for even nodes and the $\widetilde{N}$ coin for odd nodes (green). Note that for the Hadamard and $\widetilde{N}$ coins the curves overlap. The initial state is $|\psi_0\rangle = (|0\rangle_1 |1\rangle_2 + |0\rangle_1 |-1\rangle_2)/\sqrt{2}$, and the unitary evolution has been applied 100 time steps.*

represents the walker at node 0 for a Szegedy quantum walk, and provided the results of Figure 3.4(c). The probability distributions after 100 time steps are shown in Figure 4.1. The probability distribution for the $\widetilde{N}$ coin, shown in orange, is the same as for the Hadamard coin, shown in blue. This is surprising since for the $\widetilde{N}$ coin the associated weighted graph is unbiased, being the same as for the X coin. Thus, the extended phases play an important role in the walk, being able to modify somehow the transition probabilities.

Since both the Hadamard coin and the $\widetilde{N}$ coin produce the same results when they are used as global coins, we want to examine what happens if we apply both at the same time on different nodes. We have quantized the walk on the line using the Hadamard coin for even nodes, and the $\widetilde{N}$ coin for odd nodes. Taking into account our convention for the transition matrix indexes, it is constructed by taking the even columns of G_H and the odd columns of $G_{\widetilde{N}}$. For the link phases matrix φ we take the odd rows of $\varphi_{\widetilde{N}}$, and the even rows are null since the Hadamard coin has no link phases. The local APR phase for even nodes is the standard one $\theta_i = \pi$, and for odd nodes it is $\theta_i = \pi/2$. The results of the simulation are shown in Figure 4.1 in green. We observe that, despite the fact that H and $\widetilde{N}$ produce the same results when they are global coins, the results are different when they are mixed. Although not shown, a similar phenomenology is obtained for the different initial coin states of Figure 3.4. This explicitly shows that both coins are not actually equivalent, and that the graph-phased Szegedy model opens up a wide range of possibilities for quantizing classical Markov chains.

4.3. Quantum Circuits

In this section we leverage the quantum circuits for the standard Szegedy walk, whose construction was reviewed in Section 3.6, to construct the circuits of the extended models with slight modifications. Thus, provided a quantum circuit for the standard Szegedy quantum walk, we can obtain the circuit of the graph-phased model.

4.3.1. Adding APR phases

Let us consider the vertex-phased Szegedy model, where the reflection R is substituted by the phase rotation operator $R(\vec{\theta})$ in (4.5) including local APR. We diagonalize it using the update operator V, as in Section 3.6, so that $D(\vec{\theta}) = V^\dagger R(\vec{\theta}) V$. Then:

$$\begin{aligned} D(\vec{\theta}) &= \sum_{i=0}^{N-1} (1 - e^{\mathrm{i}\theta_i}) V_i^\dagger \left|\psi_i\right\rangle \left\langle\psi_i\right| V_i - \mathbb{1} = \sum_{i=0}^{N-1} (1 - e^{\mathrm{i}\theta_i}) \left|i\right\rangle_1 \langle i| \otimes \left|0\right\rangle_2 \langle 0| - \mathbb{1} \\ &= \sum_{i=0}^{N-1} \left|i\right\rangle_1 \langle i| \otimes \left[(1 - e^{\mathrm{i}\theta_i}) \left|0\right\rangle_2 \langle 0| - \mathbb{1}_N\right]. \end{aligned} \tag{4.17}$$

In the case of a global APR phase θ, we could again factorize it as a diagonal operator acting on the second register the same as in equation (3.52), which would correspond to a phase rotation around the state $\left|0\right\rangle_2$. This would have eigenvalue $-e^{\mathrm{i}\theta}$ for $\left|0\right\rangle_2$ and -1 for the rest. The circuit would implement $-D(\theta)$ instead, so that applies a phase $e^{\mathrm{i}\theta}$ to the state $\left|0\right\rangle_2$ letting the rest of the computational basis unchanged. Then, the circuit would be the same as the one shown in Figure 3.10, but changing the phase π by the general phase θ.

However, if we have local APR phases, the diagonal operator cannot be factorized. In this case, it corresponds to a conditional operator, which applies a different phase rotation on the second register around the state $\left|0\right\rangle_2$, depending on the state of the first register. In Section 2.3.1, we showed that it can be compiled as a uniformly controlled gate [88,89]. Thus, in general, it corresponds to N different phase rotations acting on the second register, controlled by the N states of the computational basis of the first register. The quantum circuit for this case is shown in Figure 4.2(a).

The construction of this circuit is in general not very efficient, since we need N controlled phase rotation gates. However, there can be cases where a more efficient circuit is possible. The local APR phases can be distributed in some manner that the same controlled phase rotation can be used to apply it to a bunch of states at the same time. For example, if we have the same phase θ_e for even nodes, and the same phase θ_o for odd nodes, they can be implemented using only two controlled phase rotations. In this case they would be controlled by the last qubit of the first register, which indicates the parity of the state.

Another important case where an efficient implementation is possible occurs when most of the nodes have the same phase θ, and there is a small set $\mathcal{M}$ of special nodes with a different phase θ_k, where $k \in \mathcal{M}$. If we have a number of special nodes $M << N$, there is a method for constructing the circuit with only $M + 1$ phase rotation gates. First, we apply a phase rotation of θ to the second register, with no control by the first register. Thus, a global APR phase θ is applied for all the

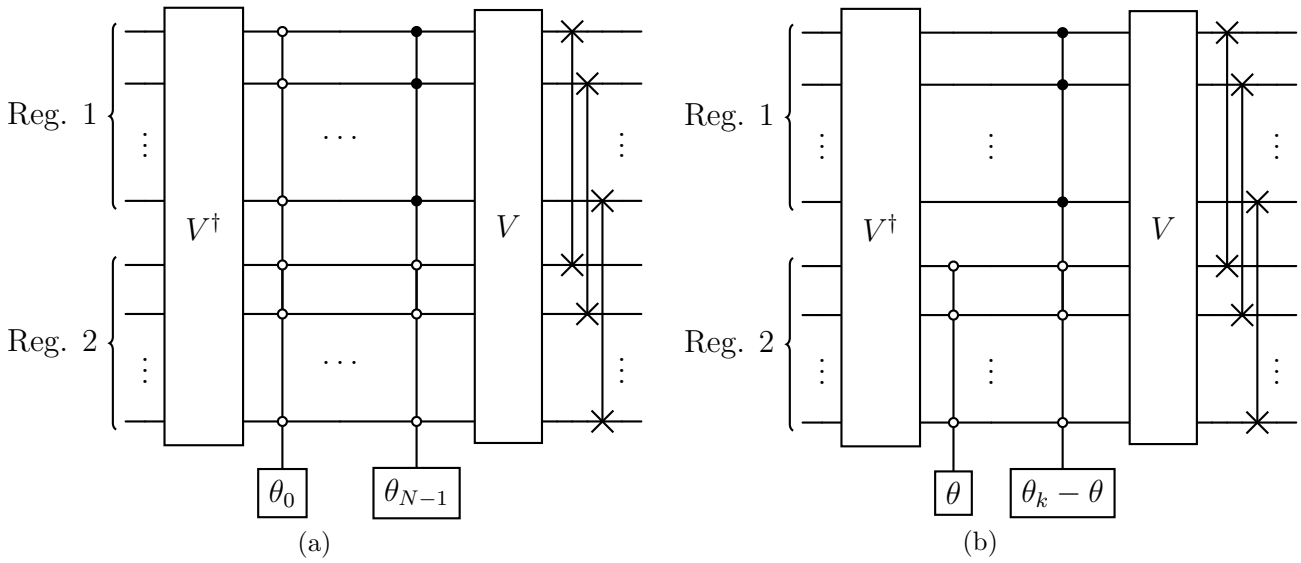


Figure 4.2: *(a) Quantum circuit decomposition of the single unitary operator $U_s(\vec{\theta}) = S_w R(\vec{\theta})$ for the vertex-phased Szegedy quantum. Each register has n qubits for a graph with $N = 2^n$ nodes. The phase rotation operator $R(\vec{\theta}) = VD(\vec{\theta})V^\dagger$, where $D(\vec{\theta})$ is a diagonal operator that implements N different phase rotations around the state $|0\rangle_2$ in the second register. Thus, this circuit needs N phase rotations controlled by the first register, from the state $|0\rangle_1$ with all qubits in $|0\rangle$ to the state $|N-1\rangle_1$ with all qubits in $|1\rangle$. (b) Quantum circuit decomposition of the single unitary operator $U_s(\vec{\theta}) = S_w R(\vec{\theta})$ for the vertex-phased Szegedy quantum walk on a graph where most nodes have the same local APR phase θ, and M special nodes have a different local APR phase. A phase rotation of θ is applied to all the nodes, and then M different phase rotations controlled by the first register are applied for the special nodes. In this example, only node $N-1$ has a different phase θ_k, so that the controlled phase rotation needs all the qubits in the first register to be in the state $|1\rangle$.*

nodes. After that, we apply a phase rotation for each of the M special nodes, controlled by the first register being in the corresponding state. Let θ_k be the phase we want to apply to node k, which is an special node. Since we have already applied a phase θ to that node, the particular phase rotation for that node must now apply $\theta_k - \theta$, so that it deletes the previous rotation with θ. In Figure 4.2(b) we show an example where node $N-1$ is the only special node. Note that, in general, each of the special nodes can have a different local APR phase.

4.3.2. Adding link phases

We have seen that after diagonalizing the reflection operator R, the diagonal operator D, even with APR phases, does not depend on the vectors $|\psi_i\rangle$. Thus, it will not depend on the link phases φ_{ij}. In this case, the modification must be done to the update operator V. The diagonalization process is the same as before, but using the modified update operator $V(\varphi)$, such that $V(\varphi)\,|i\rangle_1\,|0\rangle_2 := |\psi_i(\varphi)\rangle$.

Provided a circuit implementation of the update operator V, a naive method to construct the operator $V(\varphi)$ would be to first apply V, and then apply a controlled phase gate (see Figure 2.7) for each computational basis state whose link phase φ_{ij} is different to 0. In general we would have N^2 link phases, so this method would be inefficient unless there are very few link phases different

to 0, or they are distributed in a manner that several phases with the same value can be applied using a common controlled gate. Note that this procedure would be the same in case we wanted to implement the twisted swap $S_w(\Omega)$, since it is just a swap followed by the application of the relative phases.

Another approach consists of constructing a new circuit for the modified update operator based on the standard one. The same as with the transition probabilities, if the link phases matrix φ is sparse or the distribution of phases in the graph follows some symmetry patterns, we could construct an efficient operator $V(\varphi)$. An example of update operator with phases distributed with certain symmetry in cycles is constructed in the next subsection.

4.3.3. Quantum circuit for the $H + \widetilde{N}$ coined quantum walk

As an example, we want to provide a quantum circuit construction for the graph-phased Szegedy quantum walk equivalent to the coined walk of Section 4.2.1, which uses two coins at the same time. Recall that a quantum walk in the infinite 1D infinite line can be simulated in a finite 1D cycle as long as the time steps do not surpass a threshold from which the wavefront interferes with itself when it completes a turn on the cycle. For a graph with $N = 2^n$ nodes, we need n qubits for each register.

In this case, the local APR phases are distributed with a symmetric pattern, so that the diagonal operator $D(\vec{\theta})$ can be implemented efficiently. The last qubit of the first register indicates the parity of the node. If this control qubit is in the state $|0\rangle$, for even nodes we apply the phase rotation with $\theta = \pi$ on the second register. If, on the contrary, it is in the state $|1\rangle$, for odd nodes the phase rotation with $\theta = \pi/2$ is applied instead. Thus, the quantum circuit can be decomposed as shown in Figure 4.3(a).

For the update operator $V(\varphi)$, in Figure 4.3(b) we have constructed a circuit following an analogous procedure to the one in Section 3.6 for the standard cycle graph. The update operator must act as $V(\varphi)\,|i\rangle_1\,|0\rangle_2 = |i\rangle_1\,|\omega_i(\varphi)\rangle_2$, where

$$|\omega_i(\varphi)\rangle_2 = \begin{cases} h_R\,|i+1\rangle_2 + h_L\,|i-1\rangle_2 & \text{if } i \text{ is even,} \\ \dfrac{1}{\sqrt{2}}\,|i+1\rangle_2 + \dfrac{\mathrm{i}}{\sqrt{2}}\,|i-1\rangle_2 & \text{if } i \text{ is odd,} \end{cases} \tag{4.18}$$

where h_R and h_L were given in (3.47) and (3.48), respectively. Since we are considering a finite graph, the additions and subtractions are performed modulo N.

Let i be an even node. Then, the last qubit of the first register is in the state $|0\rangle$, and the R_Y gate is applied to the first qubit of the second register, letting it in the state $h_R\,|0\rangle_2 + h_L\,|1\rangle_2$. The following action of the CNOT gates and the last X gate let the second register in

$$h_R\,|1\rangle_2 + h_L\,|N-1\rangle_2\,. \tag{4.19}$$

We have previously seen that the join action of the $\mathcal{P}^+$ operators controlled by the first register corresponds to another permutation that transforms the computational basis states $|i\rangle_1\,|x\rangle_2$ into $|i\rangle_1\,|x+i \bmod N\rangle_2$ [91]. Therefore, when they are applied on the state in (4.19), the second register ends up in $|\omega_i(\varphi)\rangle_2$.

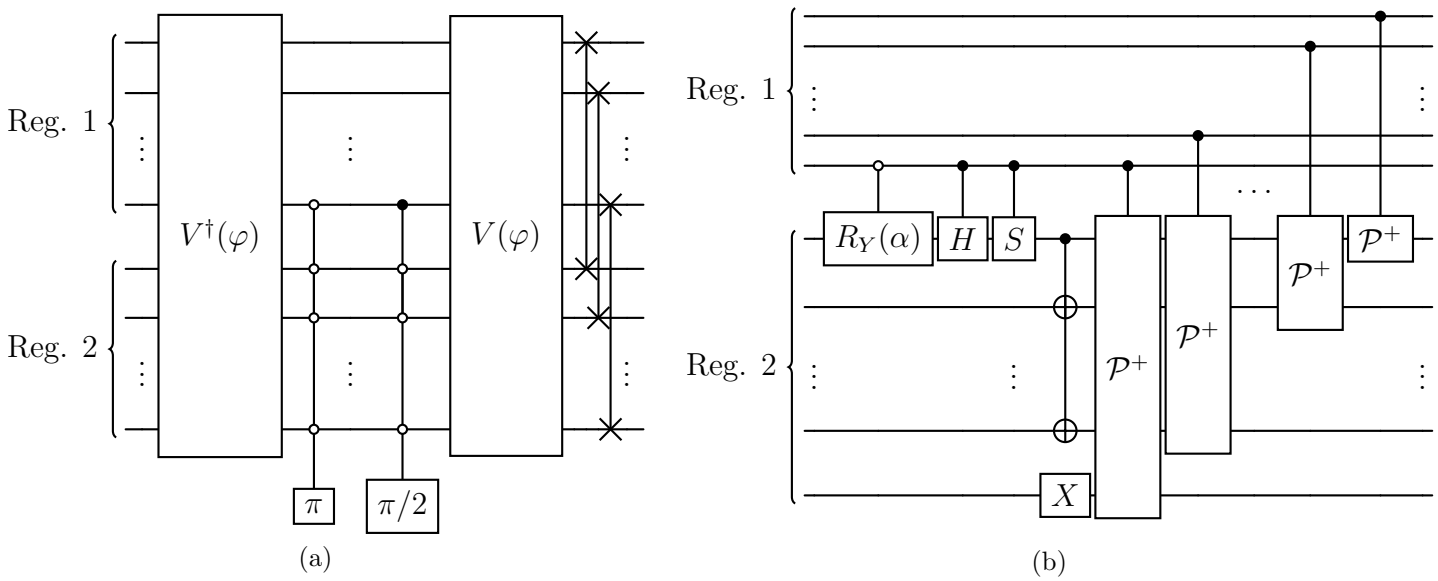


Figure 4.3: *(a) Quantum circuit decomposition of the graph-phased single-step Szegedy unitary evolution operator $U_s = S_w R(\vec{\theta}, \varphi)$ equivalent to the coined walk with the H coin for even nodes and the $\widetilde{N}$ coin for odd nodes. Each register has n qubits for a cycle with $N = 2^n$ nodes. The last qubit of the first register, which determines the parity of the state, controls the application of a local APR phase π for even nodes, and $\pi/2$ for odd nodes. (b) Quantum circuit for the update operator $V(\varphi)$ in (a). The rotation matrix $R_Y(\alpha)$ is given in (2.39), and the value of the rotation angle is $\alpha = \pi/4$.*

We can do an analogous reasoning for i being an odd node. In this case, the last qubit of the first register is in the state $|1\rangle$, so that the H and S gates are applied to the first qubit of the second register, letting it in the state $(|0\rangle + \mathrm{i}\,|1\rangle)/\sqrt{2}$. After the CNOT gates and the last X gate the second register ends up in

$$\frac{1}{\sqrt{2}}\,|1\rangle_2 + \frac{\mathrm{i}}{\sqrt{2}}\,|N-1\rangle_2\,, \tag{4.20}$$

and the controlled $\mathcal{P}^+$ operators permute the state into $|\omega_i(\varphi)\rangle_2$. Thus, we have proved that the circuit in Figure 4.3(b) effectively performs the action of the update operator $V(\varphi)$.

The same as the circuit of the standard walk, this is just a toy model to show a construction procedure for a simple graph-phased Szegedy quantum walk. Of course, we could also construct a simpler circuit from the coined walk perspective. We would just take the circuit in Figure 3.6(b) and replace the coin C with the coin H controlled by the last qubit of the position register being in the state $|0\rangle$, and the coin $\widetilde{N}$ controlled by the last qubit of the position register being in the state $|1\rangle$.

4.4. Marking Nodes with APR

Suppose there is a special set of nodes in a graph that we want to mark somehow so that the quantum evolution treats them in a different manner. We can mark them with a different local APR phase. A construction based on Figure 4.2(b) would require the knowledge about who are these special nodes. However, we usually do not know them, and they are marked by a black-box function when they satisfy some conditions.

Let $f(x)$ be a classical function that decides if a particular node x satisfies some conditions, and let $\mathcal{M}$ be the set of nodes that satisfy them. Then $f(x) = 1$ if $x \in \mathcal{M}$, and $f(x) = 0$ otherwise. In Section 2.3.6, we saw that given a classical circuit for this function $f(x)$, it can be translated into an oracle O_f. Recall that this operator takes a quantum register with the information about the node x and adds the result of $f(x)$ in an ancilla qubit [19, 70], such that

$$O_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle . \tag{4.21}$$

For the sake of simplicity, let us consider that we want to mark some nodes in a standard Szegedy walk, so that they evolve with a different local APR phase when they satisfy the conditions of an oracle O_f. We can construct a quantum circuit leveraging the update operator V of the standard walk, as shown in Figure 4.4. In this case, we make use of an ancilla qubit that takes the value of $f(x)$ to later control the phase rotation that deletes the global phase π and let the local APR phase θ_k for the marked nodes. Thus, a marked node $k \in \mathcal{M}$ evolves with the local APR phase θ_k, instead of the standard phase π. The second oracle is used to uncompute the action of the first one, so that the ancilla qubit returns to its original state and can be traced out. Note that although we have used a single oracle O_f for marking nodes, we can use multiple oracles to mark different sets of nodes with different phases, so that they evolve differently depending on the different conditions imposed by the oracles.

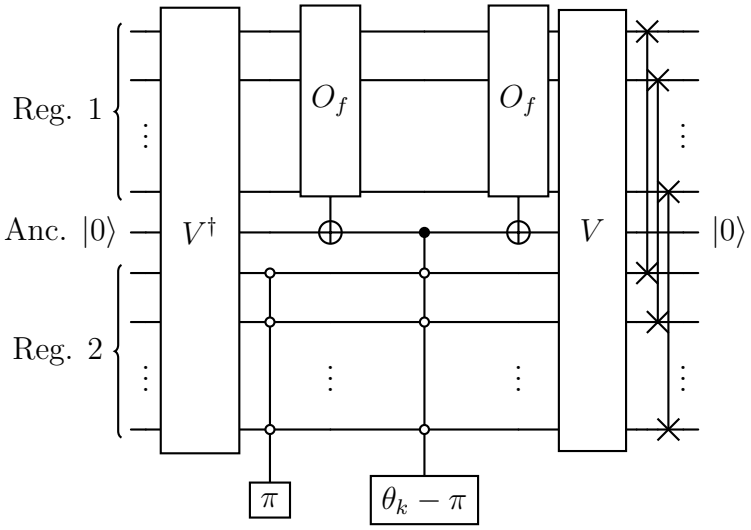


Figure 4.4: *Quantum circuit decomposition for marking nodes with local APR phases in a standard Szegedy quantum walk. The first oracle turns the ancilla qubit into* $|1\rangle$ *if the first register represents a marked nodes. Then, the ancilla applies a phase rotation on the second register deleting the global APR phase* π *and letting the local APR phase* θ_k*. Finally, the second oracle uncomputes the ancilla so that it returns to its original state and can be traced out.*

4.4.1. Searching marked nodes

The link phases have been previously used to search for marked arcs on graphs [171]. In this section, we show that an important use case of the local APR phases is also the search problem. Nevertheless,

in this case we want to use a quantum walk algorithm to search for marked nodes instead of marked arcs.

The first quantum walk search algorithm used the Grover coin G_D for unmarked nodes, and a different coin for marked nodes [44]. In particular, it was studied the case of the $-\mathbb{1}$ coin for marking nodes, finding an algorithm with quadratic speedup with respect to classical ones. This coin cannot be cast into the standard Szegedy walk since all the eigenvalues are -1. However, by Lemma 4.3 it can be cast into the vertex-phased Szegedy model using $\theta_k = 0$ for marked nodes. As a remark, note that this algorithm is different from the Grover search as a quantum walk [146] shown Section 3.9, which uses the minus Grover coin, $-G_D$, for the marked nodes. For a dimension greater than 2, this coin has more than one eigenvalue $+1$, and thus cannot be cast into any extension of the Szegedy walk.

A different approach for quantum walk search algorithms is based on the Szegedy quantum walk with absorbing vertices [40, 45]. Given a transition matrix G for a graph, the nodes are marked turning them into sinks, so that they have a self-loop that does not allow the walker to escape once the walker steps on them. Let us denote the modified transition matrix as G'. This is obtained deleting the columns of the marked nodes and allocating a diagonal 1 element, so that

$$G'_{ji} = \begin{cases} G_{ji} & \text{if } i \notin \mathcal{M}, \\ \\ \delta_{ji} & \text{if } i \in \mathcal{M}. \end{cases} \tag{4.22}$$

It was previously stated that marking with the $-\mathbb{1}$ coin is equivalent to marking with absorbing vertices, although the proof was done assuming that the initial state has no amplitudes associated to self-loops [131]. In this thesis, we show that both walks are indeed not equivalent for arbitrary states, although they are equivalent when the double-step Szegedy operator is considered. Moreover, we make the proof for general graphs, so that the global coin for unmarked nodes does not have to be the Grover coin.

On the one hand, the evolution operator for the Szegedy quantum walk using the $-\mathbb{1}$ coin for marking nodes is $U_s(\vec{\theta}) = S_w R(\vec{\theta})$, and is obtained setting $\theta_i = \pi$ for the unmarked nodes and $\theta_i = 0$ for the marked nodes in (4.5). On the other hand, the evolution operator for the Szegedy quantum walk with absorbing vertices U'_s is obtained from the standard operator in (3.34) using the transition matrix G'. If we express the reflection operators in a coined form, in a similar manner as in equation (4.13), and separate the sums for marked and unmarked nodes, the expressions of these operators are the following:

$$U_s(\vec{\theta}) = S_w \sum_{i \notin \mathcal{M}} |i\rangle_1 \langle i| \otimes (2 |\omega_i\rangle_2 \langle \omega_i| - \mathbb{1}_N) + S_w \sum_{i \in \mathcal{M}} |i\rangle_1 \langle i| \otimes (-\mathbb{1}_N), \tag{4.23}$$

$$U'_s = S_w \sum_{i \notin \mathcal{M}} |i\rangle_1 \langle i| \otimes (2 |\omega_i\rangle_2 \langle \omega_i| - \mathbb{1}_N) + S_w \sum_{i \in \mathcal{M}} |i\rangle_1 \langle i| \otimes (2 |i\rangle_2 \langle i| - \mathbb{1}_N). \tag{4.24}$$

A priori both operators are not equivalent, since for the marked nodes $U_s(\vec{\theta})$ applies the $-\mathbb{1}$ coin, whereas U'_s applies a reflection around the self-loop state of the marked node.

We need to calculate how these operators act on the different states of the computational basis in order to examine properly the equivalence. For both operators the first term, corresponding to

the unmarked nodes, is the same. This is so because the APR phase for them is $\theta_i = \pi$, and the corresponding columns of G and G' are also the same. Thus, the action on a computational basis state $|i\rangle_1 |j\rangle_2$ with $i \notin \mathcal{M}$ is the same. Now, let us consider a state $|i\rangle_1 |j\rangle_2$ with $i \in \mathcal{M}$. In this case the action is given by the second terms of the operators. If $i \neq j$, the action is $-S_w$ in both cases, so it is the same. However, if $i = j$, the action of $U_s(\vec{\theta})$ is $-S_w$, whereas the action of U'_s is $+S_w$, so that $U_s(\vec{\theta})$ acts as $-U'_s$.

Let us define $\mathcal{I}_{ML}$ as the subspace spanned by the marked self-loop states of the computational basis, i.e.,

$$\mathcal{I}_{ML} := \text{span}\left\{|i\rangle |i\rangle \ : \ i \in \mathcal{M}\right\}. \tag{4.25}$$

This space is an eigenspace for both operators, with eigenvalue $+1$ for U'_s and -1 for $U_s(\vec{\theta})$. Thus, it and its orthogonal complement are invariant subspaces. We can then factorize the action as follows:

$$U_s(\vec{\theta}) |i\rangle |j\rangle = \begin{cases} -U'_s |i\rangle |j\rangle & \text{if } |i\rangle |j\rangle \in \mathcal{I}_{ML}, \\ \\ U'_s |i\rangle |j\rangle & \text{if } |i\rangle |j\rangle \in \mathcal{I}^{\dagger}_{ML}. \end{cases} \tag{4.26}$$

Although both operators are not the same, since both subspaces are invariant, the action is equivalent when the initial state is in $\mathcal{I}^{\dagger}_{ML}$. This corresponds to a state that has no amplitudes related to self-loops, so that the previously known equivalence in this case holds [131]. Nevertheless, note that for general states with self-loops, the only difference is a relative phase -1 in the amplitudes of the self-loops. This relative phase plays no role in the probabilities of measuring the nodes in the computational basis, so that the results would be the same even in this case although the quantum states are not exactly equal. However, this equivalence would not be valid in case we measured in other different basis.

Usually, in quantum walk search algorithms based on absorbing vertices, the double-step Szegedy operator W_s is used. Since the subspaces are invariant, we can take the square of equation (4.26), obtaining that

$$W_s(\vec{\theta}) = W'_s. \tag{4.27}$$

Thus, for this operator both walks are totally equivalent, taking into account that one Szegedy step with absorbing vertices corresponds actually to two steps of the coined walk with the $-\mathbb{1}$ coin. Recall that by Lemma 4.1 the $-\mathbb{1}$ coin cannot be cast into a single step of the standard Szegedy model. However, this lemma does not apply for the double Szegedy operator, since we have managed to cast this coin into an operator W'_s without phase extensions.

4.4.2. Example: complete graph

A well studied graph for quantum walk search algorithms based on absorbing vertices is the complete graph without loops [135]. The transition matrix is $G_{ji} = (1 - \delta_{ji})/(N - 1)$. The initial state of the system is constructed from the transition matrix of the original complete graph without marked nodes as:

$$\left|\Psi^{(0)}\right\rangle := \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\psi_i\rangle . \tag{4.28}$$

In this case, the initial state has no self-loop amplitudes, so that the coined walk would be equivalent to the single-step operator U'_s. Nevertheless, the quantum walk operator is usually W'_s. A maximum of probability of measuring one of the marked nodes occurs after a number of time steps given by [135]

$$t_{max} = \frac{\pi}{4}\sqrt{\frac{N}{2M}} - \frac{1}{4} + \mathcal{O}\left(\sqrt{\frac{M}{N}}\right), \tag{4.29}$$

where M is the number of marked nodes. Thus, the algorithm has a quadratic speedup with respect to a classical search, the same as the coined quantum walk.

In order to verify numerically that both models are equivalent, we have simulated the Szegedy walk with $W_s(\vec{\theta})$ and W'_s for a graph with $N = 1000$ nodes and $M = 2$ marked nodes. The results are shown in Figure 4.5, where we observe that both curves overlap. Moreover, note that the probability oscillates, despite the fact that in the underlying classical walk the walker cannot escape from absorbing vertices. This is due to the unitary character of the evolution, and is related to the ghost links with null transition probability that play an actual role in the Szegedy quantum walk, as discussed in Section 3.5.

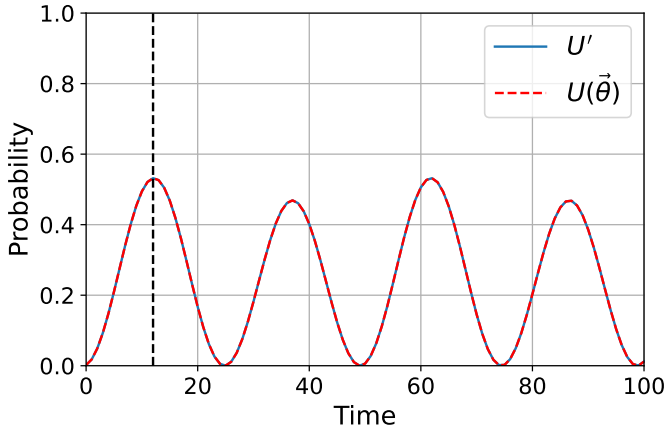


Figure 4.5: *Probability of measuring a marked node versus the time step of a quantum walk search algorithm on a complete graph with $N = 1000$ nodes and $M = 2$ marked nodes. The nodes have been marked with absorbing vertices in blue, and with local APR phases $\theta_k = 0$ in red. Both algorithms are equivalent, so that the curves overlap. The vertical dashed line indicates the theoretical position of the first maximum, which occurs for $t_{max} \approx 12$.*

An efficient implementation of the update operator V for the complete graph can be found in the literature [91]. We can then use the construction in Figure 4.4 to construct a quantum circuit based on an oracle for this quantum walk search algorithm, avoiding the explicit construction of the update operator V' for the graph with absorbing vertices. Although a form of constructing a circuit for V' using V and an oracle was already developed [130, 174], this construction needed the application of the oracle two times, so that for the operator U'_s the oracle would be applied four times, instead of two as in our circuit. Furthermore, it needed a controlled version of the operator V, increasing the complexity of the circuit. Therefore, our implementation is simpler. Moreover, our circuit can mark with different phases, so that it is not restricted only to absorbing vertices.

4.5. Summary of Results and Conclusions

- We have reviewed the current phase extensions of the Szegedy quantum walk in the literature based on link phases, to generalize it including global and local APR phases. Then, we have defined different Szegedy models depending on each kind of phase extensions, and the graph-phased Szegedy quantum walk including all the phase extensions.

- We have reviewed the equivalence between the Szegedy quantum walk and the coined model, considering also the different phase extensions, and formulated some lemmas about the conditions that need to satisfy the coin operators. The phase extensions can be used to cast a wider set of coins into the Szegedy walk, and, of course, the graph-phased Szegedy model can host the bigger set of coins.

- We have seen an example complex coin that cannot be cast into the standard Szegedy walk on the 1D line, but can be cast into the graph-phased model using the phase extensions. This coin produces the same results as the Hadamard coin when the coins are the same for all the nodes. However, when a walk is quantized using both coins simultaneously, distributing them between even and odd nodes, a new different walk is obtained. This result is quite surprising, and shows that coins that *a priori* seems to be equivalent, are not under different circumstances.

- Based on the quantum circuit construction of the standard Szegedy walk, we have shown how a circuit can be modified in order to include phase extensions. In general cases where all the phase values would be different, the circuits would not be very efficient. However, the same as happens with the transition matrix for the standard Szegedy model, the phases can be distributed in some symmetric patterns so that an efficient implementation is possible. For example, if there is a small set of nodes with a different local APR phase, or there are only two local APR phases values distributed between even and odd nodes. Furthermore, we have managed to construct a quantum circuit for an example coined walk with two coins.

- An important application of the local APR phases is that they can be used to mark nodes in a graph, so that the quantum evolution treats them in a different manner. We have seen how a quantum circuit based on oracles can be constructed, so that there is no need to modify the structure of the graph, and it is more efficient than previously known constructions. Quantum search algorithms are examples where nodes are marked. We have shown that marking with a null local APR phase is totally equivalent to marking with absorbing vertices in the case of using the double Szegedy operator W_s. When using the single operator U_s, the same results are obtained after measuring in the computational basis, although both operators are not exactly equal.

- Although the $-\mathbb{1}$ coin cannot be cast into the standard Szegedy walk with the single operator U_s, we have seen that we can cast it for the double operator W_s. This result is intriguing, as shows that, when considering the double operator, the set of coins that can host is wider, not needing any complex-phase extension. More research about this fact is needed in the future.

- In the future, it would be interesting to study further applications of the graph-phased walk, including the twisted swap operator [167]. Moreover, taking into account that local APR phases act at the level of nodes, and link phases at the level of edges, there could be a relation with gauge symmetries.

Chapter 5

Quantum PageRank with Arbitrary Phase Rotations

In Section 3.10 we introduced the quantum PageRank algorithm, an algorithm based on the Szegedy quantum walk for ranking nodes in a graph. This algorithm was first implemented in small networks, showing intriguing properties such as a violation in the ranking of the nodes [31] with respect to the classical distribution. Later, it was scaled to complex networks, showing further properties such as a better resolving of the network structure and an improved stability [32]. Since then, there has been a recent interest in the quantum PageRank. For example, other quantizations have been proposed, such as coined quantum walks with reduced coins [175], continuous-time quantum walks [165, 176–178], quantum walks on open systems [179], and even variational circuits [180]. Furthermore, it has been realized experimentally in the continuous-time version using photons [181].

Given the interest in the quantum PageRank algorithm, in this Chapter we use it to test an important practical use case of the complex-phase extensions of the Szegedy quantum walk. In particular, we extend this algorithm with global arbitrary phase rotations (APR) [1]. We compare our results with those of the original quantum PageRank using first the same small graph studied in 3.10.1, and later using scale-free networks, which are complex networks that model the World Wide Web [182]. We find that after the introduction of the APR, some issues of the original algorithm are solved, at the same time that holds the quantum advantages. Although the information that we use for our results is purely classical, we must keep in mind that these results can also be of interest for future quantum networks.

This chapter is structured as follows. In Section 5.1 we introduce the quantum PageRank algorithm with arbitrary phase rotations. In Section 5.2 we study the effect of the new quantum algorithms in a small generic graph. In Section 5.3 we apply these algorithms to complex scale-free networks. In Section 5.4 we study the stability of the new algorithms. Finally, we summarize and conclude in Section 5.5.

5.1. Quantum PageRank with APR

The formalism of both the classical and quantum PageRank algorithms is explained in Section 3.10.1. As we saw, the quantum PageRank is based on the Szegedy quantum walk, described in Section 3.4. Our aim is to introduce arbitrary phase rotations in the quantum walk. Since *a priori* we have no information about the nodes of the graph before obtaining the PageRank results, there is no point in assigning a different local APR phase to each node. For this reason, we use the Szegedy quantum walk with global APR defined in Section 4.1. As we already mentioned, the quantum PageRank algorithm needs to preserve the directedness of the graph. Therefore, due to the swap, we need to apply the double-step Szegedy operator, which, for this model, has two degrees of freedom:

$$W_s(\theta_1,\theta_2) := U_s(\theta_2)U_s(\theta_1) = S_w\left(\left[1-e^{\mathrm{i}\theta_2}\right]\Pi-\mathbb{1}\right)S_w\left(\left[1-e^{\mathrm{i}\theta_1}\right]\Pi-\mathbb{1}\right), \tag{5.1}$$

where in general $\theta_1 \neq \theta_2$. The quantum PageRank distributions are obtained in the same manner as in Section 3.10.1, but with the extended operator $W_s(\theta_1,\theta_2)$ instead. This gives rise to a new family of quantum PageRank algorithms with θ_1 and θ_2 as two degrees of freedom.

If we chose $\theta_1 = \theta_2 = \pi$, the quantum PageRank algorithm using the standard Szegedy walk is recovered. To simplify the study of the effect of choosing different phases, we define three APR schemes with only a single degree of freedom θ:

Definition 5.1 (PageRank APR schemes)**.** *Given a quantum PageRank algorithm with a double-step Szegedy operator $W_s(\theta_1,\theta_2)$, and a single phase θ, the APR schemes are defined as follows:*

- *Equal-phases scheme: both phases are equal, i.e., $\theta_1 = \theta_2 = \theta$.*
- *Opposite-phases scheme: the phases have the opposite signs, i.e., $\theta_1 = -\theta_2 = \theta$.*
- *Alternate-phases scheme: the first phase is fixed to π, while the second phase is free, i.e., $\theta_1 = \pi$,$\theta_2 = \theta$.*

We have found that the results for $\theta_1 = \theta$, $\theta_2 = \pi$ are similar to the ones of the already defined alternate-phases scheme, so we do not take into account this particular case.

The phase θ can take a value in the interval $(-\pi,\pi]$. Since there are no link phases, the projector operator Π in (3.32) is real, so that inverting the sign of the phase θ would result in complex conjugating the operator $W_s(\theta_1,\theta_2)$. The initial vector state is also real, so the final result after the evolution would just be the complex conjugated. However, the quantum PageRank scores are real probabilities obtained by taking the squared modulus of the amplitudes, so there would not be any effect. For that reason, without loss of generality, we can set $\theta \in [0,\pi]$.

All that remains is to fix a value for the phase θ. We have found that a convenient value is $\theta = \pi/2$. To justify it, in the following section we study the effect of the phase θ on a simple small generic graph.

5.2. Small Generic Graph

Once we have defined the new family of quantum PageRank algorithms that we are going to study, we want to see the effect of the free parameter θ in the three APR schemes described above. For this task we use the small generic graph with seven nodes, previously introduced to study the standard quantum PageRank [31] in Section 3.10.1, which is shown again in Figure 5.1(a).

The results for the classical and standard quantum PageRank algorithms were shown in Figure 3.13. As a summary, in both algorithms the most important node is node 6, followed by node 4, whereas the least important node was node 3. Nevertheless, there is a violation of the ranking of intermediate nodes in the quantum distribution. Furthermore, the instantaneous quantum PageRank scores of different nodes fluctuates in a manner that the relative importance can be exchanged in time, and for that reason it is necessary to average the distributions.

To see the effect of the complex phase θ on the quantum PageRank distributions, we first present the results for the alternate-phases scheme. As three interesting cases we choose the following decreasing values of θ: $\pi/2$, $\pi/10$, and $\pi/100$. For $\theta = \pi/2$ the instantaneous quantum PageRank scores, shown in Figure 5.1(c), have smaller amplitudes compared to the standard ones in Figure 3.13(d), thus allowing to properly distinguish the most important nodes from the least important node. Moreover, the fluctuation is also slower. This oscillation gets slower and slower as we reduce the phase θ, as can be seen for $\theta = \pi/10$ and $\theta = \pi/100$ in Figures 5.1(d) and 5.1(e), respectively.

The time-averaged quantum PageRank distributions for the alternate-phases scheme are shown in Figure 5.1(b), together with the standard quantum and the classical PageRank distributions. As the phase decreases, the distribution gets closer to the classical one, up to a limit. Indeed, from $\theta = \pi/10$, the ranking of the nodes is the same as that in the classical case, restoring the ranking violated by the standard quantum PageRank [31]. However, the smaller θ is, the slower the oscillation of the instantaneous PageRank becomes, so it takes more time to average the dynamics properly. This results in a slower convergence of the algorithm. In Figures 5.1(f)-5.1(h) we show how the averaged quantum PageRank converges more slowly as the complex phase decreases. For this reason, we cannot decrease the angle θ arbitrarily.

Similar results are obtained for the other two APR schemes. In Figures 5.2 and 5.3 we show that the decrease of the phase θ increases the period of the quantum fluctuations for the equal-phases and opposite-phases, respectively. This turns out in a longer time for the time-averaged quantum PageRank to converge. Furthermore, it is interesting that in the opposite-phases case the quantum fluctuations get a modulated behavior.

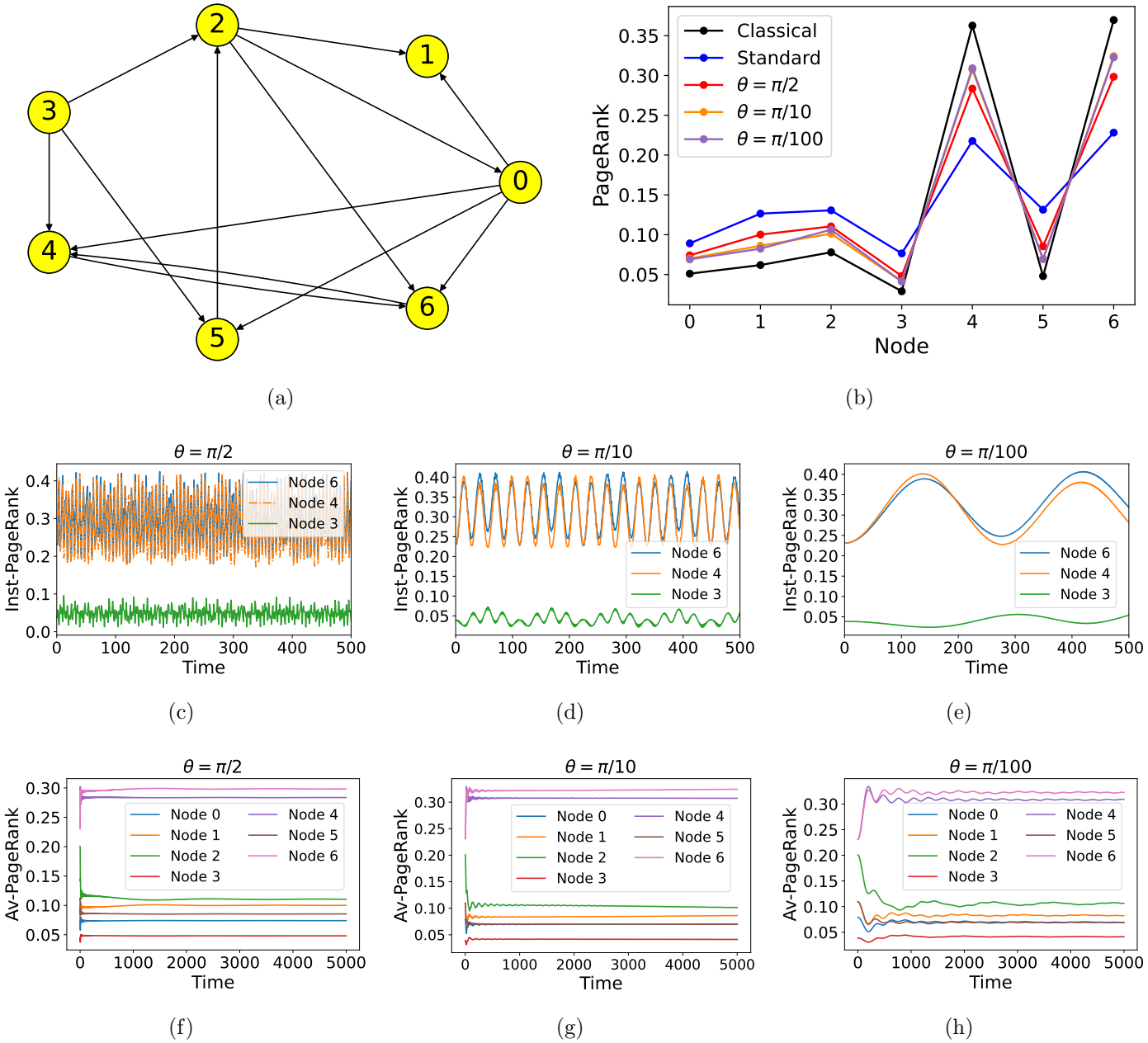


Figure 5.1: *(a) Small generic graph with $N = 7$ nodes. (b) Time-averaged quantum PageRank distributions for the alternate-phases scheme with $\theta = \pi/2$, $\pi/10$, and $\pi/100$ for the small generic graph with seven nodes. They are compared with the classical PageRank and the standard quantum PageRank distributions. (c)-(e) Instantaneous quantum PageRank scores of nodes 6, 4 and 3 of the small generic graph versus the quantum walk time for the alternate-phases scheme with (c) $\theta = \pi/2$, (e) $\theta = \pi/10$, (f) $\theta = \pi/100$. (f)-(h) Time average of the instantaneous quantum PageRank scores for all the nodes of the small generic graph versus the quantum walk time for the alternate-phases scheme with (f) $\theta = \pi/2$, (g) $\theta = \pi/10$, (h) $\theta = \pi/100$. It is observed that as θ decreases, the quantum fluctuations get slower and the algorithm takes more time to converge.*

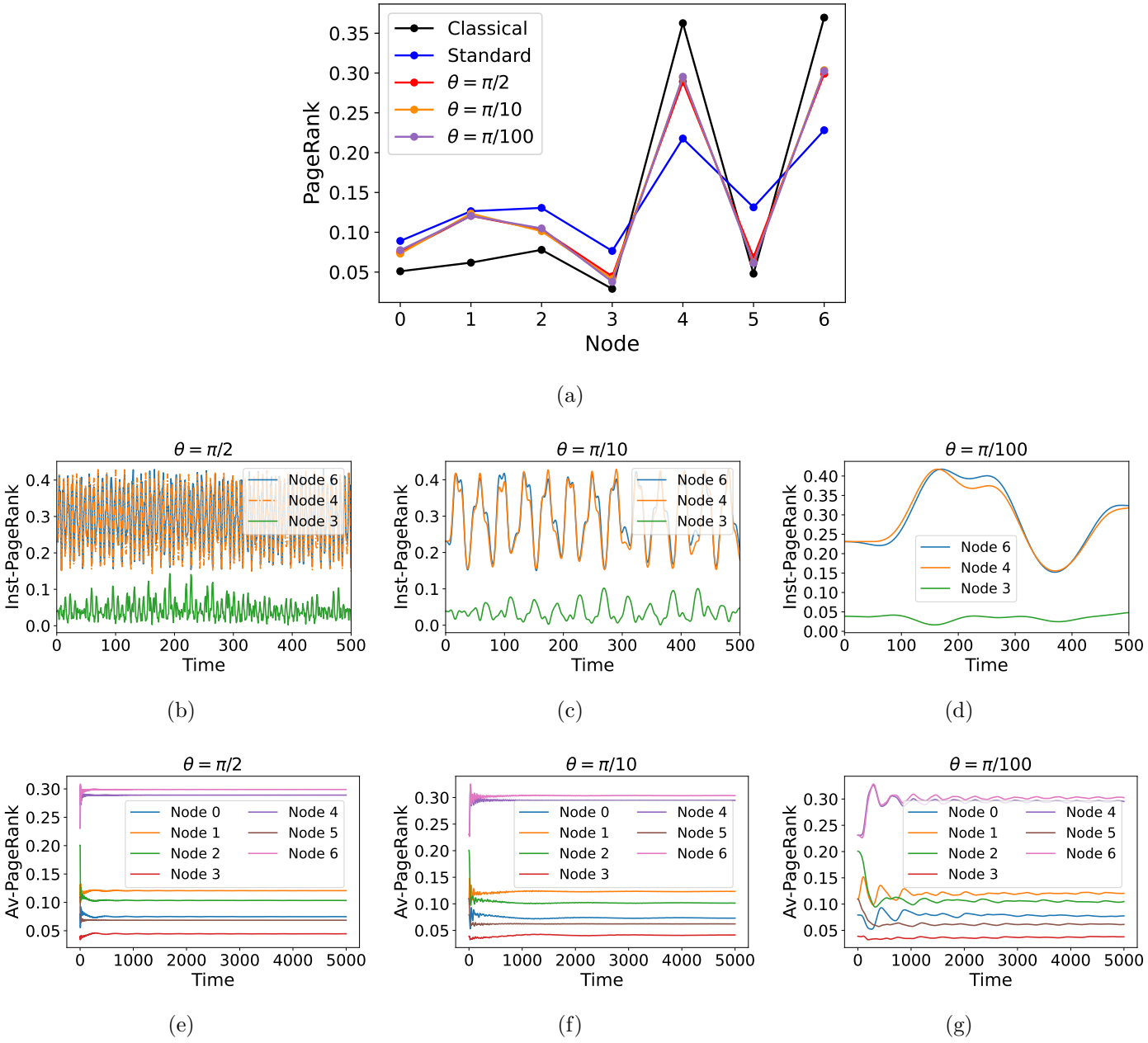


Figure 5.2: *(a) Time-averaged quantum PageRank distributions for the equal-phases scheme with $\theta = \pi/2$, $\pi/10$, and $\pi/100$ for the small generic graph with seven nodes. They are compared with the classical PageRank and the standard quantum PageRank distributions. (b)-(d) Instantaneous quantum PageRank scores of nodes 6, 4 and 3 of the small generic graph versus the quantum walk time for the equal-phases scheme with (b) $\theta = \pi/2$, (c) $\theta = \pi/10$, (d) $\theta = \pi/100$. (e)-(g) Time average of the instantaneous quantum PageRank scores for all the nodes of the small generic graph versus the quantum walk time for the equal-phases scheme with (e) $\theta = \pi/2$, (f) $\theta = \pi/10$, (g) $\theta = \pi/100$. It is observed that as θ decreases, the quantum fluctuations get slower and the algorithm takes more time to converge.*

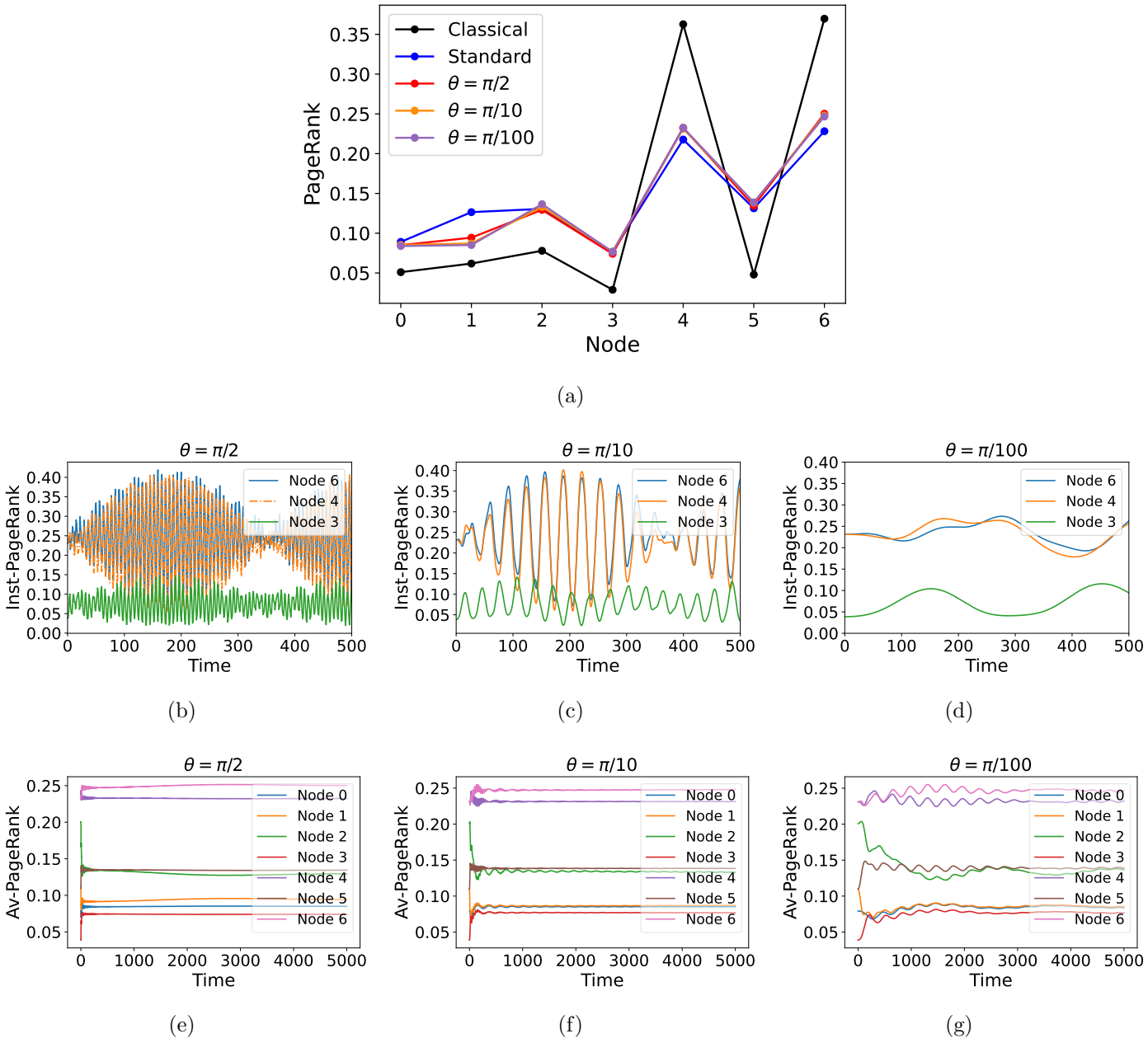


Figure 5.3: *(a) Time-averaged quantum PageRank distributions for the opposite-phases scheme with $\theta = \pi/2$, $\pi/10$, and $\pi/100$ for the small generic graph with seven nodes. They are compared with the classical PageRank and the standard quantum PageRank distributions. (b)-(d) Instantaneous quantum PageRank scores of nodes 6, 4 and 3 of the small generic graph versus the quantum walk time for the opposite-phases scheme with (b) $\theta = \pi/2$, (c) $\theta = \pi/10$, (d) $\theta = \pi/100$. (e)-(g) Time average of the instantaneous quantum PageRank scores for all the nodes of the small generic graph versus the quantum walk time for the opposite-phases scheme with (e) $\theta = \pi/2$, (f) $\theta = \pi/10$, (g) $\theta = \pi/100$. It is observed that as θ decreases, the quantum fluctuations get slower and the algorithm takes more time to converge.*

Since the APR schemes get closer to the classical distribution compared to the standard one, we can use the fidelity with respect to the classical distribution to quantitatively measure the effect of the APR. Despite originating from quantum algorithms, the quantum PageRank distributions behave as classical probability distributions, so we use the classical fidelity [32] defined as:

$$f(I_1, I_2) := \sum_{i=1}^{N} \sqrt{I_1(P_i) I_2(P_i)}. \tag{5.2}$$

The results for the three values of θ analyzed with the three APR schemes are summarized in Table 5.1. From this table and Figures 5.1(b), 5.2(a), and 5.3(a), we can see that the major effect of the APR is achieved with $\theta = \pi/2$. Then, we can use this value for the three APR schemes, allowing the algorithms to converge relatively quickly, in a manner similar to the standard quantum algorithm.

Table 5.1: *Fidelity with the classical PageRank distribution for the standard quantum algorithm and the three APR schemes with $\theta = \pi/2$, $\pi/10$ and $\pi/100$, for the small generic graph.*

Quantum case	$\theta = \pi/2$	$\theta = \pi/10$	$\theta = \pi/100$
Standard	0.9546		
Equal phases	0.9874	0.9886	0.9887
Opposite phases	0.9638	0.9622	0.9621
Alternate phases	0.9870	0.9940	0.9941

Once we have chosen a concrete value of θ, we can compare the results for the three APR schemes with $\theta = \pi/2$. The averaged quantum PageRank distributions are shown in Figure 5.4. For this graph, the equal-phases and alternate-phases cases gets qualitatively closer to the classical distribution, whereas the opposite-phases case resembles more the standard quantum distribution. As we will see later, this behavior depends on the type of graph, and is different for complex networks.

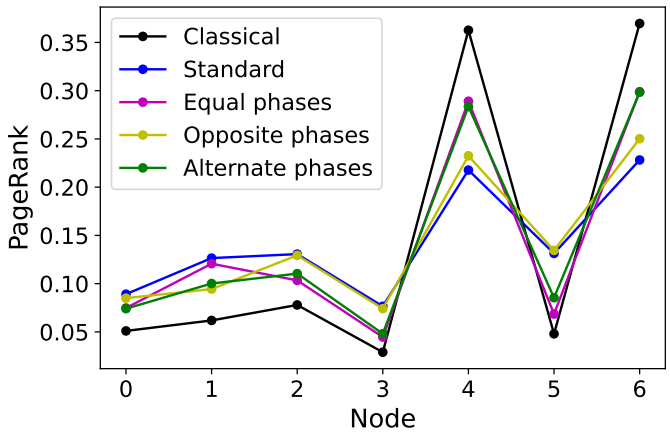


Figure 5.4: *Time-averaged quantum PageRank distributions for the three APR schemes with $\theta = \pi/2$ for the small generic graph with seven nodes. They are compared with the classical PageRank and the standard quantum PageRank distributions.*

If we consider the rankings of nodes, which are summarized in Table 5.2, we observe that the classical ranking is violated in all the quantum cases. The node that is shifted in the ranking varies between APR schemes. However, it is worth mentioning that all the algorithms detect the two most important nodes as well as the least important node properly.

Table 5.2: *Ranking of the nodes of the small generic graph for the classical algorithm, the standard quantum algorithm, and the three APR schemes with $\theta = \pi/2$. The shifted node with respect to the classical ranking is marked in red.*

Classical	Standard	Equal phases	Opposite phases	Alternate phases
6	6	6	6	6
4	4	4	4	4
2	5	1	5	2
1	2	2	2	1
0	1	0	1	5
5	0	5	0	0
3	3	3	3	3

Another important result of the introduction of the APR extension is that the standard deviation of the fluctuation of the instantaneous quantum PageRank is reduced. In Table 5.3 we summarize the standard deviations for all the nodes in the three APR schemes with $\theta = \pi/2$. The most significant reduction occurs in the alternate-phases case, although in general the three schemes show a tendency to decrease the standard deviation. This allows to better distinguish between different nodes, improving the performance of the quantum algorithm.

Table 5.3: *Standard deviations for the instantaneous quantum PageRank scores of the nodes of the small generic graph using the standard algorithm, and the three APR schemes with $\theta = \pi/2$.*

Node	Standard	Equal phases	Opposite phases	Alternate phases
0	0.046	0.044	0.030	0.029
1	0.071	0.071	0.033	0.044
2	0.063	0.053	0.055	0.046
3	0.039	0.026	0.029	0.016
4	0.105	0.081	0.088	0.072
5	0.070	0.034	0.064	0.034
6	0.102	0.078	0.084	0.068

5.3. Application to Complex Scale-Free Graphs

So far, we have seen the effect of the APR schemes in a small network. Nevertheless, despite the interesting results as the violation of the classical ranking, this graph is actually a toy model which does not provides relevant information about the nodes. In this section, we want to study the behavior of the new algorithms in complex networks, from which we can extract useful information about the structure of the nodes, and where the standard quantum algorithm has shown a good performance [32]. In this context, we are going to use scale-free networks, which not only are good models of the World Wide Web [182], but also have a wide range of applications such as in neuronal connections [183], metabolomics [184,185], and finances [186]. These kinds of graphs are characterized by a power-law distribution in the connectivity of nodes [164], and it has been observed that the classical and quantum PageRank algorithms also show a power-law behavior [32, 187].

Scale-free networks are formed by continuously adding nodes, which are connected to the existing

nodes with a probability that is proportional to the indegree and outdegree of the existing nodes. Thus, it is expected that the first nodes added to the model have the largest number of nodes linking to them, which turns out in a higher ranking. These nodes will constitute the main hubs of the network, where the term hub refers to a node with a relatively large number of links [32]. In this thesis, we create random directed scale-free graphs using the Python library NetworkX [10] with the default parameters. This model considers networks with multiple edges and loops [188]. In order to be in concordance with the PageRank definition in (3.57), we have eliminated duplicated edges, but there would not be a major difference if we considered it. At the same time, no major difference in the results has been found whether we eliminate the loops or not, and we have decided to keep them.

5.3.1. PageRank distributions

It has been observed that with the classical PageRank the least important nodes of scale-free graphs are quite degenerate. However, the standard quantum PageRank could break this degeneracy, so that it could unveil the structure of the graph in more detail [32]. To study the effect of the APR schemes with this kind of graphs, we have constructed a random scale-free graph with $N = 32$ nodes. The resulting network is shown in Figure 5.5(a), whereas the classical PageRank and all the quantum PageRank distributions are shown in the histogram of Figure 5.5(b). We can effectively see that the classical distribution has a degeneracy of the least important nodes, broken in the quantum distribution. When we look at the new algorithms with APR we find intriguing properties. Whereas the equal-phases case shows a pattern similar to the standard quantum algorithm, the opposite-phases and alternate-phases cases show a partial restoring of the degeneration of the least important nodes, resembling the classical distribution. This can be seen explicitly, for example, in nodes $19-25$, where the standard and equal-phases algorithms find differences in importance not present in the other distributions. Regarding the main hubs, as expected, they correspond to the first three nodes, and are detected properly by all the algorithms. Moreover, the classical relative importance of the three main hubs is kept by all the quantum algorithms, although this may not be the case for other graphs of the same class.

Let us look deeper at the structure of the graph. According to the classical PageRank definition in (3.57), those nodes without links pointing to them would have a null PageRank score. Due to the patches introduced to built the Google matrix, these nodes have a small non-null PageRank, which is the same for all of them. These nodes are the outer (blue) nodes of Figure 5.5(a). In the histogram we can effectively see that all of them are degenerate in the classical distribution. Node 13 is a secondary hub with two nodes linking to it, and since one of the linking nodes is a main hub (node 2), its PageRank score is high enough to distinguish it. However, nodes 4, 15, and 18, which have a node linking to them, have a small classical PageRank score that is very similar to the least important nodes. This means that the classical PageRank is not able to identify all the secondary hubs properly. In the case of the standard quantum algorithm, it lifts the importance of these secondary hubs. Nevertheless, it breaks the degeneracy of the least important nodes in a manner that some of these residual nodes overshadow the secondary hubs. See, for example, how node 8, which should be residual, has a greater importance than nodes 4 and 15. The fact that nodes which are equal from the point of view of (3.57) are different in the quantum PageRank can make us think that the quantum algorithm is sensitive not only to the nodes linking to a concrete node, but also to the nodes it points to. This fact may be related to the ghost links that appear in directed graphs for unpaired directed edges under the action of the Szegedy quantum walk, as discussed in

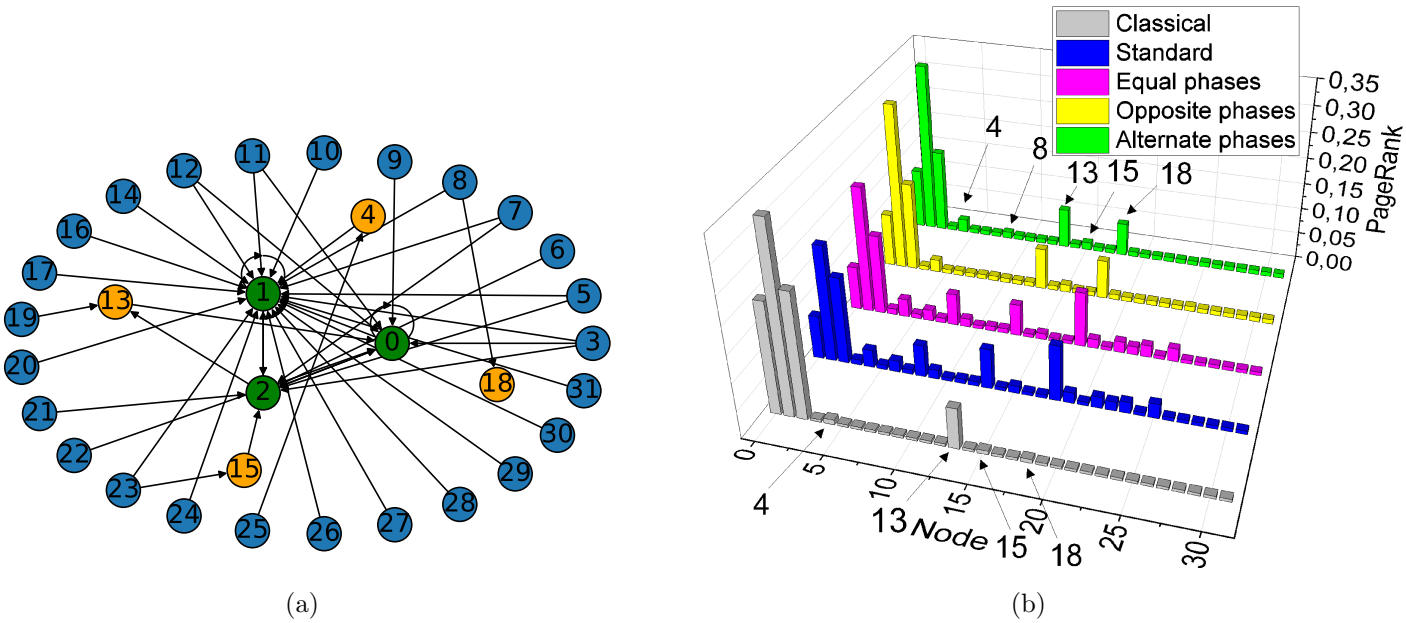


Figure 5.5: *(a) Scale-free network with $N = 32$ nodes. The inner (green) nodes correspond to the main hubs. The middle (orange) nodes correspond to secondary hubs. The outer (blue) nodes correspond to residual nodes without links pointing to them. (b) PageRank distributions of the scale-free network in (a). The classical distribution is compared with all the quantum distributions, using $\theta = \pi/2$ in the three APR schemes. There is a partial restoration of the degeneracy of the least important nodes for the opposite-phases and alternate-phases schemes.*

Section 3.5.

When we add the APR schemes to the quantum algorithm, we do not find a significant difference in the equal-phases case. However, in the opposite-phases and alternate-phases algorithms the residuals nodes are again degenerate in majority. Note that node 8 has a slightly greater importance in the alternate-phases algorithm than the other residual nodes, but it is still less important than the truly secondary hubs. This means that in these APR schemes, the quantum algorithms are practically only sensitive to the indegree distribution of the nodes, as the classical one. Moreover, these two schemes maintain the quantum property of highlighting secondary hubs with respect to the classical algorithm, as can be seen in nodes 4, 15, and 18. This makes these algorithms a valuable tool for ranking nodes in a scale-free network, because they improve the classical deficiencies while solving the problematic quantum sensitivity to the outdegree distribution.

As happens with the small graph, the standard deviation of the quantum PageRank scores can be decreased using certain APR schemes. In Figure 5.6 the standard deviations for all the nodes are shown for the four quantum algorithms. While the equal-phases scheme seems to have standard deviations similar to the standard case, the opposite-phases and alternate-phases schemes show a clear improvement, decreasing the standard deviations. Recall that these last two schemes are those that have a partial restoration of the degeneration. This highlights the valuable importance of the opposite-phases and alternate-phases schemes as APR alternatives to the standard quantum algorithm.

Note that in the small generic graph the APR schemes that most resembled the classical dis-

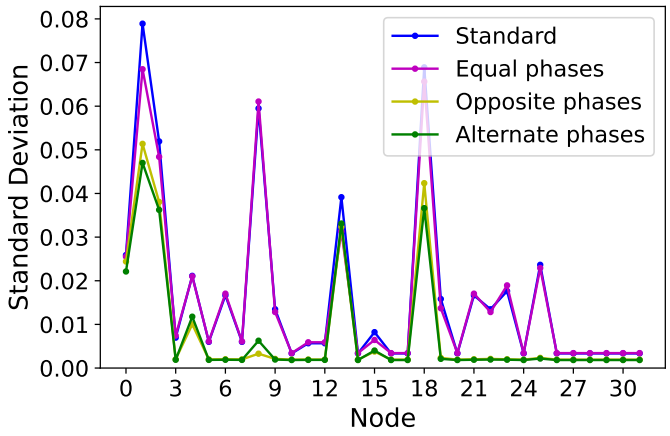


Figure 5.6: *Standard deviations for the quantum PageRank distributions of the random scale-free graph with $N = 32$ nodes of Figure 5.5(a). $\theta = \pi/2$ has been used for the three APR schemes. The standard deviations decrease for the opposite-phases and alternate-phases schemes.*

tribution were the equal-phases and alternate-phases cases. However, in this case, the equal-phases scheme is more similar to the standard quantum case, and the opposite-phases scheme is more similar to the classical distribution. This suggests that the behavior of the APR schemes depends on the kind of graph. As we will see in following results, the characteristics of the APR schemes found for this instance of scale-free graph hold for different instances within this class.

5.3.2. Power law of the PageRank distributions

Since scale-free networks follow a power-law distribution in the connectivity of the nodes, they also have a similar behavior in the PageRank distribution [32, 187]. Then, the PageRank score can be expressed as[1]:

$$I(i) \sim (i+1)^{-\beta}, \tag{5.3}$$

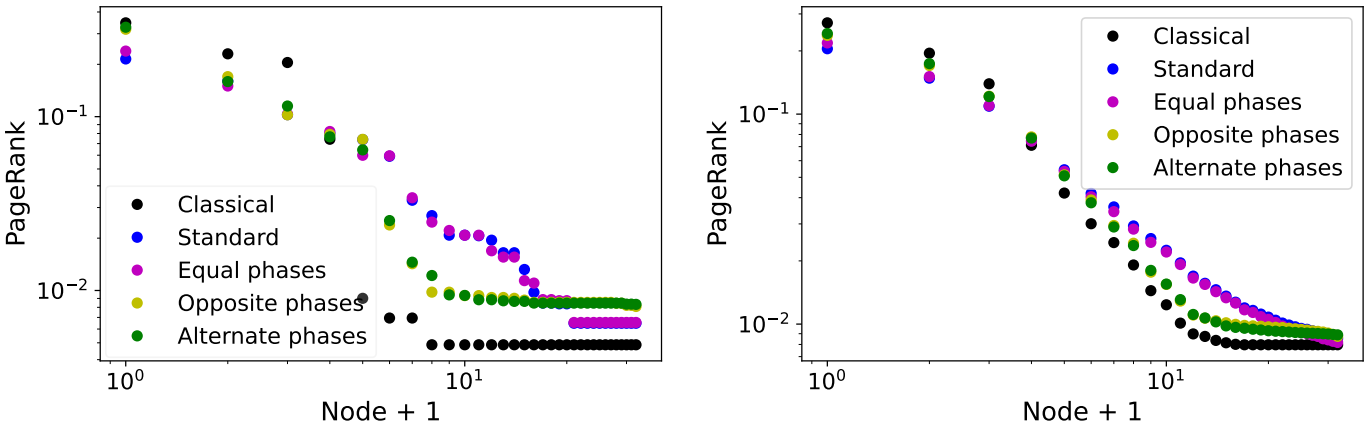
where i is the index of the node after sorting them by importance, and β is a constant coefficient. Taking logarithms on both sides of (5.3), we obtain:

$$\log I(i) \sim -\beta \log (i+1). \tag{5.4}$$

Then, plotting the sorted nodes in logarithmic scale, we expect to see a linear behavior. This plot is shown in Figure 5.7(a) for the graph with $N = 32$ nodes used previously. We can see that the standard and equal-phases quantum algorithms show a smoother behavior due to the degeneracy breaking of the least important nodes. The classical, opposite-phases and alternate-phases algorithms have a big degeneration in the least important nodes, so that the decay in the PageRank score before the degenerate region is more abrupt. Moreover, these last two APR schemes were able to highlight truly secondary hubs, and the least important nodes have a higher PageRank score than in the classical distribution, so the distribution is also smoother with respect to the classical one.

To ensure that this behavior is not particular for this concrete graph, but for the majority of the scale-free graphs with $N = 32$ nodes, we have averaged the sorted PageRank distributions from

[1]The original power law in the literature lacks the factor +1 since the nodes were labeled from 1 to N [32]. We introduce it to avoid issues with node 0 and be consistent with our convention.



(a) (b)

Figure 5.7: *(a) Logarithmic plot of the PageRank scores vs the node index (after sorting) for the random scale-free graph with $N = 32$ nodes of Figure 5.5(a). (b) Averaged logarithmic plot of the PageRank scores vs the node index (after sorting) for an ensemble of 50 random scale-free graphs with $N = 32$ nodes. The classical distribution is compared with all the quantum distributions, using $\theta = \pi/2$ in the three APR schemes. There is a partial restoration of the degeneracy of the least important nodes for the opposite-phases and alternate-phases schemes.*

an ensemble of 50 random scale-free graphs. The averaged results are shown in Figure 5.7(b). This confirms that the discussion above is valid for the scale-free graphs class with 32 nodes, rather than for a concrete graph. Furthermore, the above discussion holds for scale-free graphs with a higher number of nodes. Similar results can be found in our original paper for graphs with 64 and 128 nodes [1], and the results for graphs with 256 nodes are shown in Figure 5.8(a).

Table 5.4: *Values of the coefficient β in the power-law distribution of the PageRank for all the algorithms using ensembles of 50 scale-free graphs with $N = 32$, 64, 128 and 256 nodes. $\theta = \pi/2$ has been used in the three APR schemes.*

Algorithm	$N = 32$	$N = 64$	$N = 128$	$N = 256$
Classical	1.53 ± 0.10	1.55 ± 0.06	1.38 ± 0.02	1.27 ± 0.01
Standard	1.04 ± 0.03	1.02 ± 0.02	0.90 ± 0.01	0.83 ± 0.01
Equal phases	1.06 ± 0.03	1.02 ± 0.02	0.90 ± 0.01	0.83 ± 0.01
Opposite phases	1.34 ± 0.10	1.38 ± 0.06	1.28 ± 0.02	1.23 ± 0.01
Alternate phases	1.35 ± 0.09	1.35 ± 0.05	1.25 ± 0.02	1.20 ± 0.01

The coefficient β in (5.4) is a measurement of the smoothness of the power law distribution. We can obtain the coefficient β for each algorithm after a linear fit of the logarithmic plots. In Figure 5.8 we show the power law distribution for the ensemble of 50 graphs with $N = 256$ nodes, as well as the linear fit for each algorithm. In the classical algorithm, and in the opposite-phases and alternate-phases schemes, the fit has been made only with the non-degenerate nodes, since the degenerate region has a constant distribution. The separation between both regions is shown with a vertical line. On the one hand, the standard quantum algorithm and the equal-phases case have a smoother behavior with respect to the other algorithms, which is characterized by a smaller value of β. Moreover, the power law distribution extends to the least important nodes since they are not

degenerate. This can be related with the previous discussion that these algorithms are more sensitive to both the indegree and outdegree distributions of the nodes, thus better capturing the power law of the vertex connectivity. On other hand, as expected, the opposite-phases and alternate-phases distributions are smoother than the classical one, having a slightly smaller value of β. Finally, we have also done the linear fit for ensembles of 50 graphs with 32, 64, and 128 nodes. The values of β for each case are summarized in Table 5.4. In the four cases we see a similar qualitative behavior of β between the different algorithms, although the absolute values can change slightly with the number of nodes.

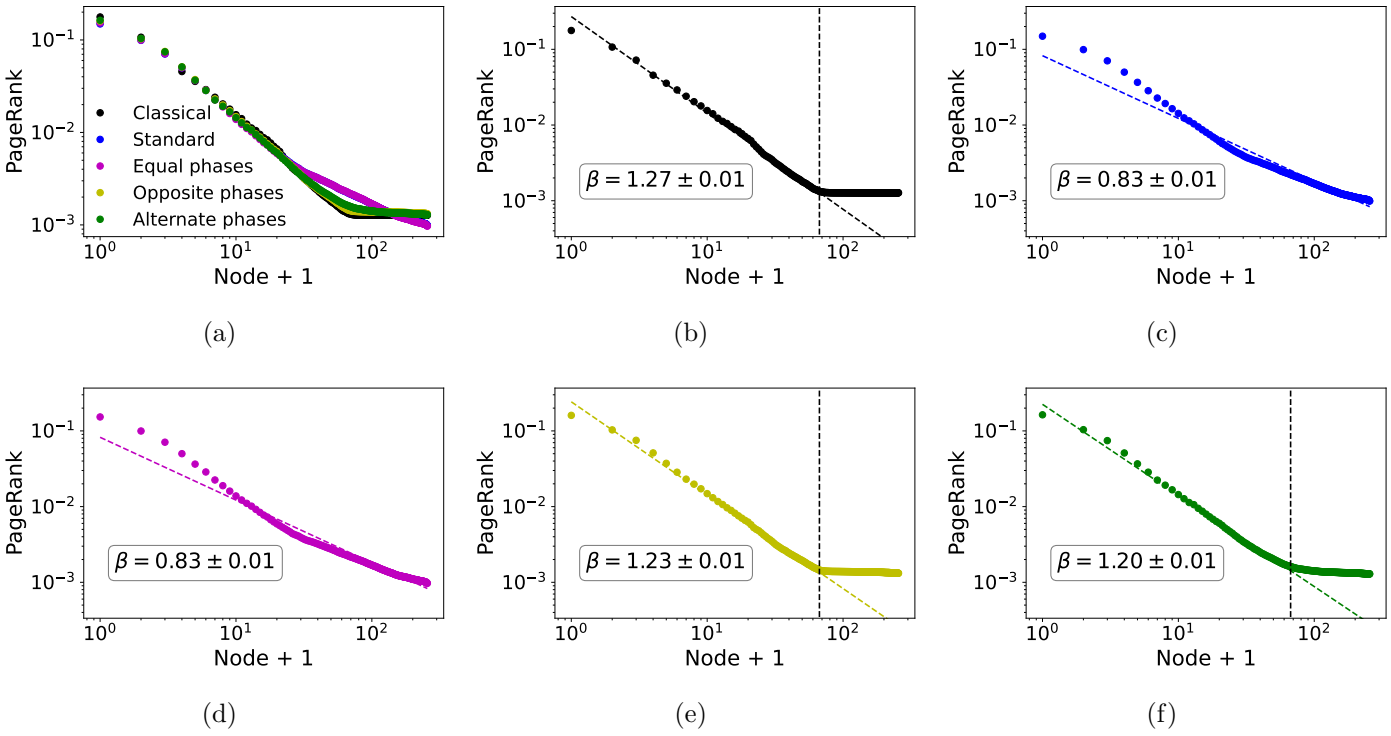


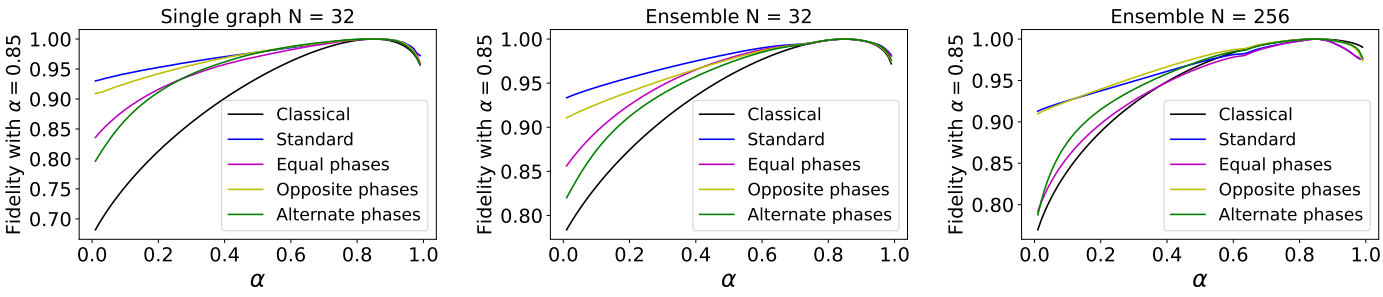
Figure 5.8: *(a) Averaged logarithmic plot of the PageRank scores vs the node index (after sorting) for an ensemble of 50 random scale-free graphs with $N = 256$ nodes. (b)-(f) Linear fit in logarithmic scale for (b) the classical algorithm, (c) the standard quantum algorithm, (d) the equal-phases algorithm, (e) the opposite-phases algorithm, and (f) the alternate-phases algorithm. In (b), (e), and (f), only the non-degenerate region has been take into account. $\theta = \pi/2$ has been used in the three APR schemes. The standard quantum algorithm and the equal-phases algorithm follow a smoother power law.*

5.4. Stability of the Quantum PageRank Algorithms

Recall that the Google matrix G is formed by a transition matrix E, related to the link structure of the graph, and a component corresponding to a random hopping, so that

$$G := \alpha E + \frac{(1-\alpha)}{N}\mathbf{1}. \tag{5.5}$$

We have fixed the damping parameter α in (5.5) to $\alpha = 0.85$ since that is the value that showed an optimal performance in the classical algorithm. It is known that the classical algorithm is very



(a) (b) (c)

Figure 5.9: *(a) Fidelity of the PageRank distributions vs the damping parameter α, with respect to the distribution with $\alpha = 0.85$, for the random scale-free graph with $N = 32$ nodes of Figure 5.5(a). (b)-(c) Averaged fidelity of the PageRank distributions vs the damping parameter α, with respect to the distribution with $\alpha = 0.85$, for an ensemble of 50 random scale-free graphs with (b) $N = 32$ nodes, and (c) $N = 256$ nodes. The classical distribution is compared with all the quantum distributions, using $\theta = \pi/2$ in the three APR schemes. All the quantum algorithms are more stable than the classical one, with the standard and the opposite-phases algorithms being the most stable.*

sensitive to the value of this parameter [189], and the standard quantum PageRank algorithm is more stable for scale-free networks [32]. The aim of this section is to study how the introduction of the APR schemes affects the stability of the quantum PageRank.

We start with the random scale-free graph with $N = 32$ nodes of Figure 5.5(a). We can measure the similarity between two distributions obtained with different damping parameter α using the fidelity defined in (5.2). We analyze the behavior of this fidelity between the distribution obtained for a value of $\alpha \in [0.1, 0.99]$ and the usual distribution with $\alpha = 0.85$. These fidelities are represented in Figure 5.9(a) for the classical and all the quantum algorithms, with $\theta = \pi/2$ in the APR schemes. We observe that the fidelity for the classical algorithm decreases very quickly with the parameter α, reaching a value under 0.70. However, as it was also shown in the original work [32], the standard quantum algorithm is by far more stable, with a minimum fidelity of approximately 0.93.

Regarding the quantum algorithms with APR, we find that they are also more stable than the classical. On the one hand, both the equal-phases and alternate-phases cases show a similar behavior (the former slightly better), which is intermediate between the standard quantum and the classical algorithms. On the other hand, the opposite-phases algorithm seems to be approximately as stable as the standard quantum algorithm. We find this result surprising, since the equal-phases algorithm is the one that shows a distribution of PageRank scores similar to the standard quantum case, whereas the opposite-phases algorithm restores the degeneracy of the nodes in a pattern similar to the alternate-phases case. Then, the opposite-phases algorithm seems to be very promising for scale-free graphs, since it resembles the classical PageRank distribution while highlighting truly secondary hubs, and maintains the quantum stability. To ensure that this intriguing behavior is not particular for this concrete network, but is a property of the scale-free networks class, we have averaged the stability results for 50 random scale-free graphs with $N = 32$ nodes in Figure 5.9(b), finding results that fit in the discussion above. Furthermore, to ensure that this behavior holds for bigger scale-free networks, we have averaged the results for 50 scale-free graphs with $N = 256$ nodes in Figure 5.9(c), effectively finding a similar behavior. It is worth noting that for the networks with 256 nodes there is a region where the opposite-phases algorithm outperforms the standard quantum algorithm. Indeed,

we have found for a lot of graphs in this class that the curve with the opposite-phases scheme is slightly above the curve of the standard quantum case for all the values of α. Similar results for graphs with 64 and 128 nodes can be found in our original paper [1].

Finally, we shall see what happens with the fidelity not only for $\alpha = 0.85$, but for any pair (α, α'), with $\alpha, \alpha' \in [0.1, 0.99]$. We show the averaged results for the ensemble of graphs with $N = 256$ nodes in Figure 5.10 using heatmaps. The standard quantum algorithm shows a good stability region that extends to all the values of α, having a minimal fidelity of 0.91 in the extreme pairs. As expected, the opposite-phases algorithm has a similar pattern, but the fidelity drops slightly at the extremal pairs of α, reaching a minimal fidelity of 0.87. The classical algorithm is the least stable, with the fidelity falling quickly when we move out of the central region. The minimal value of fidelity reached is 0.70. Last, the equal-phases and alternate-phases algorithms show an intermediate behavior between the classical and the standard quantum algorithms. The minimal values of fidelity are 0.79 and 0.73, respectively. These results seem to reinforce the previous discussion about the stability of the PageRank algorithm.

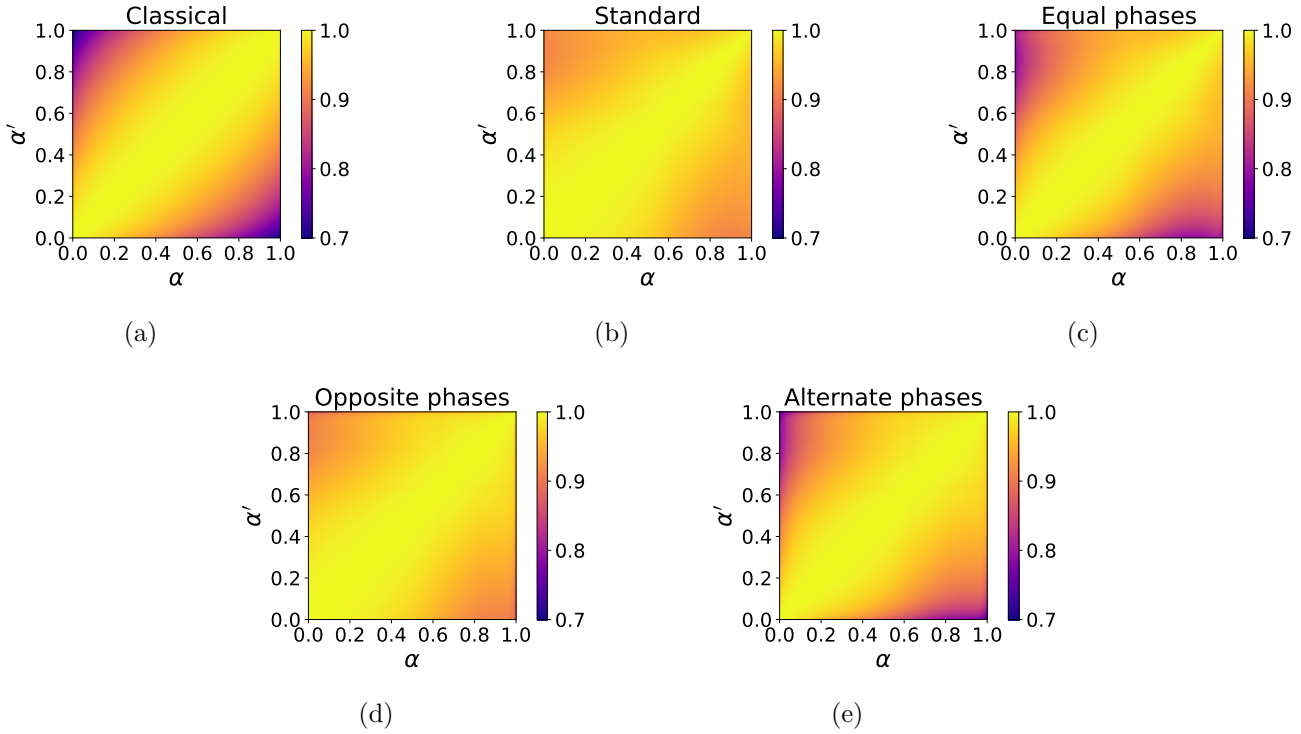


Figure 5.10: *Averaged fidelity of the PageRank distributions for all pairs (α, α'), for an ensemble of 50 random scale-free graphs with $N = 256$ nodes, using (a) the classical algorithm, (b) the standard quantum algorithm, (c) the equal-phases algorithm, (d) the opposite-phases algorithm, and (e) the alternate-phases algorithm. $\theta = \pi/2$ has been used in the three APR schemes. All the quantum algorithms are more stable than the classical one, with the standard and the opposite-phases algorithms being the most stable.*

5.5. Summary of Results and Conclusions

- We have applied the extension of the Szegedy quantum walk by means of global arbitrary phase rotations (APR) to the quantum PageRank algorithm, introducing two degrees of freedom. However, we have defined three simple schemes, called equal-phases, opposite-phases, and alternate-phases schemes, with only a single parameter θ.

- We have applied the new quantum PageRank algorithms with APR to a small generic graph with seven nodes, comparing the results with those described in the literature. We have found that a decrease in the value of θ reduces the standard deviation of the instantaneous quantum PageRank scores, allowing to better distinguish between nodes. However, the oscillation of the instantaneous PageRank gets slower, so that the time-averaged quantum PageRank distribution needs more time to converge. This means that we cannot reduce the phase θ arbitrarily. We have chosen a value of $\theta = \pi/2$ as a value where the APR schemes have a great effect on the time-averaged PageRank while the convergence time is short. With this value, the equal-phases and alternate-phases distributions more resemble the classical one, whereas the opposite-phases distribution resembles the standard quantum one.

- We have studied the time-averaged quantum PageRank with APR on complex scale-free networks, which apart of being good models of the World Wide Web, have plenty of applications. It was known that the standard quantum PageRank algorithm was able to highlight secondary hubs of the networks, whose PageRank scores were suppressed in the classical distribution. Moreover, the quantum algorithm breaks the degeneracy of the residual nodes, which should be degenerate as they do not have nodes linking to them. This could yield a problem, as those residual nodes can overshadow the actual secondary hubs. The opposite-phases and alternate-phases schemes overcome this problem, restoring the degeneration and making them residual. Somehow, the APR reduces the effect of the ghost links that appear in the Szegedy quantum walk. Thus, these two schemes have a distribution that resembles the classical one but highlighting truly secondary hubs. However, the equal-phases scheme yields a distribution very similar to the standard quantum one. Regarding the standard deviation of the quantum PageRank distribution, the opposite-phases and alternate-phases schemes can decrease it, but the equal-phases scheme does not. Since the effect of the different schemes is different from what was found in the small generic graph, we conclude that the effect may depend on the kind of network. In our original paper [1] there are some results for a particular instance of Erdős-Rényi graphs [190], observing a different behavior of some APR schemes with respect to the scale-free graphs.

- Scale-free networks follow a power-law distribution in the connectivity of the nodes. It was known that the classical and quantum PageRank distributions also have a power-law behavior, being smoother in the case of the quantum algorithm since it breaks the degeneracy of the residual nodes. Comparing all of our algorithms, we have found that the standard quantum algorithm, and that with the equal-phases scheme have the smoothest distributions, and the power law extends to the residual nodes. The fact that the residual nodes are not degenerate and also follow a power law may indicate that these two quantum algorithms are sensitive to the outdegree distribution of the nodes, since they do not have any node linking to them. Thus, they inherit the power law of the connectivity characteristic of scale-free networks. We

have also seen that the opposite-phases and alternate-phases schemes have a slightly smoother distribution than the classical algorithm.

- We have studied the stability of the PageRank algorithm with respect to the damping parameter α on scale-free networks. In the literature it was shown that the classical algorithm was quite unstable, whereas the quantum algorithm improved its stability considerably. In the case of the APR schemes, we expected the equal-phases scheme to be the most stable, since its PageRank distribution is very similar to that of the standard quantum algorithm. Surprisingly, the opposite-phases scheme is the most stable, being comparable to the standard quantum case. The equal-phases and alternate-phases schemes have a similar intermediate stability, despite the fact that their PageRank distributions are rather different.

- Taking all the results together, the fact that the algorithm with the opposite-phases scheme is able to highlight the secondary nodes that the classical cannot, keeps degenerate the residual nodes, reduces the standard deviation of the instantaneous quantum PageRank scores, and also has a good stability similar to the original quantum algorithm, makes this new algorithm a valuable tool as an alternative to both the classical and the standard quantum PageRank for scale-free networks.

- Regarding the walk complexity when considering the APR schemes, further research is necessary. Nevertheless, in the simulations performed in this thesis for $\theta = \pi/2$, we have not observed a significant variation of the number of steps needed to converge when increasing the size of the networks. Thus, *a priori* we can state that the walk complexity is approximately constant, the same as for the standard quantum PageRank.

- A recent study using APR phases other than $\pi/2$, other APR schemes, or even introducing more phase rotations to the PageRank algorithm has been published in the literature [191]. In the future, it would also be interesting to apply the APR PageRank schemes to other kinds of complex networks, such as hierarchical networks, or to other algorithms of interest, such as optimization, or machine learning.

Results II

The Semiclassical Framework

Chapter 6

Semiclassical Walks

The second part of results of this thesis is devoted to the introduction of a semiclassical framework in the context of quantum walks [2]. There are different forms of combining classical and quantum dynamics. One comes from the quantum stochastic walk [192], a parameterized walk driven by non-unitary evolution that interpolates between the quantum and the classical walk. Other idea, which is the actual precursor of our new algorithms, comes from the measurement-induced quantum walk [193], where the position of the walker is measured at regular intervals of a continuous-time unitary quantum walk. Although its dynamics is interesting, its range of application is quite limited due to the nature of the underlying walk. For this reason, our aim is to find an analogue algorithm for discrete-time quantum walks, as the Szegedy quantum walk [45], which can quantize arbitrary Markov chains and has plenty of applications, and could give rise to novel algorithms in the future.

We denote as semiclassical walks to this new kind of algorithms, since can be understood as classical walks encoding quantum dynamics. As we will see along this chapter, the implementation of such framework for discrete-time quantum walks is not straightforward. It is necessary an additional control from a classical computer to restart the system after each measurement, so that we can obtain a truly Markovian process. In this sense, reset operations and classically-controlled gates may be necessary, beyond the usual quantum gates and measurements.

This initial chapter of this second part of results presents the semiclassical walks from discrete-time quantum walks, and the implementation based on the Szegedy walk [2]. In Chapter 7 we explore further an application in the context of the quantum SearchRank algorithm [4], giving rise to an improved algorithm which we use in Chapter 8 for fraud detection in blockchain [6]. Furthermore, this results part also connects with Chapter 9, where we show the efficient classical simulation of the semiclassical Szegedy walks [3].

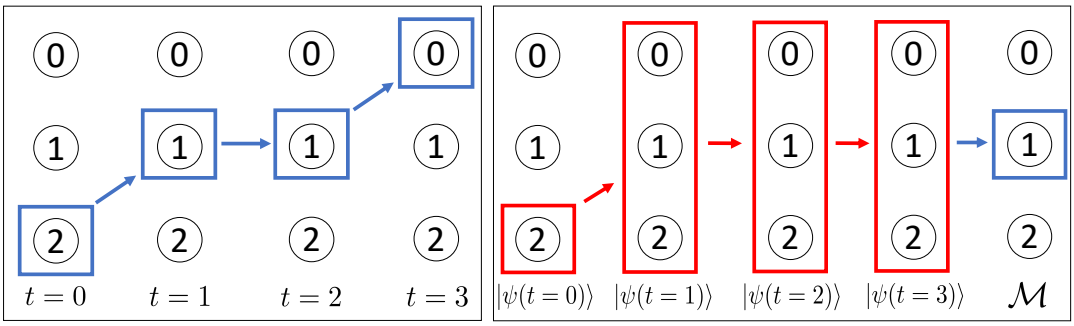
With regard to this particular chapter, it is structured as follows. In Section 6.1 we review the formulation of classical and quantum walks, to later introduce the semiclassical walks in discrete time. In Section 6.2 we focus on the semiclassical walks built from the Szegedy quantum walk. In Section 6.3 we solve analytically the problem for 1D cycles and show results for some examples. In Section 6.4 we simulate the semiclassical walks on a generic weighted graph, showing how this approach can break the symmetry of the graph. In Section 6.5 we compare our results with the previous approach in continuous time. In section 6.6 we show experimental results of semiclassical walks in a real quantum computer. Finally, we summarize and conclude in Section 6.7.

6.1. Semiclassical Walk Formulation

In this section we introduce the formulation of semiclassical walks. To do so, let us briefly review first how classical and quantum walks work. The complete details can be found in Sections 3.1 and 3.2, respectively.

A classical random walk represents a Markov chain in the nodes of a graph. From a stochastic point of view, at each time step the walker is at a particular node of the graph, and can jump to any other node, including the same node, with some probabilities. In Figure 6.1(a) it is shown an example of classical walk on a graph with three nodes. At the initial time $t = 0$ the walker is at node 2. At the first time step, the walker decides stochastically to jump to node 1. At the second time step it remains at node 1, and at the third time step it jumps to node 0. Although the process is stochastic, so that each time the walk is performed the result is different, the probability distribution of the walker being at each node for each time step can be simulated deterministically with the transition matrix G. This is a column-stochastic matrix whose elements G_{ji} are the probabilities of the walker jumping from node i to node j.

A quantum walk is a quantization of a classical walk. In this case, the walker can be in a superposition of the nodes of the graph, so that the position at each time step is represented by a quantum state $|\psi(t)\rangle$, and the system evolves by a unitary operator U. An example of a quantum walk on a graph with three nodes is shown in Figure 6.1(b). The walker starts in a quantum state that represents node 2, it performs three time steps of the quantum walk, being in a superposition of all the nodes, and finally the position is measured, obtaining stochastically node 1 in this example. Running each time the quantum walk on a quantum computer may produces different results due to the probabilistic nature of the measurement. Moreover, the coherence of the system is broken after the measurement, so that the quantum walker is no more in a superposition of the nodes. For this reason, we cannot measure at intermediate steps of the walk and resume it. If we wanted to sample the probability distribution at each time step we would have to perform a different quantum walk for each final time.



(a) (b)

Figure 6.1: *Examples of trajectories followed by a particle in the different types of walks on a graph with three nodes. Blue color represents classical information whereas red color represents quantum information. (a) Classical walk. The particle is in a particular node at each time step, and it jumps to other node with a certain probability. (b) Quantum walk. The walker can be in a superposition of the nodes at each time step, and it is represented by a quantum state. In the end of the quantum walk the position is measured, so that the walker stays at a particular state with a certain probability.*

In this thesis, we wonder what type of phenomenology we obtain if we continue applying the unitary evolution U to the quantum system after the measurement. Due to the coherence breaking, of course we do not obtain the quantum walk that we would have if the measurement would have not be performed. Therefore, a new kind of walk is obtained. In particular, we are interested in the case where the measurement is performed at regular time intervals of the quantum evolution. An example is the measurement-induced quantum walk [193], which applies measurements at regular time intervals of a continuous time quantum walk. We saw in Section 3.8 that the Hilbert space of this walk is simply formed by the computational basis states $|i\rangle$ associated to the nodes of the graph. Therefore, after the evolution with $U(t)$ and the measurement, the system turns into a particular state $|j\rangle$. If the evolution plus measurement cycle is repeated for a fixed quantum evolution time t, it is straightforward to see that the overall process is equivalent to a classical Markov chain whose transition matrix is

$$G_{ji}(t) = |\langle j| U(t) |i\rangle|^2 . \tag{6.1}$$

This is so because the probability of measuring each node after each quantum evolution $U(t)$ only depends on the previous measured state, which is always a computational basis state and is unique for each node.

Our aim is to abstract this idea to the discrete-time quantum walks we deal with in this thesis. In this kind of walks the Hilbert space is not so simple, so that there is an extra register associated to an inner degree of freedom, usually referred as coin register. This register prevents the construction of a Markov chain based on a quantum walk with regular measurements. Let us see an example of measurement-induced quantum walk based on the Hadamard coined quantum walk with the flip-flop shift operator S_f, which we studied in Section 3.3. We perform it on a cycle with three nodes, for two quantum steps. As initial state, we chose node 2 with a coin state pointing to the right:

$$U_c |2\rangle_P |R\rangle_C = \frac{1}{\sqrt{2}} |0\rangle_P |L\rangle_C + \frac{1}{\sqrt{2}} |1\rangle_P |R\rangle_C , \tag{6.2}$$

$$U_c^2 |2\rangle_P |R\rangle_C = \frac{1}{2} |0\rangle_P |R\rangle_C + \frac{1}{2} |1\rangle_P |L\rangle_C - \frac{1}{\sqrt{2}} |2\rangle_P |-\rangle_C , \tag{6.3}$$

where $|-\rangle_C = (|R\rangle_C - |L\rangle_C)/\sqrt{2}$. There is a probability of $1/2$ of measuring again node 2, and $1/4$ of measuring nodes 0 and 1. Suppose that after measuring the position register we obtain node 2 again. The systems turns into the state $|2\rangle_P |-\rangle_C$. We perform again two quantum steps:

$$U_c |2\rangle_P |-\rangle_C = |1\rangle_P |R\rangle_C , \tag{6.4}$$

$$U_c^2 |2\rangle_P |-\rangle_C = \frac{1}{\sqrt{2}} |0\rangle_P |R\rangle_C + \frac{1}{\sqrt{2}} |2\rangle_P |L\rangle_C . \tag{6.5}$$

Now the probabilities are $1/2$ for node 2 and $1/2$ for node 0. Therefore, despite the fact that the quantum evolution has been performed from a state representing the same classical position as at the beginning, the probabilities are not the same due to the different coin state. Therefore, from a classical point of view, where the only information about the state is the position, which in both cases is node 2, the process is not Markovian. The probabilities do not depend uniquely on the current node, but on the whole history of the walker, which in turns depends on the initial coin state.

In order to obtain a Markov chain from a discrete-time quantum walk, we need the coin state before the following quantum evolution to be the same as in the initial state of the walk. To do so, we would have to restart the system, deleting the information of the coin register after the measurement,

and preparing it again in the initial coin state. Thus, we need to generalize the measurement-induced quantum walk with additional operations. This gives rise to our semiclassical walks, which apart from regular measurements, include a restart scheme.

After this example, we can formally introduce the semiclassical framework. Let us first define the time evolution parameters:

Definition 6.1 (Semiclassical time evolution)**.** *The time evolution of a semiclassical walk is determined by two parameters:*

- t_q*: Quantum time*[1]*. It is the number of times we apply the unitary evolution* U *between measurements.*
- t_c*: Classical time. It is the number of times we apply the quantum evolution* U^{t_q}*, measure the position of the walker, and restart the system if needed.*

Each cycle of quantum evolution, measurement and restart is understood as a classical step, so that we end up having a classical trajectory of positions for each classical time step t_c. Since we want the process to be Markovian, each classical position must be associated with a unique quantum state representing it in the Hilbert space, so that when that position is the result of the measurement, the system is restarted in that quantum state before the following quantum evolution. We denote these states as proxy states, since they are the representative agents of the classical nodes:

Definition 6.2 (Proxy states)**.** *Associated to each classical position* i *of a semiclassical walker on a graph is a unique proxy state* $|\xi_i\rangle$*, which represents the classical position prior to quantum evolution.*

If the walker is at node i at each classical time step t_c, then we prepare the state $|\xi_i\rangle$ to perform the quantum evolution U t_q times. An example is shown in Figure 6.2(a) for a graph with three nodes. There, the walker starts at node 2. We prepare the quantum proxy state $|\xi_2\rangle$ and perform the quantum evolution t_q times. After measuring the position we obtain that it is at node 1. This corresponds to the first classical step $t_c = 1$. For the second classical step $t_c = 2$, we prepare the quantum proxy $|\xi_1\rangle$, perform the quantum evolution t_q times, and measure, obtaining that the walker is at node 0. If we treat the quantum evolution as a black box and we only deal with the positions after each measurement, then we can treat the walk as a classical walk as shown in Figure 6.2(b). Thus, we only see that the walker starts at node 2, jumps to node 1, and after that it jumps to node 0.

As in a classical walk, a particular trajectory of the walker is obtained with a certain probability each time we run the algorithm. If we knew the probability of measuring each node after the quantum evolution starting with the proxy state $|\xi_i\rangle$, then we could define a transition matrix, and simulate the walk deterministically in a similar way as equation (3.5). Thus, let us define the semiclassical transition matrix for a semiclassical walk as $G^{(t_q)}$, whose elements are

$$G_{ji}^{(t_q)} := \left|\left|\langle j|U^{t_q}|\xi_i\rangle\right|\right|^2. \tag{6.6}$$

Note that there is a different semiclassical matrix for each value of t_q, that is, there is a different semiclassical walk for each number of times we perform the quantum unitary evolution U between

[1]By quantum time we refer to a parameter that determines the duration of the quantum evolution, rather than a quantum operator.

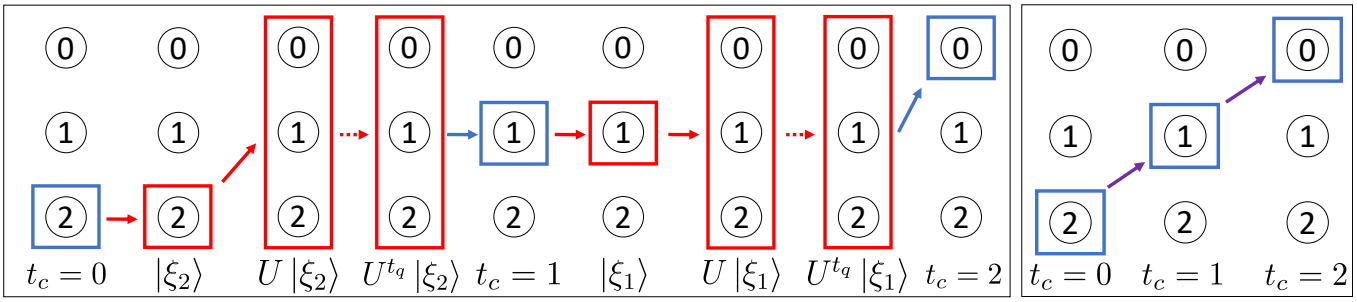


Figure 6.2: *(a) Example of trajectory followed by a particle in a semiclassical walk on a graph with three nodes. Blue color represents classical information whereas red color represents quantum information. At each classical step the walker is in a particular classical position. For each of these steps, the proxy quantum state is prepared and it is performed the quantum evolution t_q times. After that, the position is measured obtaining a new classical position. (b) Representation of the semiclassical trajectory in (a) as a classical walk. Purples arrows indicate that the evolution is affected by both classical and quantum dynamics.*

measurements. Thus, we actually have a family of semiclassical walks. The quantum time t_q is actually a parameter that defines a particular semiclassical walk in the family, whereas the classical time t_c is the actual evolution time, since we would only deal with the particle position at each classical step, and not at intermediate steps of the quantum evolution.

Finally, we have to define how to construct the proxy states $|\xi_i\rangle$. If the Hilbert space where the quantum evolution takes place were the span of the states $|i\rangle$, we could just take these states as proxies, and no restart scheme would be needed. This is what occurs in the measurement-induced quantum walk with a continuous quantum time [193]. For discrete-time coined quantum walks it seems natural to set in the position register the index of the corresponding node, so that the proxy states are defined as:

$$|\xi_i\rangle := |i\rangle_P \otimes |c_i\rangle_C \,, \tag{6.7}$$

where c_i is a coin state that can be different for each node in the network.

We have freedom to chose the initial coin states, the same as in the normal quantum walk. For the Hadamard walk, we could take $|c_i\rangle_C = |R\rangle_C$, $|c_i\rangle_C = |L\rangle_C$, or whatever superposition. Different definitions of the proxy states produce different families of semiclassical walks. As we will see, in the case of the Szegedy walk, the same as in the quantum case there is a natural choice for the initial coin states, there is a natural form of defining the proxy states.

6.2. Semiclassical Szegedy Walk

In this section we introduce the semiclassical framework for the Szegedy quantum walk. From now on, and along this results part, we only consider the standard model, lacking of the phase extensions introduced in Chapter 4. The details of this walk were explained in Section 3.4. Recall that the Hilbert space is $\mathcal{H}_S := \text{span}\{|i\rangle_1 |j\rangle_2 \,,\ i,j = 0,1,...,N-1\} = \mathbb{C}^N \otimes \mathbb{C}^N$, so there are two registers as in a coined quantum walk. Usually, the first register is used to measure the position, and the second one is considered as a coin register. For convenience, let us show again the single-step evolution

operator U_s:

$$U_s := S_w(2\Pi - \mathbb{1}), \quad \Pi := \sum_{i=0}^{N-1} |\psi_i\rangle \langle\psi_i| , \tag{6.8}$$

where

$$|\psi_i\rangle := |i\rangle_2 \otimes |\omega_i\rangle_2 = |i\rangle_1 \otimes \sum_{k=0}^{N-1} \sqrt{G_{ki}}\, |k\rangle_2 . \tag{6.9}$$

These states are the natural choice for setting the initial state of the quantum walk as a linear combination of them. Therefore, to formulate the semiclassical walk from the Szegedy quantum walk, we can use the set of states $|\psi_i\rangle$ as the proxy states in (6.7).

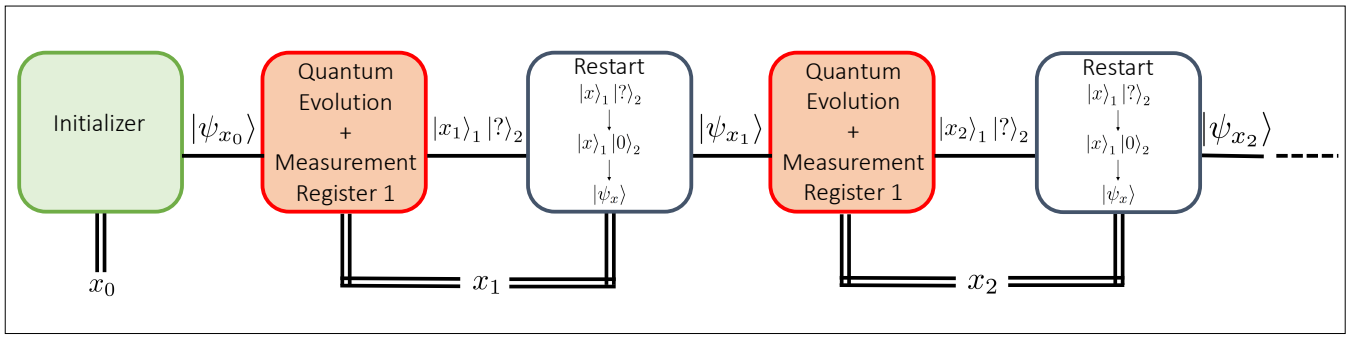
An example of implementation of a semiclassical Szegedy walk is shown in Figure 6.3(a). Let us denote x_{t_c} as the position at classical time step t_c. Thus, we start at node x_0. We prepare the proxy $|\psi_{x_0}\rangle$ and perform the quantum evolution, parameterized by the quantum time t_q. After measuring the first register, it turns into a particular node x_1. The remaining information in the second register, represented by a question mark in the figure, plays no role in the algorithm, so we do not worry about it. Before proceeding to the next step, the system must be restarted. To this end, the second register is reset, so that it is forced non-unitarily to be in $|0\rangle_2$. After that, we use the measured information about the node x_1 to prepare with a suitable unitary evolution the new proxy state, $|\psi_{x_1}\rangle$, completing the system restart. This constitutes a classical step of the semiclassical walk, and the process is repeated the number of classical steps t_c as desired. We call this a semiclassical walk of class I since we are measuring in the first register.

With regard to the unitary preparation after the reset of the second register, there are two options. On the one hand, since the information about the current position is stored in classical bits, we can use classically-controlled gates acting on the second register to prepare the corresponding state $|\omega_i\rangle_2$. On the other hand, we can use quantum gates controlled by the first quantum register, since after the measurement it also contains the information about the position. *A priori* this approach would be less efficient because we would have to apply the gates corresponding to the preparation of all the different proxy states, whereas with classically-controlled gates, only those required by the specific node would be applied. Nevertheless, as discussed in Section 3.6, for certain graphs we could find an efficient implementation of the update operator V, which prepares the proxy state reading the information of the first register:

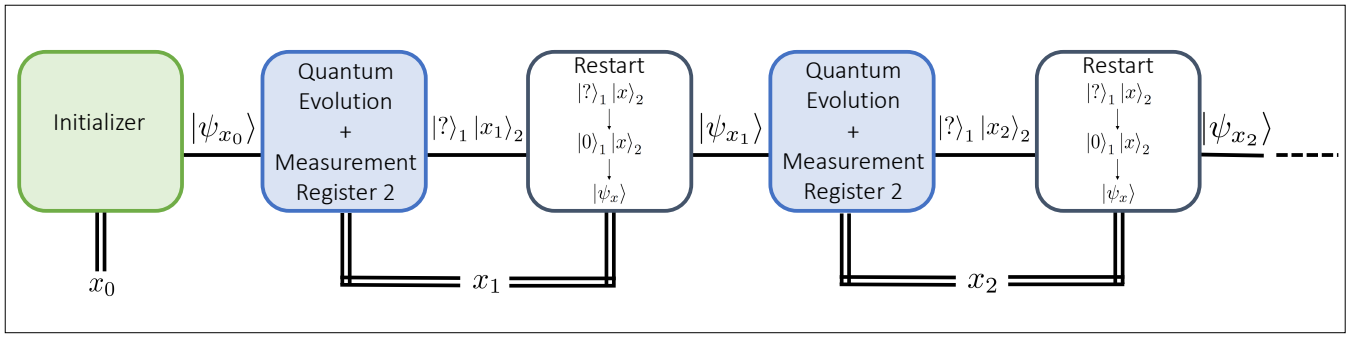
$$V |x\rangle_1 |0\rangle_2 := |x\rangle_1 |\omega_x\rangle_2 = |\psi_x\rangle . \tag{6.10}$$

Although it is common to measure the first register, there are other applications where the second register is measured instead to obtain the information about the nodes. An example is the quantum PageRank algorithm [31, 32]. Thus, we can define a semiclassical walk of class II by measuring in the second register. An implementation is shown in Figure 6.3(b). In this case the information of the nodes are obtained measuring the second register, and the information of the first register is discarded. Nevertheless, the proxies are prepared in the same form as before, according to equation 6.9. Therefore, an additional swap is needed to allocate the position information in the first register before preparing the corresponding proxy state.

Finally, although these semiclassical walks are though to be run on quantum computers, we can also simulate deterministically both classes as classical walks by constructing their semiclassical



(a)



(b)

Figure 6.3: *(a) Scheme of the semiclassical Szegedy walk of class I. The position of the walker at each classical time step is represented by x_{t_c}. At each classical step, the previous classical information is used to restart the system in the corresponding proxy state, the quantum evolution is performed, and finally a new classical position is measured from the first register. (b) Scheme of the semiclassical Szegedy walk of class II. In this case, the second register is measured, and that information is used to prepare the new proxy state.*

transition matrices. Let us use a left-subscript in the semiclassical matrix to denote the class of the walk. Then, the semiclassical matrices are obtained as:

$$ {}_1G_{ji}^{(t_q)} := \left|\left| {}_1\langle j| U_s^{t_q} |\psi_i\rangle \right|\right|^2, \tag{6.11} $$

$$ {}_2G_{ji}^{(t_q)} := \left|\left| {}_2\langle j| U_s^{t_q} |\psi_i\rangle \right|\right|^2. \tag{6.12} $$

The deterministic simulation on a classical computer would imply simulating the quantum walk evolution for each of the N proxy states. Therefore, the time complexity is the one of the Szegedy quantum walk multiplied by N. In the case that we use our optimal simulation algorithm described in Chapter 9, which scales as $\mathcal{O}(N^2)$ for a quantum walk, the semiclassical walk would require a time scaling as $\mathcal{O}(N^3)$.

6.2.1. Limits and equivalences

From the semiclassical matrices we can formulate some theorems about the limits and equivalences of the semiclassical Szegedy walks.

Theorem 6.1 (Classical Limit I)**.** *The classical walk is recovered for the semiclassical Szegedy walk of class I with a single quantum time step, $t_q = 1$, using the single-step operator U_s. Therefore:*

$$ {}_1G^{(1)} = G. \tag{6.13}$$

Proof. We start calculating the quantum state that results of applying the unitary evolution once:

$$ U_s \left|\psi_i\right\rangle = S_w(2\Pi - \mathbb{1}) \left|\psi_i\right\rangle = S_w \left|\psi_i\right\rangle, \tag{6.14}$$

since $\Pi \left|\psi_i\right\rangle = \left|\psi_i\right\rangle$, due to the fact that the space where Π projects is the subspace spanned by the $\left|\psi_i\right\rangle$ states. The swap operator swaps the states between both registers, so

$$ U_s \left|\psi_i\right\rangle = \sum_{k=0}^{N-1} \sqrt{G_{ki}} \left|k\right\rangle_1 \left|i\right\rangle_2. \tag{6.15}$$

To obtain the semiclassical matrix ${}_1G^{(1)}$, we take the inner product with the computational basis of the first register and take the squared modulus:

$$ {}_1\langle j| U_s \left|\psi_i\right\rangle = \sum_{k=0}^{N-1} \left(\delta_{jk} \sqrt{G_{ki}} \left|i\right\rangle_2\right) = \sqrt{G_{ji}} \left|i\right\rangle_2, \tag{6.16}$$

$$ {}_1G^{(1)}_{ji} = || \, {}_1\langle j| U_s \left|\psi_i\right\rangle ||^2 = G_{ji}. \quad \square \tag{6.17}$$

This theorem reinforces the idea that the set of states $\left|\psi_i\right\rangle$ is natural for defining the proxies for the semiclassical walks, since the classical walk is obtained in the limit of only applying once the unitary evolution between measurements, which corresponds to a lack of coherent quantum evolution.

Theorem 6.2 (Classical Limit II)**.** *The classical walk is recovered for the semiclassical walk of class II with two quantum time steps, $t_q = 2$, using the single-step operator U_s. Therefore:*

$$ {}_2G^{(2)} = G. \tag{6.18}$$

For a proof see the supplementary material of our original paper [2]. This theorem makes us think that when measuring the second register it is more natural to use the original double-step Szegedy operator $W_s = U_s^2$ instead of U_s. Recall that this is indeed what is done in the quantum PageRank algorithm [31, 32].

Theorem 6.3 (Equivalence Between Semiclassical Classes)**.** *The semiclassical walk of class I obtained with a quantum time t_q is the same as the one of class II obtained with a quantum time $t_q + 1$, using the single-step operator U_s. Therefore:*

$$ {}_1G^{(t_q)} = {}_2G^{(t_q+1)}. \tag{6.19}$$

For a proof see the supplementary material of our original paper [2]. Due to this theorem, in the rest of this chapter, we only regard to the semiclassical walks of class I, since we are dealing with the general single-step operator U_s. However, for other scenarios where the evolution were performed with the double-step operator W_s, both classes would not be equivalent, since the quantum time steps would only correspond to even steps of U_s. For example, in the next chapter we will see an algorithm using the operator W_s, and the semiclassical walks of class II.

Theorem 6.4 (Quantum Limit). *The first classical step, $t_c = 1$, of a Szegedy semiclassical walk starting from a probability distribution vector p corresponds to the Szegedy quantum walk acting on the mixed state*

$$\rho = \sum_{i=0}^{N-1} p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{6.20}$$

Therefore:

$$\left({}_kG^{(t_q)}p\right)_i = \mathrm{Tr}\left[{}_k\langle i| U^{t_q}\rho U^{t_q\dagger} \left|i\right\rangle_k\right] \tag{6.21}$$

for both classes, i.e., $k = 1, 2$.

Proof. This theorem can be proved intuitively, although a mathematical proof is given in Section 9.4. To perform the semiclassical walk starting from a classical probability distribution p, each time we run the algorithm we initialize the system in a proxy state at random, according to the probabilities p_i. This formally corresponds to an initialization in the mixed state ρ, so that the first quantum evolution just does the quantum walk on it. □

6.3. Semiclassical Szegedy Walk on 1D Cycles

Once we have defined the semiclassical Szegedy walk, let us see some examples on 1D lattices. We are going to analytically solve the problem for the infinite line, and after that put cyclic boundary conditions to obtain the semiclassical walks on 1D cycles.

Recall that the classical transition matrix of the walk on the undirected line is given by

$$(G_u)_{ji} = \frac{1}{2}\delta_{j+1,i} + \frac{1}{2}\delta_{j-1,i}, \tag{6.22}$$

so that a walker at node i can jump to either node $i-1$ or $i+1$ with a 50% probability for both cases. The proxy states for the semiclassical Szegedy walk are

$$\left|\psi_i\right\rangle = \left|i\right\rangle_1 \otimes \frac{1}{\sqrt{2}}\left(\left|i-1\right\rangle_2 + \left|i+1\right\rangle_2\right). \tag{6.23}$$

We can define a set of orthogonal states to these proxy states as follows:

$$\left|\psi_i^{\perp}\right\rangle := \left|i\right\rangle_1 \otimes \frac{1}{\sqrt{2}}\left(\left|i-1\right\rangle_2 - \left|i+1\right\rangle_2\right). \tag{6.24}$$

With these sets we can calculate easily the action of the unitary operator $U_s = S_w(2\Pi - \mathbb{1})$ on the set of states $\left|i\right\rangle_1\left|i\pm 1\right\rangle_2$. These states can be expressed as:

$$\left|i\right\rangle_1\left|i-1\right\rangle_2 = \frac{1}{\sqrt{2}}\left(\left|\psi_i\right\rangle + \left|\psi_i^{\perp}\right\rangle\right), \tag{6.25}$$

$$\left|i\right\rangle_1\left|i+1\right\rangle_2 = \frac{1}{\sqrt{2}}\left(\left|\psi_i\right\rangle - \left|\psi_i^{\perp}\right\rangle\right). \tag{6.26}$$

Since the projector Π projects onto the subspace generated by the $\left|\psi_i\right\rangle$ states, we have that $\Pi\left|\psi_i\right\rangle = \left|\psi_i\right\rangle$. Moreover, since the states $\left|\psi_i^{\perp}\right\rangle$ are perpendicular to all the states $\left|\psi_i\right\rangle$, they are in the kernel

of the projector, so $\Pi \left|\psi_i^{\perp}\right\rangle = 0$. Using the expressions (6.23) and (6.24), the reflection part of the unitary operator yields:

$$(2\Pi - \mathbb{1}) \left|i\right\rangle_1 \left|i-1\right\rangle_2 = \left|i\right\rangle_1 \left|i+1\right\rangle_2 , \tag{6.27}$$

$$(2\Pi - \mathbb{1}) \left|i\right\rangle_1 \left|i+1\right\rangle_2 = \left|i\right\rangle_1 \left|i-1\right\rangle_2 . \tag{6.28}$$

Finally, we apply the swap S_w between the two registers, obtaining the action of U_s:

$$U_s \left|i\right\rangle_1 \left|i-1\right\rangle_2 = \left|i+1\right\rangle_1 \left|i\right\rangle_2 , \tag{6.29}$$

$$U_s \left|i\right\rangle_1 \left|i+1\right\rangle_2 = \left|i-1\right\rangle_1 \left|i\right\rangle_2 . \tag{6.30}$$

With (6.29) and (6.30) we can calculate the quantum evolution of the proxy states:

$$U_s \left|\psi_i\right\rangle = \frac{1}{\sqrt{2}} \left(\left|i+1\right\rangle_1 \left|i\right\rangle_2 + \left|i-1\right\rangle_1 \left|i\right\rangle_2\right) , \tag{6.31}$$

and for a general number t_q of quantum steps:

$$U_s^{t_q} \left|\psi_i\right\rangle = \frac{1}{\sqrt{2}} \left(\left|i+t_q\right\rangle_1 \left|i+t_q-1\right\rangle_2 + \left|i-t_q\right\rangle_1 \left|i-t_q+1\right\rangle_2\right) . \tag{6.32}$$

We see then that the quantum walk starting from a single node moves apart from that node in a symmetric form. The walker jumps t_q times from the starting node i, reaching nodes $i \pm t_q$ with a probability of 50% each.

If we impose cyclic boundary conditions, the additions and subtractions are performed modulo N. Then, we have the identification $-N = 0 = N$, and the same for each two integers with a difference of N. There are two interesting cases. The first one is when N is even and $t_q = N/2$. For the sake of simplicity, let us see the effect on the state $\left|\psi_0\right\rangle$ in a graph with $N = 6$ nodes, so $t_q = 3$ and

$$U_s^3 \left|\psi_0\right\rangle = \frac{1}{\sqrt{2}} \left(\left|3\right\rangle_1 \left|2\right\rangle_2 + \left|-3\right\rangle_1 \left|-2\right\rangle_2\right) = \frac{1}{\sqrt{2}} \left(\left|3\right\rangle_1 \left|2\right\rangle_2 + \left|3\right\rangle_1 \left|4\right\rangle_2\right) = \left|\psi_3\right\rangle , \tag{6.33}$$

where we have used the boundary conditions to identify -3 with 3, and -2 with 4. In this case, the walker reaches the same node from both sides, so the probability of measuring it is of 100%. The second case is when $t_q = N$ for any value of N. In that case:

$$U_s^N \left|\psi_0\right\rangle = \frac{1}{\sqrt{2}} \left(\left|N\right\rangle_1 \left|N-1\right\rangle_2 + \left|-N\right\rangle_1 \left|-N+1\right\rangle_2\right) = \frac{1}{\sqrt{2}} \left(\left|0\right\rangle_1 \left|-1\right\rangle_2 + \left|0\right\rangle_1 \left|1\right\rangle_2\right) = \left|\psi_0\right\rangle , \tag{6.34}$$

so the walker reaches the same starting point with certainty. Thus, the Szegedy quantum walk has a period of N acting on the proxy states $\left|\psi_i\right\rangle$. This is a great difference with the Hadamard coined quantum walk in 1D cycles, which is only periodic for a few values of N [40, 194].

Finally, from (6.32) we can calculate the semiclassical matrices using (6.11) for the semiclassical family of class I:

$${}_1G_{ji}^{(t_q)} = \frac{1}{2} \left(\delta_{j-t_q,i} + \delta_{j+t_q,i}\right) . \tag{6.35}$$

This is equivalent to a classical walk where each node i connects only to nodes $i \pm t_q$. Due to the periodicity of U_s acting on the proxy states, the semiclassical family also has a periodicity in the quantum time:

$${}_1G^{(t_q)} = {}_1G^{(t_q+N)}, \tag{6.36}$$

so there can be at most N different semiclassical walks.

Since a classical walk occurs on a graph where the edges encode the probabilities of the walker jumping from one node to another, we can also represent the semiclassical walk family as a set of weighted graphs, denoted as semiclassical graphs. In Figure 6.4 it is shown an example of the semiclassical graphs for the cycle with $N = 6$ nodes. For the first quantum time, $t_q = 1$, we obtain the same graph as in the classical walk, so each node links to its immediate neighbors. The same result is obtained for $t_q = 5$. However, for other values of the quantum time we obtain genuine walks. For $t_q = 2$ and $t_q = 4$ each node connects to the second nearest neighbors, so the graphs breaks into triangles. Thus, if a particle starts at node 0, it will perform a walk equivalent to a classical one in the triangle formed by nodes 0, 2 and 4. For $t_q = 3$ each node links only to the opposite node in the graph, breaking the graph into lines of 2 nodes. Finally, for $t_q = 6$ each node links only to itself with a loop, so the graphs breaks into single nodes. This would be equivalent to $t_q = 0$, which is not actually a walk since there is not quantum evolution, so that the transition matrix is just the identity. For a larger value of the quantum time the sequence of graphs is repeated due to the periodicity of $N = 6$. The breaking in the connectivity of the graph is due to a degeneration of the eigenvalue 1 of the semiclassical matrix, which agrees with the results of the continuous quantum time version [193], where the same cycle with six nodes was also broken for concrete values of the quantum time.

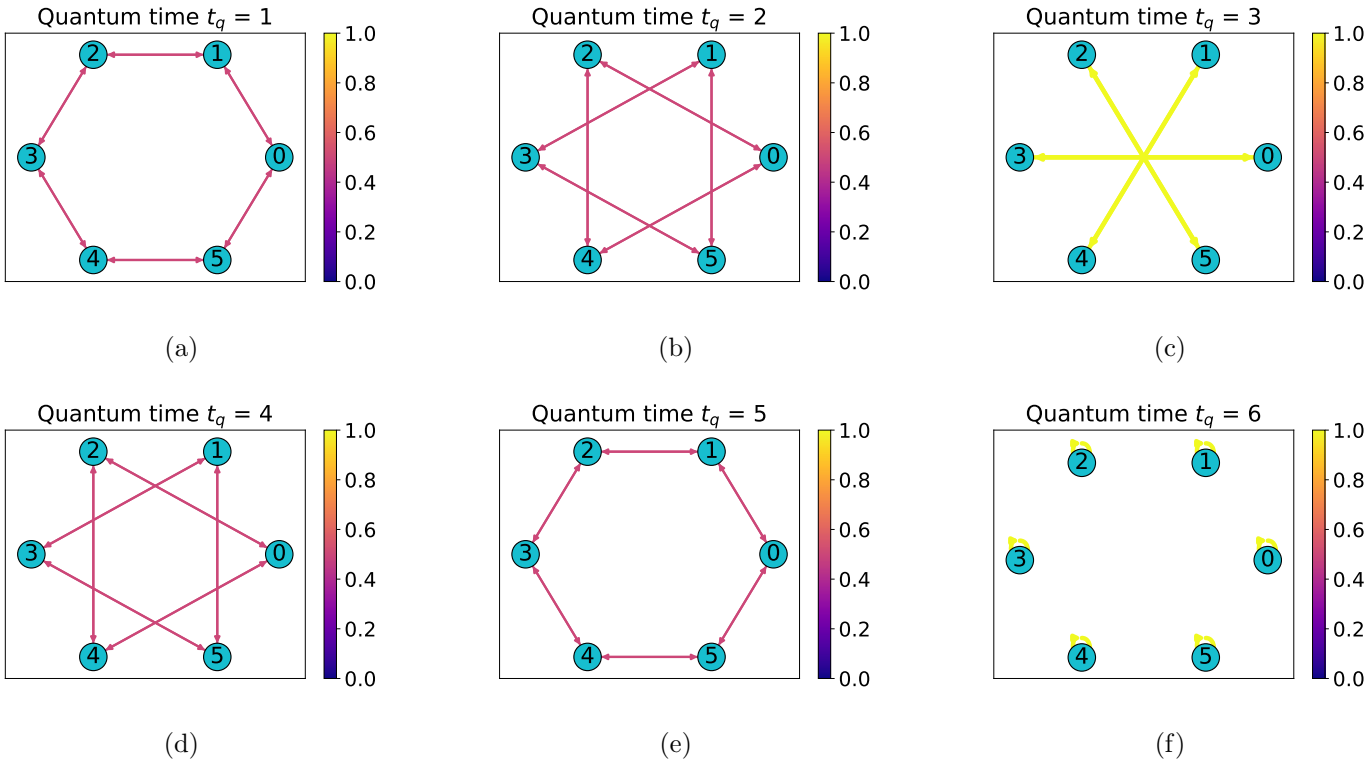


Figure 6.4: *Semiclassical graphs for the cycle with $N = 6$ nodes for (a) $t_q = 1$, (b) $t_q = 2$, (c) $t_q = 3$, (d) $t_q = 4$, (e) $t_q = 5$, (f) $t_q = 6$. The weights of the edges are represented by the colormap. In this case, there are only two possible weights:* 0.5 *represented by a magenta line, or* 1 *represented by a yellow line. All the edges are bidirectional.*

We have seen that *a priori* we could have N different semiclassical graphs. Nevertheless, here there are some graphs inside a period that are repeated. This is due to the fact that the semiclassical

walks come from the projection of quantum states, so that different quantum states can yield the same position after the measurement. As an example, note that the quantum states after the evolution from $|\psi_0\rangle$ are not the same for $t_q = 2$ and $t_q = 4$:

$$U_s^2 |\psi_0\rangle = \frac{1}{\sqrt{2}} \left(|2\rangle_1 |1\rangle_2 + |4\rangle_1 |5\rangle_2 \right), \tag{6.37}$$

$$U_s^4 |\psi_0\rangle = \frac{1}{\sqrt{2}} \left(|4\rangle_1 |3\rangle_2 + |2\rangle_1 |3\rangle_2 \right). \tag{6.38}$$

Nevertheless, when the first register is measured, in both cases there is a 50% probability of measuring node 2 or node 4, giving rise to the same semiclassical walk. To see better how the semiclassical matrices between different quantum times are related, in Figure 6.5 it is shown the oscillation of the semiclassical matrices ${}_1G^{(t_q)}$ with respect to the quantum time, where we can see that there are only 4 different semiclassical walks in the family. In the same figure it is shown the period of the unitary operator U_s, and we effectively observe that there are six different operators $U_s^{t_q}$.

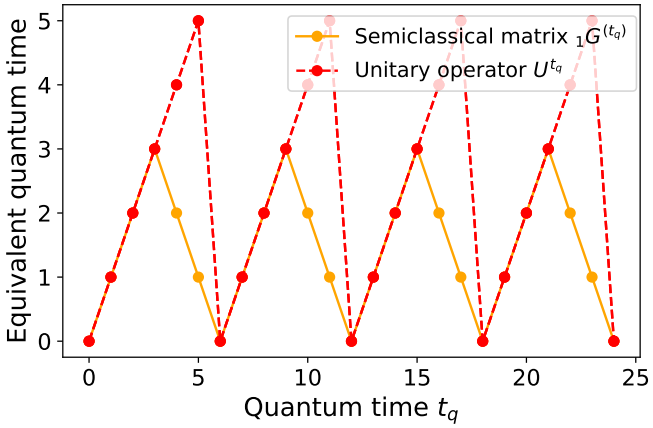


Figure 6.5: *Periodicity of the semiclassical matrices ${}_1G^{(t_q)}$ and the unitary evolution operator $U_s^{t_q}$ with respect to the quantum time t_q for the cycle with $N = 6$ nodes. At each quantum time it is represented the minimum value of t_q for which there is a matrix that is equal. For example, for $t_q = 4$ the semiclassical matrix is equal to the one at $t_q = 2$. However, the unitary operator is not still repeated, so that the equivalent quantum time is also $t_q = 4$. Time $t_q = 0$ is not an actual walk, but is used to represent that the matrix is equal to the identity.*

The number of different semiclassical walks depends on how many jumps the walker makes between the measurements. The number of jumps is just t_q, and since it jumps in both directions, due to the cyclic boundary conditions we would have in general that the number of different graphs is

$$\#\text{ graphs} = \lfloor N/2 \rfloor + 1. \tag{6.39}$$

Furthermore, the type of subgraphs that the classical graph can be broken into depends on how the number of nodes N can be factorized. For $N = 6$ we have $6 = 2 \times 3 = 1 \times 6$, so we can have a single hexagon, two triangles, three lines or six separate nodes.

As another example, in Figure 6.6 it is shown the semiclassical graphs for the cycle with $N = 7$ nodes. Since N is prime, in this case the graph cannot be broken in more than single nodes, although we can also have different graphs. For $t_q = 1$ we recover the classical walk as expected, and the same

for $t_q = 6$. For $t_q = 2$ and $t_q = 5$ node 0 connects with nodes 2 and 5. But node 2 connects to 4, 4 to 6, 6 to 1, 1 to 3, 3 to 5, and 5 to node 0 again. So the graph is not broken. In this case it is again as a classical walk on the cycle with seven nodes, but the nodes are permuted. If we unroll the graph, it is similar to having a cycle formed by the chain of nodes $0-2-4-6-1-3-5$. For $t_q = 3$ and $t_q = 4$ something similar happens, but with a different order of the nodes. Finally, for $t_q = N = 7$ the graph is broken into single nodes, closing a period in the quantum time. We can check that relation (6.39) holds true, having 4 different semiclassical walks in this case.

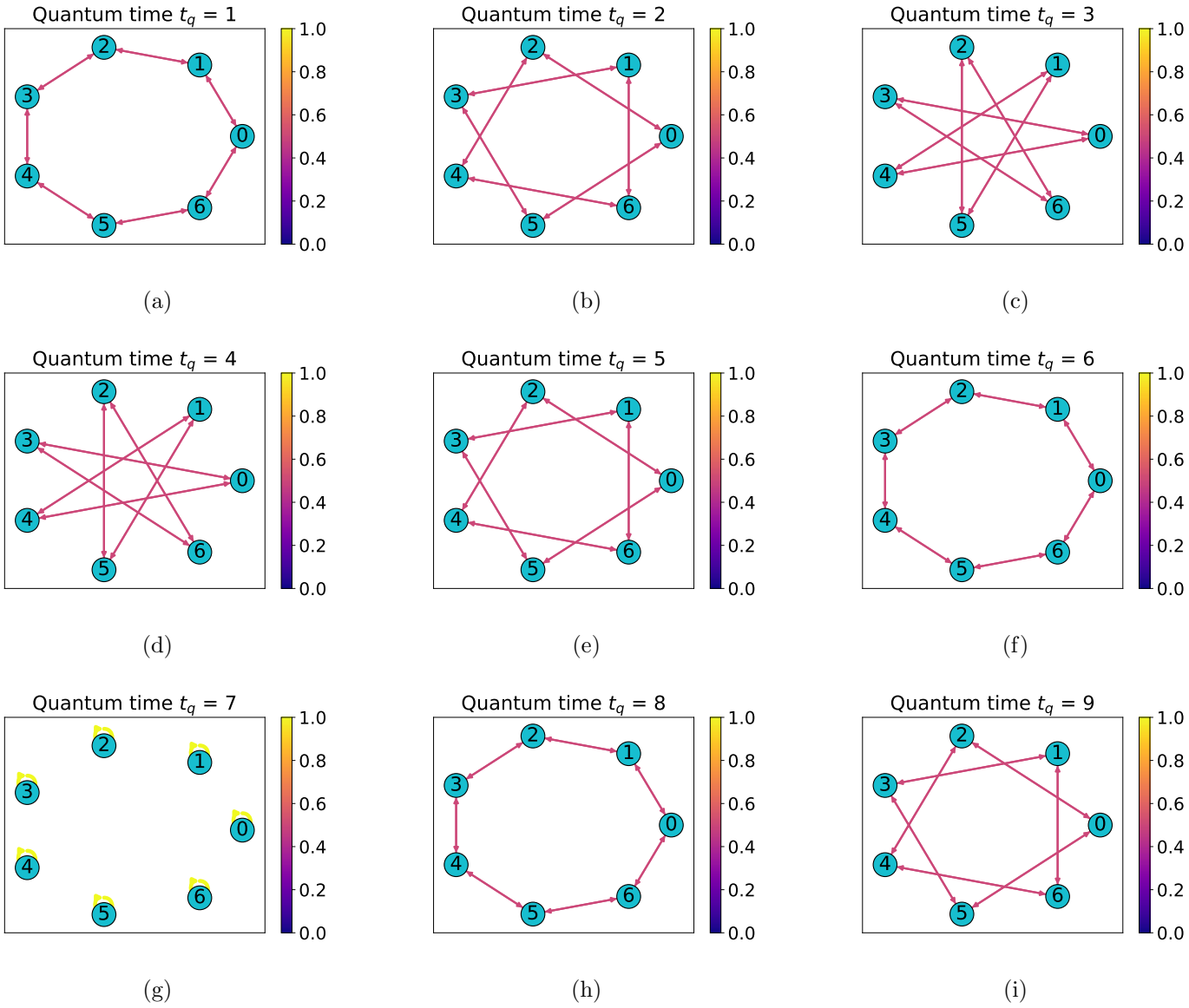


Figure 6.6: *Semiclassical graphs for the cycle with $N = 7$ nodes for (a) $t_q = 1$, (b) $t_q = 2$, (c) $t_q = 3$, (d) $t_q = 4$, (e) $t_q = 5$, (f) $t_q = 6$, (g) $t_q = 7$, (h) $t_q = 8$, (i) $t_q = 9$. The weights of the edges are represented by the colormap. In this case, there are only two possible weights:* 0.5 *represented by a magenta line, or* 1 *represented by a yellow line. All the edges are bidirectional.*

In Figure 6.7 it is shown the periodicity of the semiclassical matrices and the quantum evolution operator U_s with the quantum time. It is clearly seen that the period of the semiclassical walks is $N = 7$, having only 4 different semiclassical matrices. However, in this case the period of the

unitary operator is 14 instead of 7. This is due to the fact that despite U_s having a period of N when acting on the set of states $|\psi_i\rangle$, this set only generates an N-dimensional subspace of the entire N^2-dimensional Hilbert space. Since $U_s|\psi_i\rangle = S_w|\psi_i\rangle$, U_s also has a period of N acting on the set of the swapped states $S_w|\psi_i\rangle$. This two sets of states generate an invariant subspace known as dynamical subspace [31]:

$$\mathcal{H}_D := \text{span}\{|\psi_i\rangle, S_w|\psi_i\rangle\}. \tag{6.40}$$

The proof that it is invariant can be found in Appendix A.1. Any vector $|\phi\rangle$ in the orthogonal complement of $\mathcal{H}_D$ is perpendicular to both the $|\psi_i\rangle$ and the $S_w|\psi_i\rangle$ states. Thus, $\Pi|\phi\rangle = 0$, and the first application of U_s yields $U_s|\phi\rangle = -S_w|\phi\rangle$. Since $|\phi\rangle$ is perpendicular to the states $S_w|\psi_i\rangle$, then $S_w|\phi\rangle$ is perpendicular to the states $|\psi_i\rangle$, so $\Pi S_w|\phi\rangle = 0$. Thus, a second application of U_s yields:

$$U_s^2|\phi\rangle = -U_s S_w|\phi\rangle = +S_w^2|\phi\rangle = |\phi\rangle, \tag{6.41}$$

so the period of U_s on $\mathcal{H}_D^\perp$ is just 2. The total period of the unitary operator U_s is the least common multiple of the periods in both subspaces. Thus, for N even the period is N, whereas for N odd the period is $2N$.

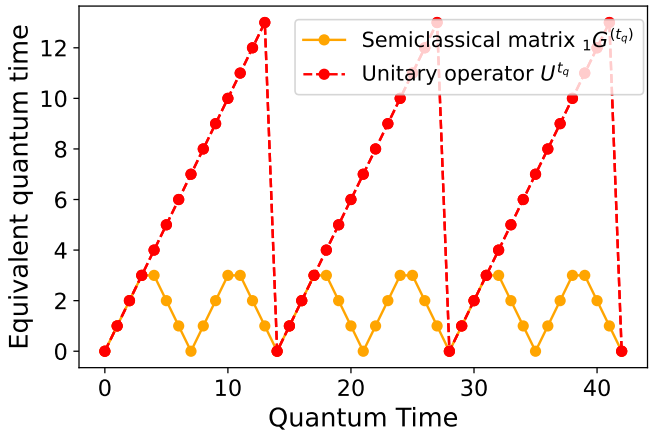


Figure 6.7: *Periodicity of the semiclassical matrices ${}_1G^{(t_q)}$ and the unitary evolution operator $U_s^{t_q}$ with respect to the quantum time t_q for the cycle with $N = 7$ nodes. At each quantum time it is represented the minimum value of t_q for which there is a matrix that is equal. For example, for $t_q = 10$ the semiclassical matrix is equal to the one at $t_q = 3$. However, the unitary operator is not still repeated, so that the equivalent quantum time is also $t_q = 10$. Time $t_q = 0$ is not an actual walk, but is used to represent that the matrix is equal to the identity.*

In this thesis, we have shown only the results for $N = 6$ and $N = 7$ nodes. More results on different 1D cycles can be found in the supplementary material of our original paper [2].

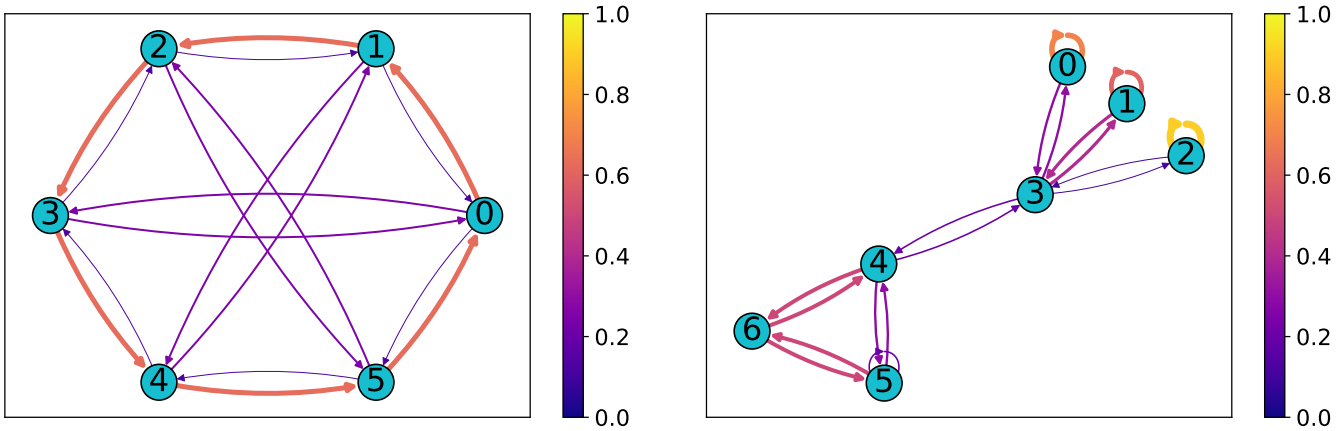
6.4. Inhomogeneity-Driven Symmetry Breaking

Let us introduce the following two concepts about weighted graphs:

Definition 6.3 (Symmetric graph)**.** *It is a weighted graph whose transition matrix is symmetric, meaning that between each pair of nodes the probability of going from one to the other is the same in both directions. If G is the transition matrix, then $G = G^T$.*

Definition 6.4 (Homogeneous graph)**.** *It is a weighted graph whose transition matrix elements only depend on the relative position of the nodes. Thus, all the nodes have the same connectivity pattern and the same weights in their links.*

In Figure 6.8(a) it is shown an instance of asymmetric and homogeneous graph. On the one hand, the transition matrix is not symmetric, since the probability of going from node 0 to node 1 is greater than from node 1 to node 0. On the other hand, each node has the same behavior, meaning that the weights of their links depend only on the relative distance to the other nodes. In this case, each node i connects with nodes $i \pm 1$ and $i + 3$, and the weights are the same for each node i. Thus, the graph is homogeneous.



(a) (b)

Figure 6.8: *(a) Asymmetric and homogeneous graph. Between each pair of nodes the weights can be different in both directions, so it is asymmetric. However, all the nodes have the same connectivity pattern, with weights that only depend on the relative distance between the nodes. (b) Symmetric and inhomogeneous graph. Between each pair of nodes the weights are the same in both directions, so it is symmetric. However, each node has a different connectivity pattern, so it is inhomogeneous.*

In the examples of 1D cycles, all the semiclassical graphs are symmetric. This could be due to the fact that the classical graphs were also symmetric. However, they were also homogeneous. So, we wonder what happens when the classical graph is symmetric but inhomogeneous. With this purpose, we have built the graph shown in Figure 6.8(b). It can be seen that the weights between each pair of nodes have the same intensity, so the transition matrix is symmetric. Nevertheless, each node has different connectivity patterns. For example, node 3 connects to four nodes, whereas node 0 only connects to another node and itself with a loop. Moreover, nodes 0, 1 and 2 have also different weights in their links with node 3 and the self-loop.

We have simulated the semiclassical walks of the inhomogeneous graph in Figure 6.8(b), and the first six semiclassical graphs are shown in Figure 6.9. For the first quantum time, $t_q = 1$, we obtain the same as the classical graph, which is symmetric. However, for any other quantum time, we observe that the symmetry has been broken. For example, for the graph with $t_q = 2$ note that the weight in the edge that goes from node 6 to 5 is stronger than the weight from node 5 to 6. Moreover, since this is a more general case than the 1D cycles, there is not a periodicity in the semiclassical family. We have made simulations with other homogeneous symmetric graphs constructed at random, finding

that the symmetry is never broken. Thus,we can conclude that the inhomogeneity is the cause of the symmetry breaking.

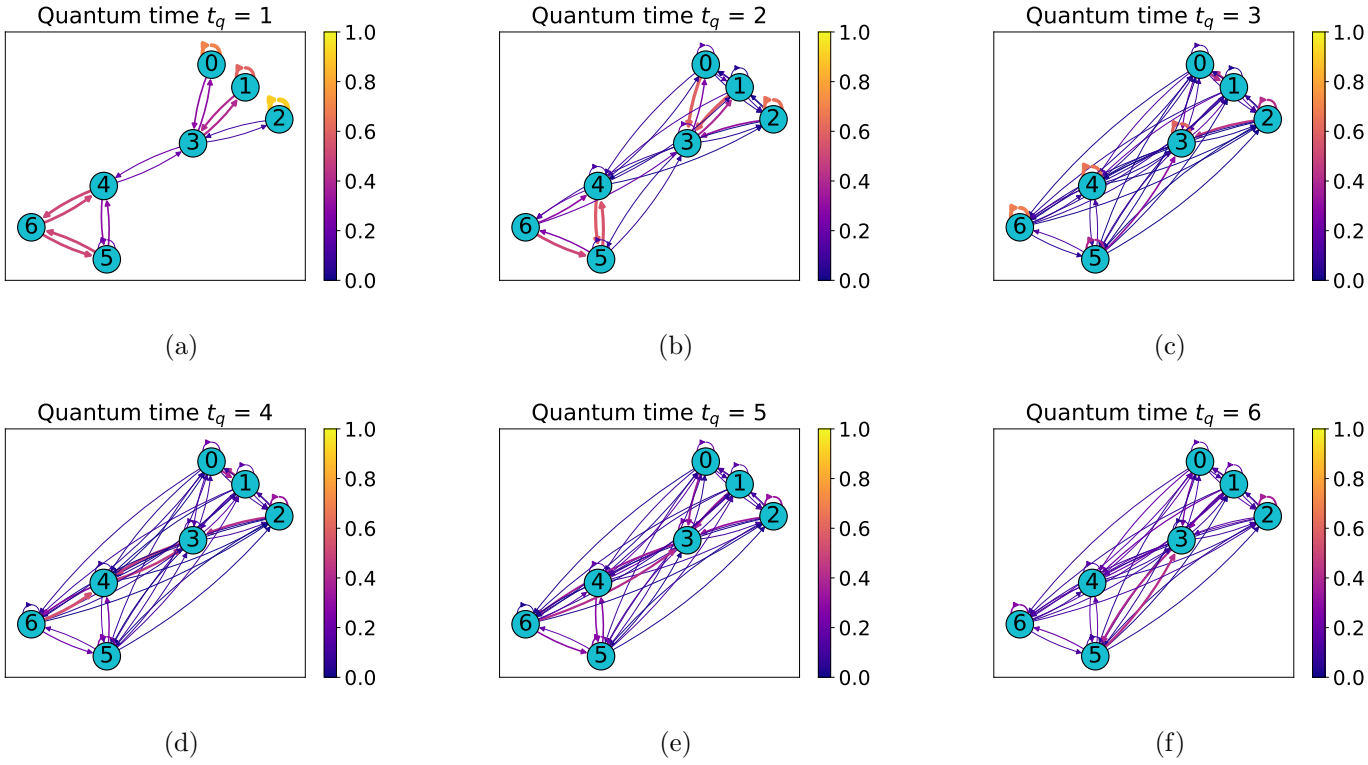


Figure 6.9: *Semiclassical graphs for the graph of Figure 6.8(b) for (a) $t_q = 1$, (b) $t_q = 2$, (c) $t_q = 3$, (d) $t_q = 4$, (e) $t_q = 5$, (f) $t_q = 6$. The weights of the edges are represented by the colormap. Moreover, to ease the visualization, the width of the edges are proportional to their weights. It can be seen how the symmetry of the graph is broken from $t_q = 2$ onwards.*

The fact that the semiclassical transition matrices are not symmetric anymore opens an application of the semiclassical walk to the problem of ranking nodes. Let us formulate the following theorems about symmetric transition matrices.

Theorem 6.5 (Uniform Distribution for Symmetric Classical Walks)**.** *For a symmetric transition matrix $G = G^T$, the uniform distribution is an eigenvector with eigenvalue* 1 *[195].*

Proof. Let G be the transition matrix, so that $G_{ij} = G_{ji}$, and let $\mathbf{v}$ be the uniform probability vector, so that $\mathbf{v}_i = 1/N \ \forall \ i$. We apply the transition matrix to this vector:

$$[G\mathbf{v}]_i = \sum_{j=0}^{N-1} G_{ij}\mathbf{v}_j = \sum_{j=0}^{N-1} G_{ij}\frac{1}{N} = \frac{1}{N}\sum_{j=0}^{N-1} G_{ji} = \frac{1}{N} = \mathbf{v}_i, \tag{6.42}$$

where we have used that G is column-stochastic, so that each column adds up to 1 (3.2). □

Theorem 6.6 (Uniform Distribution for Symmetric Szegedy Quantum Walks)**.** *Let G be a symmetric transition matrix, and U_s the associated Szegedy unitary evolution operator. Then, the uniform linear*

combination of all the $|\psi_i\rangle$ *states, denoted as*

$$\left|\Psi^{(0)}\right\rangle := \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\psi_i\rangle \,, \tag{6.43}$$

is an eigenvector of U_s *with eigenvalue* 1.

Proof. Since $\left|\Psi^{(0)}\right\rangle$ is a linear combination of the $|\psi_i\rangle$ states, the action of U_s on it is just S_w. So, using the fact that $G_{ij} = G_{ji}$:

$$\begin{aligned} U_s \left|\Psi^{(0)}\right\rangle &= S_w \left[\frac{1}{\sqrt{N}} \sum_{i,k=0}^{N-1} \sqrt{G_{ki}} \, |i\rangle_1 \, |k\rangle_2\right] = \frac{1}{\sqrt{N}} \sum_{i,k=0}^{N-1} \sqrt{G_{ki}} \, |k\rangle_1 \, |i\rangle_2 \\ &= \frac{1}{\sqrt{N}} \sum_{i,k=0}^{N-1} \sqrt{G_{ik}} \, |k\rangle_1 \, |i\rangle_2 = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |\psi_k\rangle = \left|\Psi^{(0)}\right\rangle . \quad \square \end{aligned} \tag{6.44}$$

The limiting distribution of the classical walk could be used to rank the nodes, in a similar manner as in the PageRank algorithm [141–144] (see Section 3.10), using as Google matrix directly the transition matrix G. However, for a symmetric transition matrix, in the case that the walk converges, it will converge to the uniform distribution, so no useful information can be obtained. In the case of a quantum walk starting from the uniform superposition of all the $|\psi_i\rangle$ states, no information can be obtained either, since it is an eigenvector of the unitary evolution operator U_s. Therefore, the quantum PageRank algorithm [31,32] based on a symmetric transition matrix is also useless. Note that even if we mixed the transition matrix G with the uniform matrix as in equation (3.59), the new Google matrix would still be symmetric.

Thanks to the symmetry breaking in the semiclassical graphs, the semiclassical walks converge to distributions different to the uniform one. The limiting distributions for the six semiclassical graphs of Figure 6.9 are shown in Figure 6.10(a). We now obtain different rankings for the nodes, but they are different for each semiclassical graph. The same as in the quantum PageRank algorithm [31], the distributions oscillate with the quantum time, as shown in Figure 6.10(b). To obtain an objective classification, we average the distributions over the different quantum times, in a similar manner as in the quantum PageRank algorithm [31,128]. For the semiclassical walk, the averaged probability distribution ends up converging, the same as for the quantum walk [32,163], as can be seen in Figure 6.10(c). Finally, in Figure 6.10(d) it is shown the averaged distribution, compared with the uniform ones using the classical and quantum PageRank algorithms with the symmetric classical transition matrix G.

The final question is how this ranking relates to the structure of the network. Node 3 is the most important, being the node with more connections. The following most important node is node 4, which is the other central node. So it seems that the connectivity of the nodes plays a major role in the ranking. Furthermore, note that the differences in weights also play a role. Nodes 0, 1, and 2, all linking to node 3 and having a self-loop, are not degenerate due to the differences in the values of the weights. Node 1 has the strongest weights in the edges with node 3, and thus is the most important of the three.

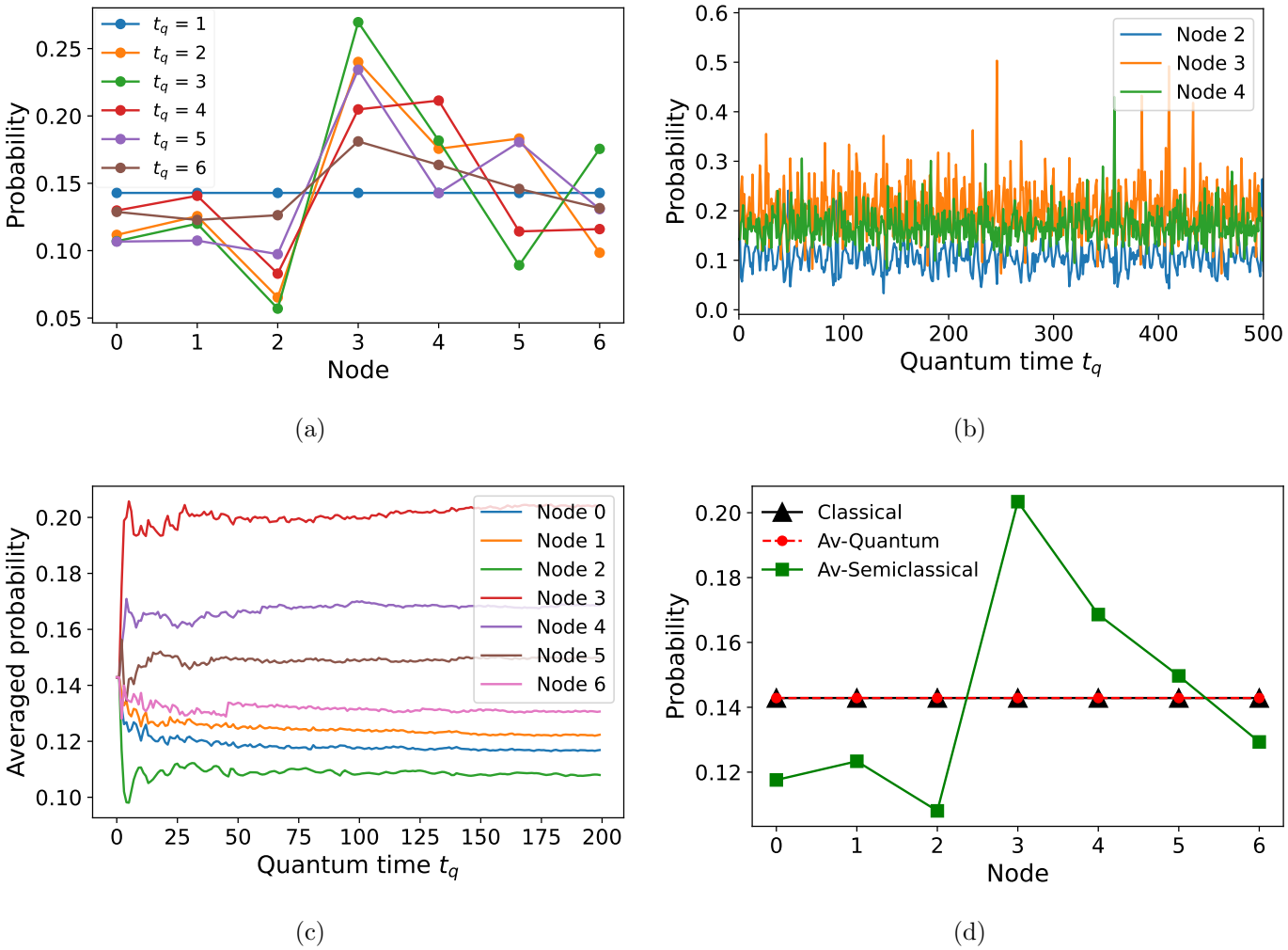


Figure 6.10: *Limiting distributions for the first six semiclassical walks of the graph in Figure 6.8(b). (b) Probability of some nodes in the different semiclassical walk limiting distributions versus the quantum time. (c) Time average of the limiting distributions versus the quantum time. (d) Averaged limiting distribution of the semiclassical walks. It is compared with the classical and quantum PageRank algorithms on the classical graph, which yield uniform distributions.*

6.5. Comparison with the Continuous Time Approach

The measurement-induced quantum walk [193] can be understood as a semiclassical walk where the quantum time is a continuous variable. As we discussed above, the Hilbert space is simply the span of the computational basis $|i\rangle$. Thus, in contrast with our discrete-time approach, there is no need to restart any coin register after the measurement.

In contrast to the Szegedy quantum walk, the unitary evolution operator is obtained from the exponentiation of a Hermitian operator H related to the adjacency matrix of a graph, so that $U(t_q) = \exp(-\mathrm{i}Ht_q)$. Thus, whereas the Szegedy quantum walk can be performed on any weighted graph, in the continuous-time quantum walk there is the restriction that the graph must be undirected, so

that the adjacency matrix is symmetric. Therefore, H is real, so that $H^T = H$, and the unitary matrix is also symmetric:

$$U^T(t_q) = e^{-\mathrm{i}H^T t_q} = e^{-\mathrm{i}Ht_q} = U(t_q). \tag{6.45}$$

Given the definition of the semiclassical matrices of the measurement-induced quantum walk in (6.1) and the symmetry of $U(t_q)$, we have that

$$G_{ji}(t_q) = |\langle j| U(t_q) |i\rangle|^2 = |U_{ji}(t_q)|^2 = |U_{ij}(t_q)|^2 = |\langle i| U(t_q) |j\rangle|^2 = G_{ij}(t_q), \tag{6.46}$$

so that the transition matrices $G(t_q)$ are also symmetric. This means that the uniform distribution is always a stationary state of the measurement-induced quantum walk. In the case that the eigenvalue 1 is not degenerate, it is as if the repeated measurements drove the system to a high temperature classical limit [193, 196]. This is not the case for our discrete-time semiclassical walks, since we are not only measuring repeatedly, but also restarting to the desired proxy states using the classical information from the measurement. Thus, the restart scheme prevents this behavior.

Regarding the results in 1D cycles, the measurement-induced quantum walk is also able to break the graph into subgraphs [193]. However, this breaking occurs for exceptional values of the quantum time, so they are very infrequent inside the family of semiclassical walks. Moreover, due to the continuous behavior of the quantum walk, except in the cases that the graph is broken, all the transition matrices are fully connected, meaning that there is a non-null probability of jumping from a node to any other or itself. Thus, the equivalent classical walk encoded by the Hermitian matrix H is never recovered in contrast to our discrete quantum time version.

6.6. Experimental Semiclassical Walk on IBM-Q

In order to show that the semiclassical walk framework functions on a real quantum processor unit (QPU), we have performed an experiment on the IBM Quantum Platform [33], with the processor ibmq-manila. This platform has been previously used to demonstrate experimentally other quantum walks such as the staggered quantum walk [149, 197], and even the measurement-induced quantum walk [198]. Since current QPUs are very error prone, we have chosen a graph with only two nodes, where the Szegedy quantum walk requires only two qubits, one per register. The decoherence effects of quantum computers make the system end up in a uniform distribution. Therefore, to be able to distinguish between environment noise or actual results, we have taken a weighted asymmetric graph, so that the equilibrium distribution is different to the uniform one. The classical transition matrix of this graphs is

$$G = \begin{pmatrix} 0.1 & 0.2 \\ 0.9 & 0.8 \end{pmatrix}. \tag{6.47}$$

In our experiment, we wanted to obtain the probability distribution $p(t)$ of the walker being at each node for each classical time t_c. To do so, we must run the semiclassical walk several times to sample from the probability distributions, and then estimate the probabilities by dividing the number of times the walker ends up at a node by the number of times the circuit has been run. Moreover, the classical walk evolution equation (3.5) tells us that we can start from a non-trivial probability distribution $p(0)$. This can be achieved by initializing the circuit each time with a state taken at random from $p(0)$. To demonstrate that this actually works, we have taken the initial probability distribution as $p(0) = (0.8, 0.2)^T$.

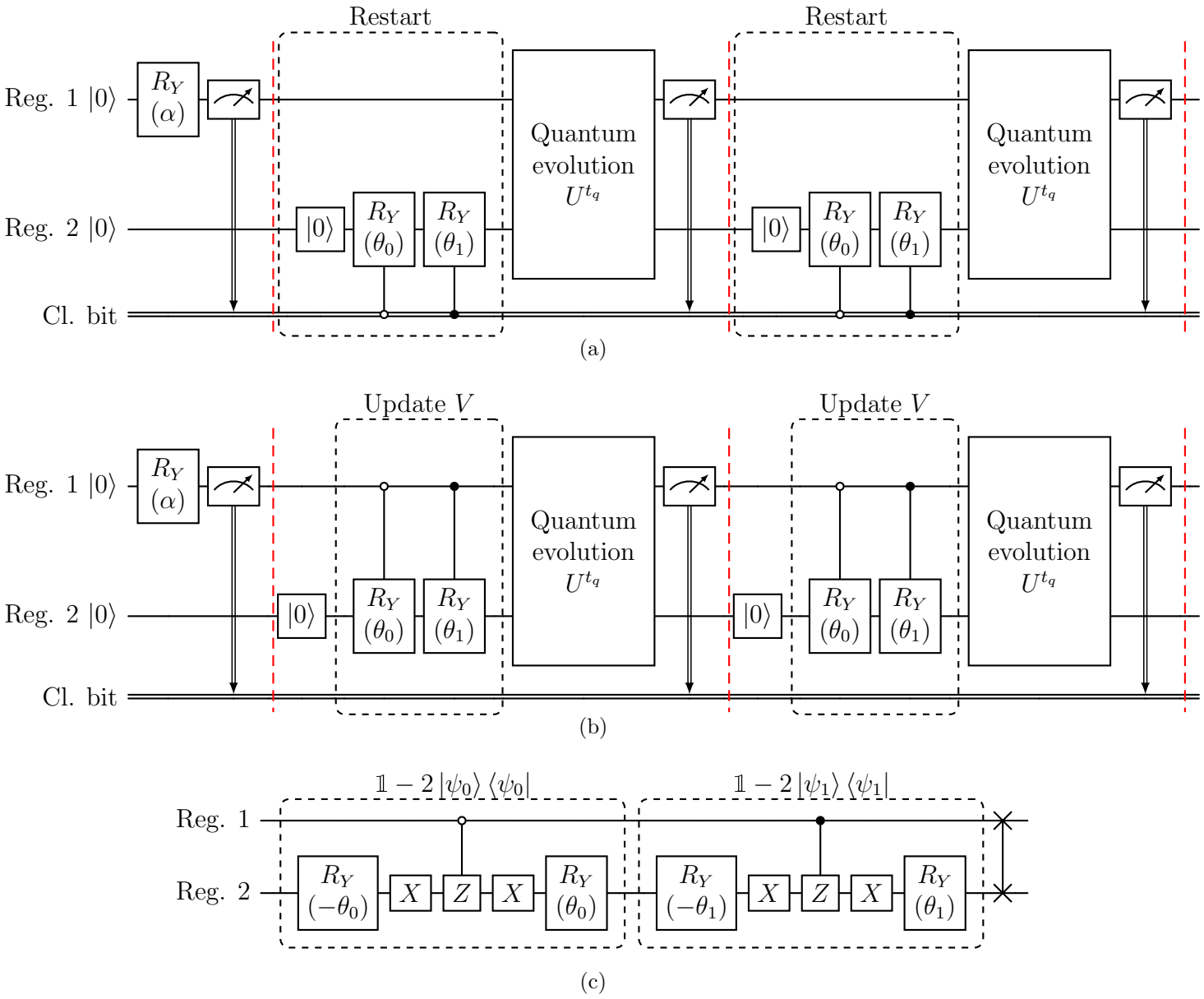


Figure 6.11: *(a) Quantum circuit for the semiclassical walks on the graph with two nodes whose classical transition matrix is given in (6.47). This figure is an example with two classical steps, $t_c = 2$, for a generic quantum time t_q. (b) Quantum circuit for the semiclassical walks where the classically-controlled gates of (a) have been substituted by purely quantum gates. (c) Circuit for the unitary operator U_s. This operator can be expanded as $U_s = -S_w(\mathbb{1} - 2|\psi_1\rangle\langle\psi_1|)(\mathbb{1} - 2|\psi_0\rangle\langle\psi_0|)$. Moreover, using (6.48) we have that $\mathbb{1} - 2|\psi_i\rangle\langle\psi_i| = (\mathbb{1} \otimes R_Y(\theta_i))(\mathbb{1} - 2|i,0\rangle\langle i,0|)(\mathbb{1} \otimes R_Y(-\theta_i))$. Finally, the minus reflections $\mathbb{1} - 2|i,0\rangle\langle i,0|$ can be implemented with controlled-Z gates surrounded by X gates. The values of the parameters are $\alpha = 0.927$, $\theta_0 = 2.5$ and $\theta_1 = 2.21$.*

The general quantum circuit to perform the semiclassical walks is shown in Figure 6.11(a). The circuit always starts with both qubits in the state $|0\rangle$. Ideally, we would initialize the first register in the state $|0\rangle_1$ 80% of the time, and the other 20% in the state $|1\rangle$. To emulate this effect with a quantum circuit, we apply an $R_Y(0.927)$ gate, so that the first register is converted to $\sqrt{0.8}\,|0\rangle + \sqrt{0.2}\,|1\rangle$. Thus, after measuring it, the first register is initialized in $|0\rangle_1$ with a 80% probability, and in $|1\rangle_1$ with a 20% probability. The result of this measurement corresponds to the position of the walker at $t_c = 0$. For any other classical time, we repeat the following block. First, the second register is reset to $|0\rangle_2$, which do not affect to the first register since they are not entangled due to the previous measurement. After that, we prepare the proxy state conditioned by the result of the previous measurement, using classically-controlled $R_Y(\theta_i)$ gates. We know that

$$(\mathbb{1} \otimes R_Y(\theta_i))\,|i\rangle_1\,|0\rangle_2 = |\psi_i\rangle\,, \tag{6.48}$$

where $\theta_0 = 2.5$ and $\theta_1 = 2.21$. Thus, when the classical bit is in 0, and so the first register is in $|0\rangle_1$, the proxy state $|\psi_0\rangle$ is prepared, whereas when it is in 1 the state $|\psi_1\rangle$ is prepared instead. Once the proxy state is prepared, we apply the quantum evolution U_s the number of quantum times t_q required. Finally, we measure the state in the first register obtaining the position of the walker for that classical time t_c. Note that this circuit uses classically-controlled gates to restart the system, so that only the gate corresponding to the current position is applied. Nevertheless, despite the fact that classically-controlled gates are theoretically possible to implement, current QPUs of IBM do not allow them. For this reason, we substitute them by purely quantum controlled gates, obtaining the circuit in Figure 6.11(b). In this case, the two controlled gates form the update operator V for this walk.

With regard to the operator U_s, its circuit compilation is shown in Figure 6.11(c). *A priori* we could diagonalize the reflection operator as $R = VDV^\dagger$, as explained in Section 3.6. The diagonal operator would apply a reflection around the state $|0\rangle_2$ on the second register, which would be compiled as a minus Z gate. Therefore, this circuit would need four two-qubit gates, two for each update operator. In real quantum computers, these gates are the main contributors to the noise, so we want to reduce their number as much as possible. For this reason, in this case we provide an alternate compilation, expanding the reflection as the product of reflections around each $|\psi_i\rangle$ state, as shown in Section 2.3.4. Thus, only two controlled gates are needed.

We have performed the experiments for the first three semiclassical walks of the family, $t_q = 1, 2, 3$, whose semiclassical graphs are shown in Figures 6.12(a)-6.12(c). In each case, we have performed ten independent experiments and averaged the results. For each experiment the probability distributions were obtained sampling the circuit 20000 times. The results are shown in Figures 6.12(d)-6.12(f). On the one hand, in order to check the accuracy of the results, we can simulate the same circuit in the Aer simulator of Qiskit [84]. This is a stochastic simulator that simulates a fault-tolerant quantum computer, also sampling a finite number of times to estimate the probability distributions. In our case we sample 20000 times the circuits as in the real experiments. On the other hand, we can also calculate deterministically the theoretical probability distributions obtaining the semiclassical matrices and using equation (3.5). In all cases, the simulated circuits yield the same results as the theoretical formulation, verifying that the implemented circuit in Figure 6.11(b) works.

The first semiclassical walk, for $t_q = 1$, corresponds actually to the classical walk. We can see that the initial probability of the walker being at node 1 is 0.2. After the first step this probability rises to 0.88. Then, it goes down to 0.82 and converges. The experimental results agree with the theoretical behavior. However, due to the errors of the real QPU, it converges to approximately 0.77,

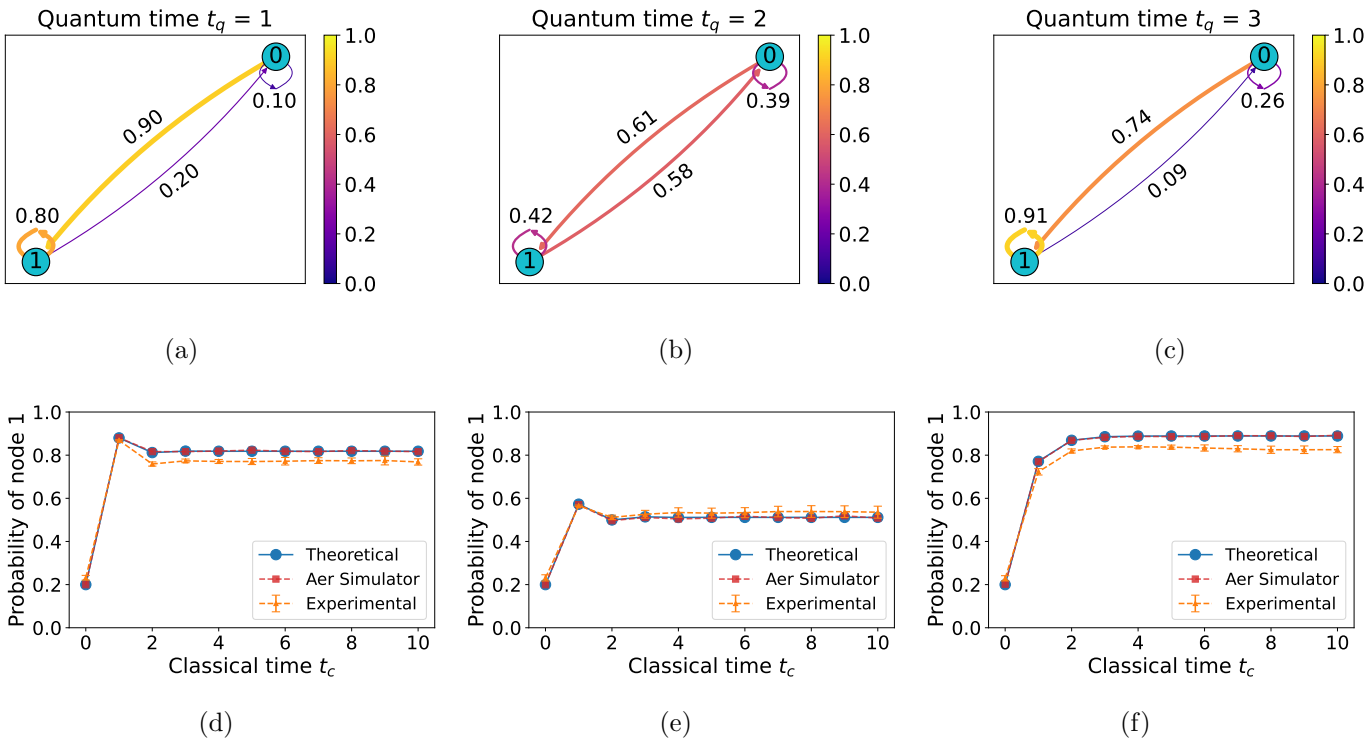


Figure 6.12: *(a)-(c) Semiclassical graphs for the classical walk given by (6.47) for (a) $t_q = 1$, (b) $t_q = 2$, (c) $t_q = 3$. The weights of the edges are represented by the colormap and also over the edges. (d)-(f) Experimental results for the semiclassical walks for (d) $t_q = 1$, (e) $t_q = 2$, (f) $t_q = 3$. Since there are only two nodes, the probability distribution can be represented by the probability of measuring node 1. The error bars are computed with the standard deviations between ten different experiments. The results are compared with the theoretical results and the ones from the Aer simulator.*

so the relative error is of 6%. For the second semiclassical walk, the graph is roughly symmetric, so it is not surprising that the limiting distribution is almost uniform. In this case, the theoretical result converges to 0.51, whereas the experimental one to 0.54, so there is an approximate error again of 6%. Finally, for the third semiclassical walk, the theoretical result converges to 0.89, whereas the experimental one to 0.83, with an error of 7%. Taking into account that current QPUs are still very error prone, and we have not used any error mitigation nor error correction technique, our experimental results have an incredibly well agreement with the theoretical ones.

6.7. Summary of Results and Conclusions

- We have introduced the semiclassical framework in quantum walks. Each semiclassical walk can be seen as a classical walk where the transition matrix encodes a quantum evolution, and thus the dynamics is a mixture of classical and quantum evolution. Moreover, to be able to perform the semiclassical walk from quantum walks in discrete time, we have introduced the restart scheme and the proxy states, which encode the position of the walker on a graph.

- We have formulated the semiclassical walks from the Szegedy quantum walk, so that they can

be applied to arbitrary weighted graphs. The quantum states for the Szegedy quantum walk are formed by two registers, and each of both can yield information about the position of the walker. Thus, we have defined two classes of semiclassical Szegedy walks depending on which register we measure. Furthermore, we have proved that these two classes of semiclassical walks include the original classical walk, and there is an equivalence between them. Although these theorems have been proved in the absence of oracles or complex-phase extensions, similar proofs not covered in this thesis could be performed, and the same results hold.

- The semiclassical Szegedy walks can be solved analytically for 1D lattices, such as cycles. We have obtained the semiclassical transition matrices for this case, and we have observed that for certain members of the semiclassical family the walk occurs in a broken graph. Moreover, there are some cases where the cycle is not broken, but the nodes are permuted. This results agree with the measurement-induced quantum walk [193], which is similar to a semiclassical walk but with a continuous quantum time, where it has also been observed that the 1D cycles could be broken.

- We have observed that if the classical graph is symmetric but not all the nodes have the same behavior, i.e., it is not homogeneous, the symmetry of the transition matrix is broken in the semiclassical walk family. This can be useful for ranking nodes in graphs with symmetric transition matrices, where both classical and quantum walks yield the uniform distribution. Due to the asymmetry in the semiclassical walks, these can converge to non-trivial distributions, which can yield information about the graph. Therefore, we can devise a kind of semiclassical PageRank algorithm, which is able to rank nodes on symmetric weighted graphs. These results contrast with the ones of the continuous quantum time version, where all the semiclassical matrices are symmetric, so no ranking can be obtained.

- To demonstrate that the semiclassical walks can be implemented on quantum computers, we have performed some experiments on the IBM Quantum Platform using a real QPU. We have used an asymmetric graph with two nodes so that the limiting distributions are different to the trivial uniform one. We have done the experiments for the first three members of the semiclassical family, which include the classical walk. The results that we have obtained agree incredibly well with the theoretical ones, with a maximum error of 7% due to the errors of current QPUs.

- In following chapters, we apply these semiclassical Szegedy walks to more complex graphs, in the context of the quantum SearchRank algorithm, and signature validation in blockchain. In the future, it would interesting to look for other applications, such us optimization or machine learning, where the Szegedy quantum walk has shown good performance. Moreover, in contrast to quantum walks, the position of the walker can be measured at intermediate time steps. This could be crucial in algorithms that require knowing the position not only at the end of the walk.

Chapter 7

Quantum SearchRank in the Semiclassical Framework

The PageRank algorithm was a revolution in the field of search engines for surfing the Internet [141–144]. This algorithm is able to rank pages objectively, taking into account the structure of the network formed by them, and has a multitude of applications [151–157]. In Section 3.10 we introduced this algorithm and its quantization [31, 32], which relies on the Szegedy quantum walk [45].

Classifying the information is not the only task that a search engine performs. It also has to search for the pages of interest, providing them to the user. Since there exist quantum algorithms for searching problems that outperforms classical ones, as for example the Grover algorithm [26, 27], in 2014 a new quantum algorithm that integrates the quantum search into the quantum PageRank was devised. It was dubbed quantum SearchRank, and it was the first quantum algorithm able to search the nodes of interest at the same time that provides a ranking for them [102]. Moreover, it was the first algorithm implementing a Szegedy quantum walk with queries to an oracle, with a quadratic speedup. Later, different formulations of Szegedy quantum walks with queries were analyzed in the field of quantum search [101].

The quantum SearchRank algorithm seems a promising tool for a future quantum search engine. However, it has some problems that need to be solved. One of them is that the search functionality seems to break down when the size of the network is large enough, so that the nodes of interest are not found correctly. In this thesis, we propose a semiclassical approach for the quantum SearchRank [4]. In addition, we provide a simplification to speed up the semiclassical walk, giving rise to a quantum algorithm that we refer to as randomized SearchRank. This algorithm performs a quantum search from a mixed state, which has been previously studied in the context of the Grover algorithm [199, 200]. As we will show throughout this chapter, we are able to measure the nodes of interest with a high probability regardless of the size of the problem, at the same time that the time complexity of the quantum SearchRank is preserved. Another issue in the quantum SearchRank algorithm is a lack of statistical analysis about the performance of the ranking functionality, which we tackle properly in this thesis. We will show that the SearchRank algorithms provide a ranking compatible with the classical PageRank, and thus our randomized SearchRank is useful for sampling this distribution with a quadratic speedup. Finally, we shall also analyze how the damping parameter of the PageRank algorithms affects to the SearchRank, obtaining a maximum threshold for our semiclassical approach

to be useful.

This chapter is structured as follows. In Section 7.1 we review the formulation of the quantum SearchRank algorithm, and introduce the semiclassical framework in this context. In Section 7.2 we apply the SearchRank algorithms to an example of scale-free graph. In Section 7.3 we focus on the searching feature of the SearchRank algorithms, analyzing the time complexity and the amplification of the probability. In Section 7.4 we focus on the ranking feature of the SearchRank algorithms. In Section 7.5 we study the dependence with the damping parameter inherent to the PageRank algorithm. Finally, we summarize and conclude in Section 7.6.

7.1. The SearchRank Algorithms

In this section we describe the quantum SearchRank algorithm [102], to later introduce the semiclassical framework, giving rise to our new two algorithms: the semiclassical SearchRank and the randomized SearchRank.

7.1.1. Quantum SearchRank

The quantum SearchRank is an extension of the quantum PageRank described in Section 3.10. Let us recall how the transition matrix for the Szegedy quantum walk is obtained. Given a directed network with N pages P_i, we define the patched connectivity matrix E as:

$$E_{i,j} := \begin{cases} 1/\text{outdeg}(P_j) & \text{if } j \in B_i, \\ 1/N \quad \text{if outdeg}(P_j) = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{7.1}$$

where B_i is the set of nodes linking to node i, and $\text{outdeg}(P_j)$ is the outdegree of node P_j. This matrix is mixed with another matrix $\mathbf{1}$ where all the entries are equal to 1, obtaining the Google matrix G:

$$G := \alpha E + \frac{(1-\alpha)}{N}\mathbf{1}. \tag{7.2}$$

Whereas for the PageRank algorithm the value of α is set to 0.85, for the quantum SearchRank algorithm the Google matrix is constructed with $\alpha = 0.25$ without further ado [102]. In section 7.5 we examine further the effect of the value of this parameter, so that we can provide a justification.

At the core of quantum search algorithms there is the oracle operator O_f [26, 40]. This operator, described in Section 2.3.6, computes a boolean function $f(x)$ for each node of the network. In the case of a quantum search engine, it would take the index x of the node, read the information of the page from a database, and check if it satisfies the search condition introduced by a user. Recall that we can use it to construct a phase-flip oracle Q_f, which marks the corresponding computational basis states inverting their sign, so that

$$Q_f |i\rangle := \begin{cases} -|i\rangle & \text{if } i \in \mathcal{M}, \\ |i\rangle & \text{otherwise,} \end{cases} \tag{7.3}$$

where $\mathcal{M}$ is the set of marked nodes.

In the case of the quantum SearchRank algorithm, inspired by the Grover algorithm [26, 27], an oracle that marks the nodes in the first register of the Szegedy Hilbert space is introduced in the quantum walk. Thus, we define an oracle operator for the first register as:

$$Q_1 := Q_f \otimes \mathbb{1}_N. \tag{7.4}$$

This oracle is introduced in the single-step unitary evolution operator U_s between the swap S_w and the reflection R, obtaining U_Q:

$$U_Q := S_w Q_1 R. \tag{7.5}$$

Although this operator was originally formulated like this, given the block-diagonal structure of the reflection R, it turns out that it commutes with the oracle, so that we can express the evolution operator as $U_Q = S_w R Q_1$. Therefore, it resembles the quantum walk operator that reproduces the Grover search algorithm on the complete graph with loops, as shown in Section 3.9. Note that for $\alpha = 0$, the Google matrix is the uniform one, $\mathbf{1}/N$, which indeed corresponds to the Grover walk on the complete graph with loops. Therefore, the parameter α interpolates between the pure Grover algorithm and the quantum walk on the raw graph.

As in the quantum PageRank algorithm, the unitary evolution must be applied an even number of times, so the actual unitary evolution operator is $W_Q := U_Q^2$. The initial state of the system is also constructed as the equal superposition of the $|\psi_i\rangle$ states in (3.31):

$$\left|\Psi^{(0)}\right\rangle := \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\psi_i\rangle \,. \tag{7.6}$$

After the quantum evolution, the second register is measured to obtain the instantaneous SearchRank distributions at each time step:

$$S_q(P_i, t) := \left|\left|\, {}_2\langle i| \, W_Q^t \left|\Psi^{(0)}\right\rangle \right|\right|^2 . \tag{7.7}$$

Whereas in the quantum PageRank algorithm the different time distributions are averaged to obtain the ranking of the nodes [31, 32], in this case we are only interested in the distribution where the probability of measuring the marked nodes is amplified. It was shown that the optimal time for measuring is approximately $t \approx \sqrt{N/M}$ [102]. Later on we examine this scaling more carefully.

In contrast to the quantum PageRank, which is also useful as a quantum-inspired algorithm, the SearchRank is intended to be run on quantum computers, since we are not interested in the total probability distribution but rather in sampling the distribution of the marked nodes after amplifying them. The quantum circuit would be constructed from the one of the quantum walk, adding the oracle Q_f on the first register. The same as for the quantum PageRank implementation, we expect that for sparse graphs, as scale-free networks, an efficient implementation of the reflection R is possible. With regard to the oracle, its circuit depends on the particular function $f(x)$ marking the nodes, and it is in general an open problem. Therefore, so far we cannot study properly the implementation complexity of the SearchRank algorithm. For this reason, we study the time complexity only in terms of the walk complexity, i.e., the number of time steps, as usual in the quantum walks literature (see Section 3.7).

7.1.2. Semiclassical and randomized SearchRank algorithms

In this thesis, we propose a semiclassical version of the quantum SearchRank algorithm, replacing the underlying Szegedy quantum walk by a semiclassical walk of class II (see Section 6.2), since the results are obtained by measuring the second register.

If we substitute the general unitary operator U_s by the SearchRank operator W_Q in (6.12), we obtain the semiclassical transition matrices ${}_2G^{(t_q)}$ for the semiclassical SearchRank algorithm. Each of these matrices can be treated as a Google matrix, so that their stationary distributions give us an instantaneous semiclassical SearchRank distribution for each value of the quantum time t_q. Let us represent these distributions as column vectors and denote them as $S_{sc}(t_q)$, so that they satisfy the following matrix fixed-point equation:

$$ {}_2G^{(t_q)} S_{sc}(t_q) = S_{sc}(t_q). \tag{7.8}$$

From an operational point of view, we have to perform each of these semiclassical walks on an initial probability distribution until they converge. An example is shown in Figure 7.1 (upper panel). In this case, the initial distribution is the uniform one. After the initialization, we perform the quantum evolution for the particular value of the quantum time t_q, measure the second register and, restart for the following classical step. The process is repeated t_c^* times, which is defined as the number of classical steps required for the semiclassical walk to converge. Finally, the outcome of the last measurement is used to sample the semiclassical SearchRank distribution.

Since the searching functionality of the algorithm is in the quantum evolution, there is a value of the quantum time t_q for which the probability of measuring the marked nodes is maximum. As in the case of the quantum SearchRank, the time complexity is close to $t_q \approx \sqrt{N/M}$, as is proved in section 7.3.2. Thus, this algorithm maintains the same complexity with respect to the quantum time evolution. Nevertheless, the semiclassical walk requires repeating the quantum evolution the number of classical steps t_c^* required to converge. Thus, the actual number of times that the operator W_Q is called is $t_q \times t_c^*$.

For the simulations performed in this thesis, we have not found a significant scaling of the value t_c^* with the size N of the graph, so it seems that it grows very slowly, as in the classical PageRank. Nevertheless, since the bigger graph we have used has only $N = 1024$ nodes, the actual scaling of the classical time remains an open problem. Therefore, it could worsen the time complexity of the semiclassical algorithm with respect to the quantum one.

In order to overcome this issue, we propose a simplification fixing the number of classical steps. In particular, we analyze the extreme case where only a single classical step is carried out. This simplification is shown in Figure 7.1 as a blue dashed box. Recall that since the quantum circuit must be repeated to sample the final distribution, in order to initialize the classical distribution we must prepare one of the proxy states $|\psi_i\rangle$ in (6.9) at random each repetition. Formally, this corresponds to the preparation of the uniform mixed state

$$ \rho^{(0)} := \frac{1}{N} \sum_{i=0}^{N-1} |\psi_i\rangle \langle\psi_i| . \tag{7.9}$$

Thus, in the extreme case where only one classical step is performed, the algorithm is equivalent to a quantum walk on the randomized mixed state instead of the equal superposition (7.6), as stated by

Theorem 6.4. Therefore, we shall refer to it as the randomized SearchRank algorithm. The quantum circuit scheme is very similar to that of the quantum SearchRank shown in Figure 7.1 (bottom left panel). In both algorithms the quantum time t_q corresponds directly to the total time of the walk.

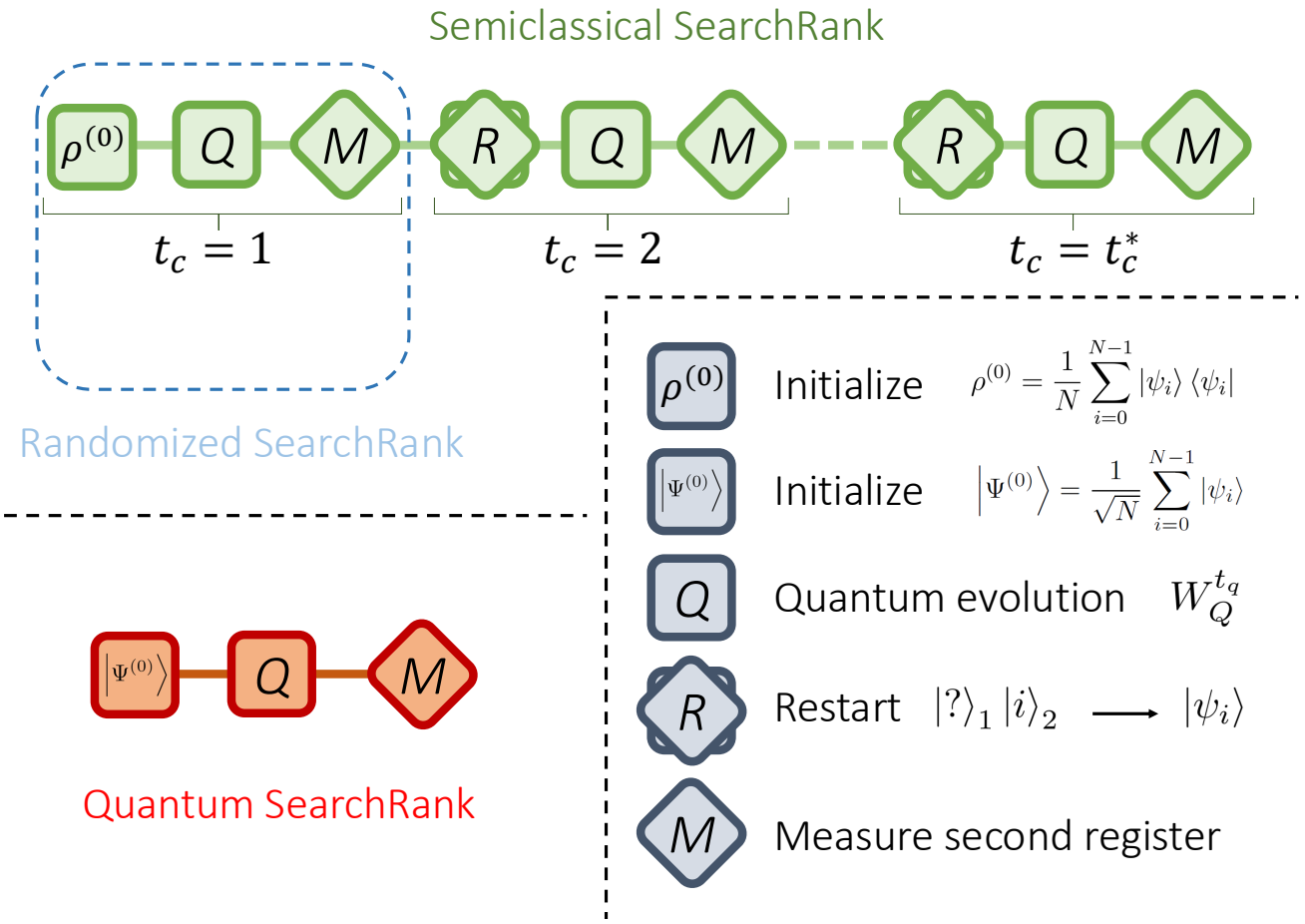


Figure 7.1: *Quantum circuit diagrams of the SearchRank algorithms. In the semiclassical SearchRank (upper panel in green), the first step consists of the initialization of the mixed state $\rho^{(0)}$, the quantum evolution of t_q times the unitary operator W_Q, and a measurement in the second register. After that, each classical step consists of a restart of the system depending on the previous measurement, the quantum evolution, and a measurement. In total, t_c^* classical steps are carried out until convergence. The blue dashed box represents the randomized SearchRank, which is a simplified semiclassical algorithm with only one classical step. In the quantum SearchRank (bottom left panel in red), the initial state $|\Psi^{(0)}\rangle$ is prepared, the quantum evolution is performed and the system is measured. The right dashed box is a legend explaining the meaning of the different elements (quantum circuits) from which the SearchRank algorithms are constructed. In particular, notice that the restart operation is a combination of unitary and non-unitary evolution.*

7.1.3. Numerical simulations

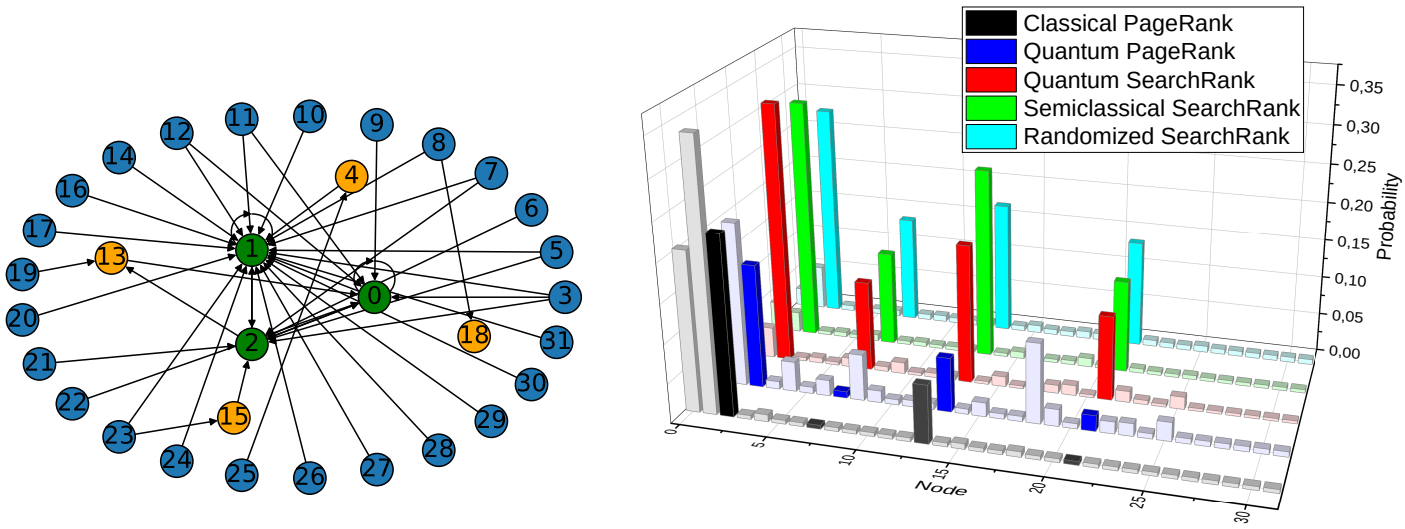
Although the SearchRank algorithms are intended to be run on quantum computers, to study them nowadays we need to use a classical simulator, since fault-tolerant quantum computers are not yet available. For the simulations of these algorithms, we have used our optimized algorithm shown in

Chapter 9, whose time complexity scales as $\mathcal{O}(N^2)$ for the quantum walks, and as $\mathcal{O}(N^3)$ for the semiclassical walks.

In order to simulate the action of the oracle operator Q_f, we do not need to know how the actual function $f(x)$ works. We can just construct a diagonal operator with eigenvalue 1 for unmarked nodes, and -1 for marked nodes, thus acting as in equation (7.3). To do so, we need to choose beforehand a set of nodes for which the algorithm must search, and then we can check if the algorithm succeeds in finding them.

7.2. Example on a Scale-Free Graph

In this section, we look at an example of the SearchRank algorithms using a scale-free graph. As previously mentioned in this thesis, this type of graphs are not only good models for the World Wide Web [182], but also have a wide range of applications such as in neural networks [183], metabolomics [184, 185] and finances [186].



(a) (b)

Figure 7.2: *(a) Scale-free network with $N = 32$ nodes. The inner (green) nodes correspond to the main nodes. The middle (orange) nodes correspond to secondary nodes. The outer (blue) nodes correspond to residual nodes without links pointing to them. (b) PageRank and SearchRank distributions of the network in (a). The marked nodes (2, 7, 13 and 21) have a highlighted color. In the three SearchRank algorithms the marked nodes have an amplified importance.*

We use the same instance of scale-free network with $N = 32$ nodes of Chapter 5, which was constructed with the Python library NetworkX [10], and whose graph is shown in Figure 7.2(a). Due to the way the network is constructed [188], the first nodes (inner green) have the most links pointing to them. Therefore, they are expected to be the most important in the classical PageRank distribution. The middle orange nodes are secondary nodes that have few internal links. Finally,

the outer blue nodes are residual nodes, which lack links pointing to them, and will have a minimal degenerate classical PageRank score. As for the quantum PageRank, this algorithm breaks the degeneracy of the residual nodes, so that some of them may receive higher importance than the secondary nodes [1, 32]. The classical and quantum PageRank distributions are shown in Figure 7.2(b).

For the SearchRank algorithms we have marked four nodes, namely, node 2, which is one of the most important, the secondary node 13, and the residual nodes 7 and 21. The probability of finding one of these nodes at each value of the quantum time is shown in Figure 7.3 for the three SearchRank algorithms we have considered in this thesis. In this case, it is maximum at $t_q = 3$ for the randomized SearchRank, while it is maximum at $t_q = 2$ for the quantum and semiclassical algorithms. In all cases the probability is greater than 0.7, so there are many possibilities to measure them. Note the difference with the baseline of the PageRank algorithms, where the probability is around 0.3, and therefore it is more probable to measure an unmarked node. Nevertheless, this is only one example. In a later section, we analyze the probability achieved by the SearchRank algorithms for graphs of increasing size and different number of marked nodes, as well as the quantum time complexity of the algorithms.

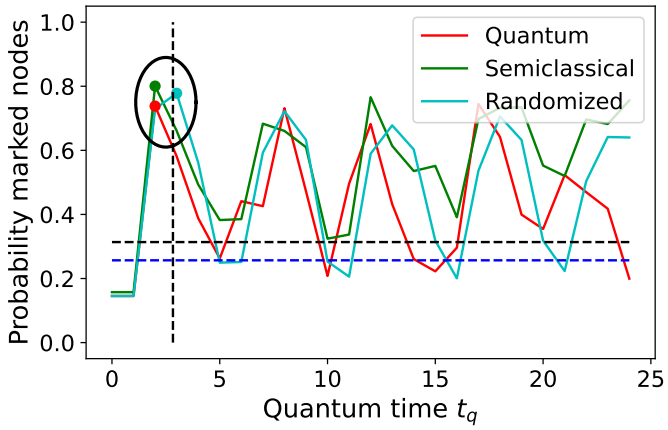


Figure 7.3: *Probability of measuring one of the marked nodes versus the quantum time for the three SearchRank algorithms applied on the scale-free graph with $N = 32$ nodes of Figure 7.2(a). The first maximum of each curve is marked with a dot, and they are surrounded by a circle. The vertical dashed line represents the value $\sqrt{N/M}$. The horizontal dashed lines represent the probability of the marked nodes in the classical (black) and quantum (blue) PageRank distributions.*

As we can see in Figure 7.2(b), the four marked nodes have the greatest ranking in the three SearchRank distributions, so that their probability has been effectively amplified. In order to compare the ranking of the marked nodes with respect to the PageRank distributions, we have isolated their distributions in Figure 7.4. Since the probability of the marked nodes has been amplified in the SearchRank algorithms, we represent the SearchRank distributions on a different scale from the PageRank distributions. The most important of the marked nodes, both in the classical and quantum PageRank, is node 2, and this one has been properly detected as the most important node by the three SearchRank algorithms. The second node in importance is node 13, which is a secondary node of the network. Again, the three SearchRank algorithms detect it properly as the second most important node. So far the order of importance is maintained. However, for nodes 7 and 21 there is a violation of the order. In all the SearchRank algorithms, node 7 is more important than node

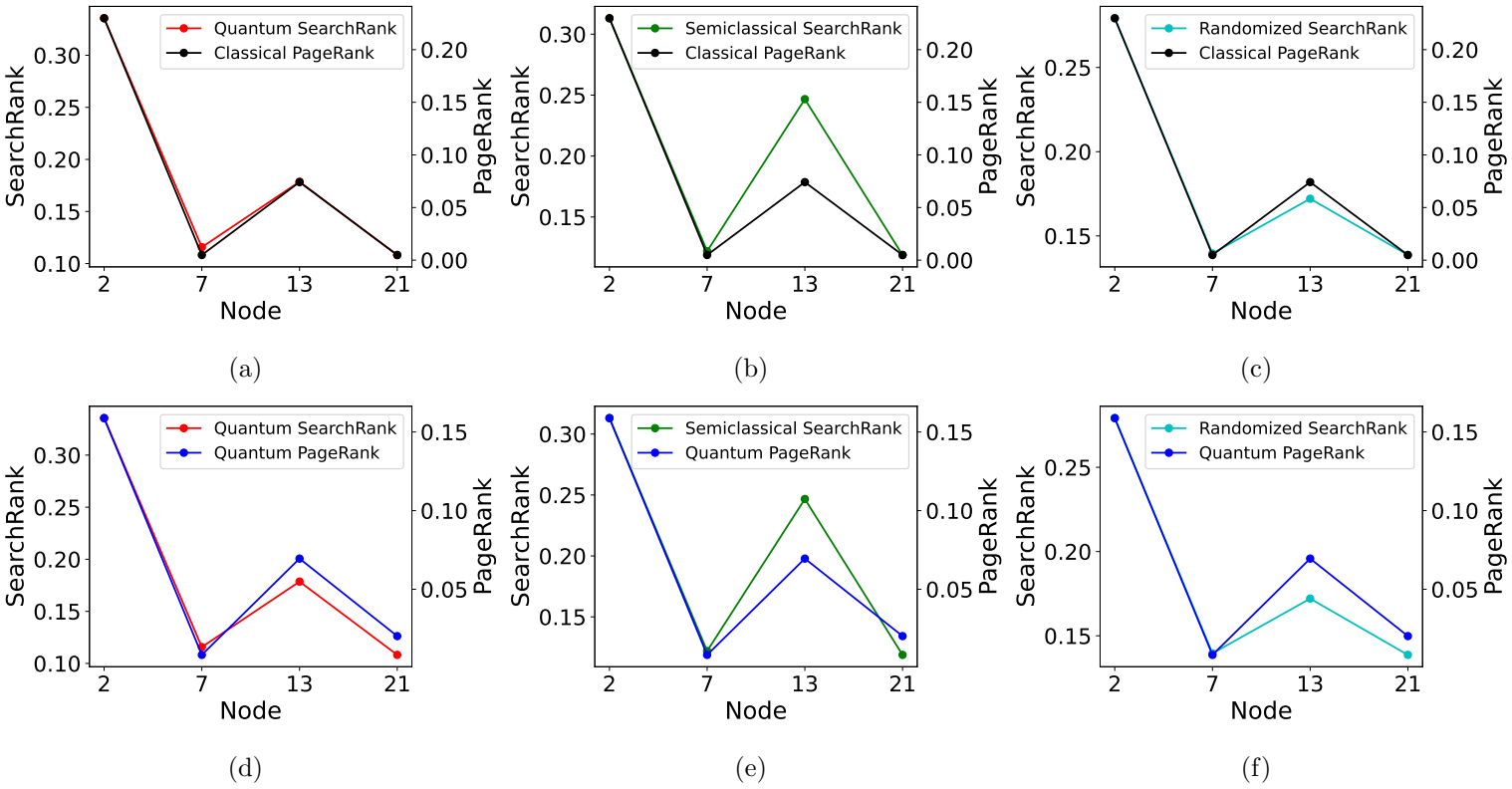


Figure 7.4: *Comparison of the SearchRank distributions of the marked nodes with the PageRank distributions for the network with $N = 32$ nodes of Figure 7.2(a). (a)-(c) Comparison with the classical PageRank. (d)-(f) Comparison with the quantum PageRank. The PageRank (right axis) is represented on a different scale from the SearchRank (left axis).*

21, while in the classical PageRank they are degenerate. Nevertheless, in the case of the randomized SearchRank they are almost degenerate, as can be seen in Figure 7.4(c). In the quantum PageRank distribution, node 21 is more important than node 7, so the order is reversed.

In this thesis, we have previously discussed that the quantum PageRank can introduce fluctuations in the classically degenerate node order that can be misleading (see Section 5.3). Thus, the SearchRank algorithms are expected to introduce fluctuations that may be different, and it is not uncommon for the residual node order to be reversed with respect to the quantum PageRank. Therefore, in general it seems that the three SearchRank algorithms correctly rank the nodes. In a later section, we study statistically the agreement between the PageRank and SearchRank distributions for a large set of different scale-free networks and marked nodes.

7.2.1. Visualizing the semiclassical search

As done in Chapter 6, we can use the semiclassical transition matrix to represent the semiclassical walk as a weighted graph. This allows us to visualize how the probability of the marked nodes in the search process is amplified. In Figure 7.5(a) we have represented the weighted network whose weights are given by the Google matrix (7.2) using $\alpha = 0.25$. As expected, we can observe a large flow of information to the main nodes of the graph, and some flow to the secondary nodes. Thus,

this classical walk gives a higher rank to these nodes in the limiting distribution, as shown in Figure 7.5(c), where the probability of each node at each classical step is plotted. When we perform the semiclassical walk from this graph with $t_q = 2$, it is equivalent to a classical walk whose weighted graph is shown in Figure 7.5(b). We can now observe that the flow of information is mainly directed to the marked nodes in red, and therefore, there will be a high probability of measuring them in the limiting distribution. As shown in Figure 7.5(d), the probability of the marked nodes converges rapidly to an amplified value, while the rest of the nodes adopt a residual classification. Therefore, the semiclassical algorithm modifies the classical network to search for the marked nodes.

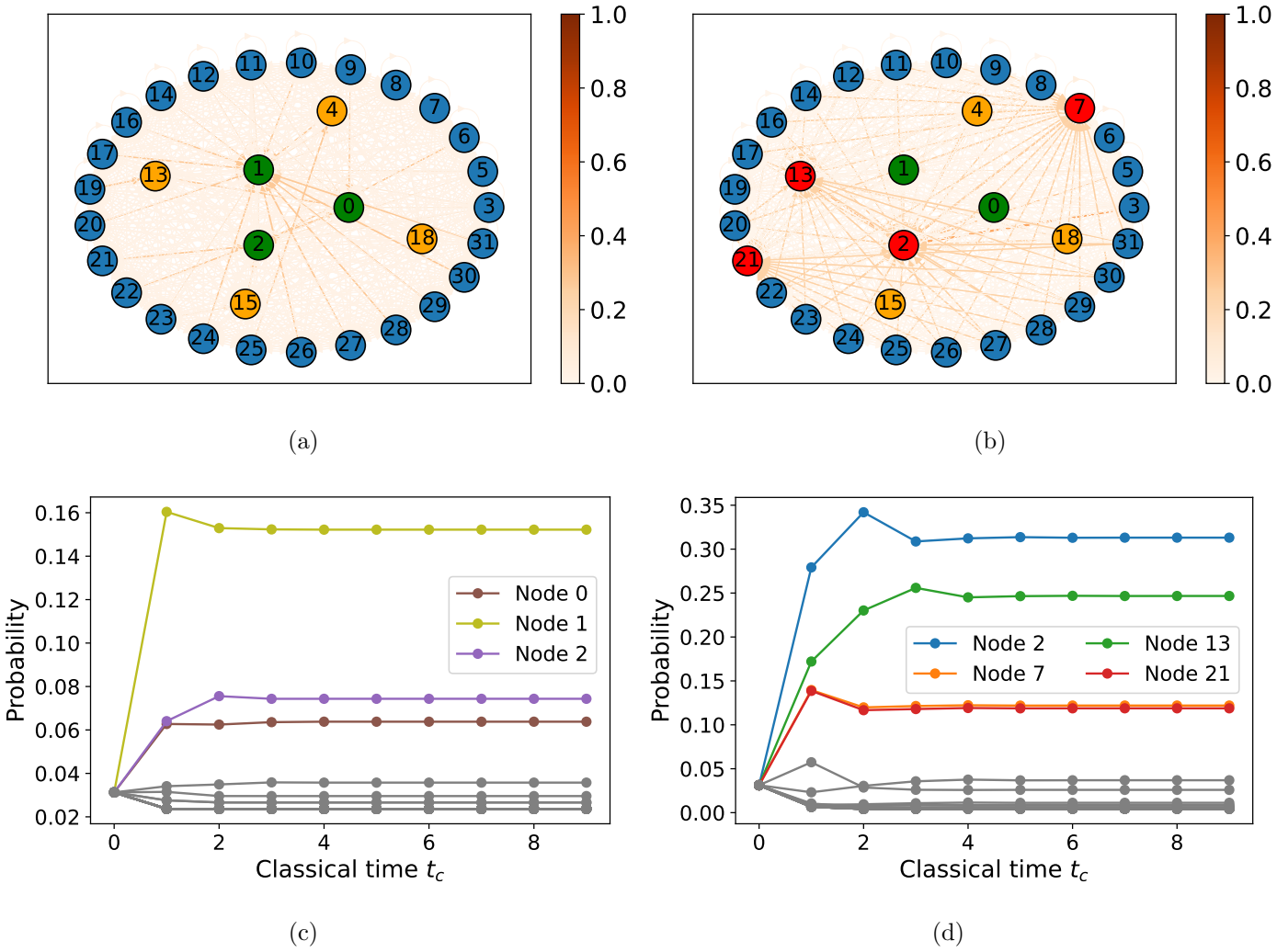


Figure 7.5: *(a) Weighted graph representing the Google matrix with $\alpha = 0.25$ for the network with $N = 32$ nodes of Figure 7.2(a). The strongest links point to the main nodes in green. (b) Weighted graph representing the semiclassical matrix of the semiclassical SearchRank for $t_q = 2$. In this case the strongest links point to the marked nodes in red. (c) Probability of each node versus the classical time for the classical walk represented in (a). The probability attains a higher value for the main nodes. (d) Probability of each node versus the classical time for the semiclassical walk represented in (b). The probability of the marked nodes is amplified whereas the other nodes obtain residual SearchRank scores.*

7.3. Searching Power of the SearchRank Algorithms

One of the two virtues of the SearchRank algorithms is their ability to amplify the probability of measuring the marked nodes. In this section we analyze what is the probability that each algorithm can achieve, and what is the time complexity to reach the maximum probability.

Table 7.1: *Parameters of the scaling law* (7.10) *for the fits to the measurement probability data of the marked nodes and quantum time complexity.*

Function	Parameter	Quantum	Semiclassical	Randomized
Optimal time	n	0.455 ± 0.016	0.523 ± 0.009	0.473 ± 0.008
	A	1.19 ± 0.08	0.91 ± 0.03	1.05 ± 0.03
Optimal probability	n	-1.046 ± 0.035	—	—
	A	11.45 ± 1.90	—	—
Probability reference time	n	-1.109 ± 0.078	—	—
	A	6.20 ± 2.29	—	—

7.3.1. Probability at the maximum

In the previous example, all the SearchRank algorithms were able to extend the probability of the marked nodes above 0.7. However, in this section we show that the probability at the maximum drops with N/M for the quantum algorithm, while it remains at a high value for the semiclassical and randomized SearchRank algorithms. Although here we deal with the actual maximum of the probability curves, in a later section we will show that it also remains at a high value for the reference measurement time $t_q = \left\lfloor \sqrt{N/M} \right\rceil$.

On the one hand, to show that the quantum probability drops with the network size N, we simulated the SearchRank algorithms for scale-free random graphs with $N = 64$, 128, and 256 nodes, all three with $M = 6$ randomly chosen marked nodes. The probability curves with respect to the quantum time are shown in Figures 7.6(a)-7.6(c). We can observe how indeed the maximum point of the curve corresponding to the quantum algorithm reaches a lower value as the size of the graph N increases. On the other hand, in order to analyze the effect of the number of marked nodes, we have simulated the algorithms for three scale-free random graphs with $N = 512$ nodes, and $M = 24$, 12, and 1 marked nodes. The corresponding probability curves are shown in Figures 7.6(d)-7.6(f). In this case, the probability of measuring a marked node in the quantum SearchRank decreases as the number M of nodes decreases.

Now that we have demonstrated the qualitative relationship between the quantum probability and the N and M parameters of the problem, let us obtain a quantitative relationship. To do so, we have simulated the SearchRank algorithms for different scale-free random networks with sizes ranging from $N = 64$ to $N = 1024$. For all networks we have chosen $M = 1$, 3, 6, 12, 24, and 48 nodes at random. Since some results of quantum search problems, such as the Grover algorithm, depend directly on the N/M ratio [19,40,103], we have plotted the maximum likelihoods with respect to N/M in Figure 7.7.

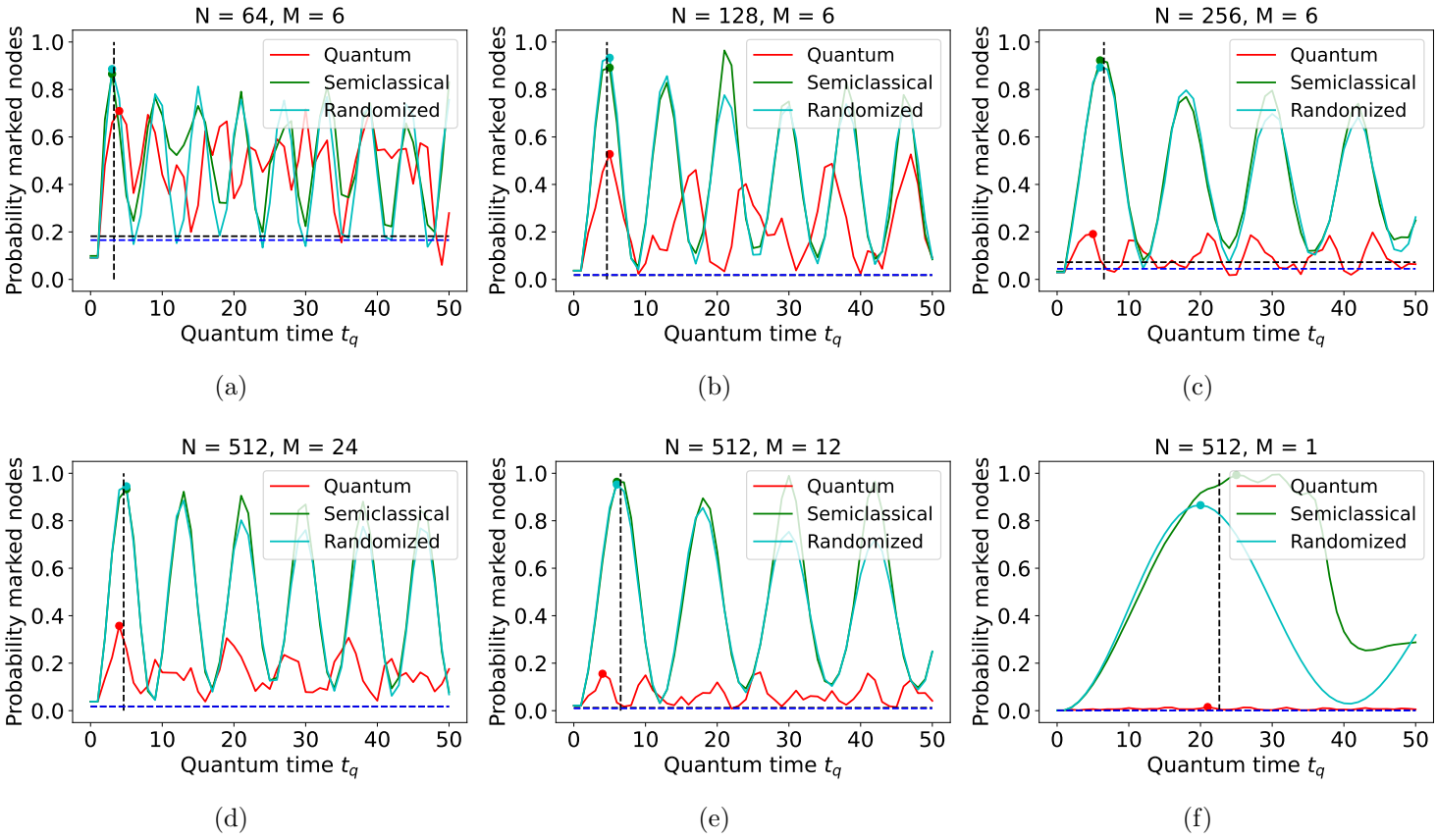


Figure 7.6: *Probability of measuring one of the marked nodes versus the quantum time for the three SearchRank algorithms on different scale-free graphs. (a) On a graph with $N = 64$ nodes and $M = 6$ marked nodes. (b) On a graph with $N = 128$ nodes and $M = 6$ marked nodes. (c) On a graph with $N = 256$ nodes and $M = 6$ marked nodes. (d) On a graph with $N = 512$ nodes and $M = 24$ marked nodes. (e) On a graph with $N = 512$ nodes and $M = 12$ marked nodes. (f) On a graph with $N = 512$ nodes and $M = 1$ marked node. The first maximum of each curve is marked with a dot. The vertical dashed line represents the value $\sqrt{N/M}$. The horizontal dashed lines represent the probability of the marked nodes in the classical (black) and quantum (blue) PageRank distributions.*

In the case of the quantum SearchRank, in Figure 7.7(a) we can observe that the probability drops rapidly with N/M as expected. To obtain a mathematical expression, we intend to fit the data to the following scaling law:

$$f(N/M) = A\left(\frac{N}{M}\right)^n . \tag{7.10}$$

We can linearize this expression by taking logarithms:

$$\log f(N/M) = \log A + n \log\left(\frac{N}{M}\right). \tag{7.11}$$

We have plotted the same data in logarithmic scale in Figure 7.7(b). Although there is an initial region where the probability remains very high, from $N/M \approx 20$ there is a clear linear relationship. A linear fit in this asymptotic region yields an exponent $n = -1.046$, so that the quantum probability falls asymptotically as approximately $\mathcal{O}(M/N)$. The parameters of the fit are summarized in Table 7.1.

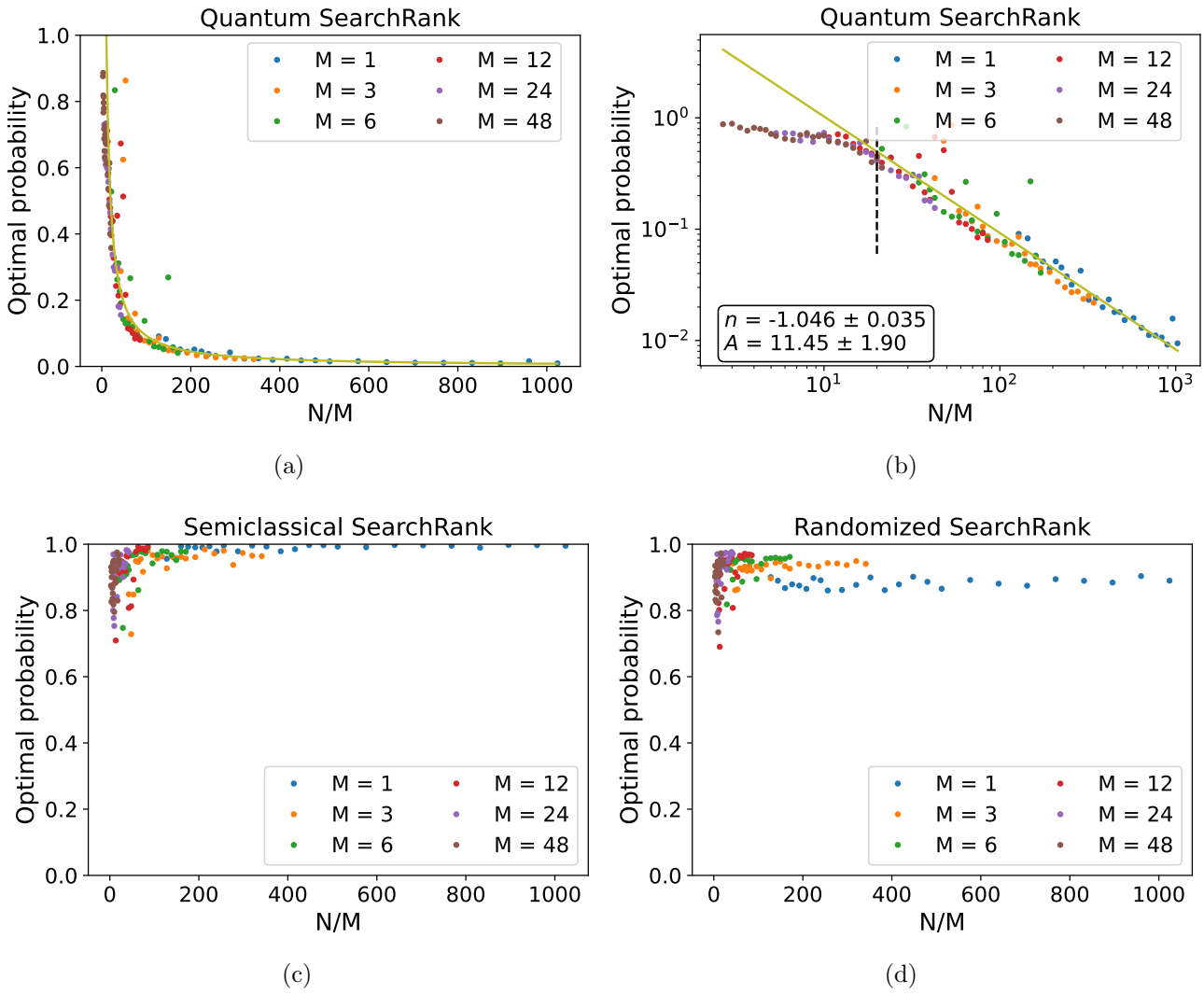


Figure 7.7: *Probability at the maximum versus the relation N/M for (a) the quantum SearchRank, (b) the quantum SearchRank in logarithmic scale, (c) the semiclassical SearchRank, and (d) the randomized SearchRank. The vertical dashed line in (b) indicates the region from which the linear fit has been done.*

For the semiclassical SearchRank algorithm, in Figure 7.7(c) we can see that in the asymptotic region the maximum probability is close to 1. Finally, in Figure 7.7(d) the maximum probability for the randomized SearchRank is represented. In this case, the probability is a little lower but still close to 1. Table 7.2 shows the average probability achieved in the asymptotic region. It seems that the results do not depend on the number of marked nodes M, as expected, and the probability remains around 0.9. We can conclude, therefore, that the semiclassical framework solves the problem with the probability of the quantum SearchRank algorithm, even after simplification.

Table 7.2: *Average probability of measuring one of the marked nodes in the asymptotic region for the semiclassical and randomized SearchRank algorithms and different number M of marked nodes. Both measuring at the optimal time and at the reference time $t_q = \left\lfloor \sqrt{N/M} \right\rfloor$, showing that a high probability around 0.9 is obtained. Errors correspond to one standard deviation.*

M	Optimal		Reference time	
	Semiclassical	Randomized	Semiclassical	Randomized
1	0.99 ± 0.00	0.89 ± 0.01	0.97 ± 0.01	0.85 ± 0.02
3	0.97 ± 0.01	0.94 ± 0.01	0.94 ± 0.05	0.91 ± 0.01
6	0.97 ± 0.01	0.95 ± 0.02	0.95 ± 0.03	0.94 ± 0.02
12	0.94 ± 0.07	0.94 ± 0.05	0.93 ± 0.07	0.93 ± 0.05
24	0.95 ± 0.03	0.96 ± 0.02	0.95 ± 0.03	0.96 ± 0.02
48	0.92 ± 0.04	0.93 ± 0.04	0.92 ± 0.04	0.93 ± 0.04

7.3.2. Time complexity

Despite the fact that we have shown how the semiclassical SearchRank is able to amplify effectively the probability of measuring the marked nodes, the question that arises is whether the time complexity is similar to that of the quantum algorithm. In this section, we examine the quantum time scaling for which the maximum probability occurs.

Again, we expect the time value of the maximum to depend on the N/M ratio. Thus, we have plotted these time values with respect to N/M in Figures 7.8(a)-7.8(c) for the three SearchRank algorithms. As expected, there is a clear relationship with the N/M quantity. To fit the data to the scaling (7.10), we have plotted the same data in logarithmic scale in Figures 7.8(d)-7.8(f). The results of those fits are summarized in Table 7.1. The linear fits give an exponent of $n = 0.455$ for the quantum SearchRank, $n = 0.523$ for the semiclassical SearchRank, and $n = 0.473$ for the randomized SearchRank. As for the prefactor A, we obtain $A = 1.19$ for the quantum SearchRank, $A = 0.91$ for the semiclassical SearchRank, and $A = 1.05$ for the randomized SearchRank. Thus, we have found that the optimal measurement point in the semiclassical algorithm and its simplification is close to $t_q = \sqrt{N/M}$. In the case of the quantum SearchRank, it seems to be slightly faster than the others. Nonetheless, it should be noted that the data fluctuates a lot. This is mainly due to the difficulty of identifying the maximum when the probability is very small. Therefore, we can conclude that the semiclassical framework does not worsen the time complexity of the quantum algorithm, being relatively the same.

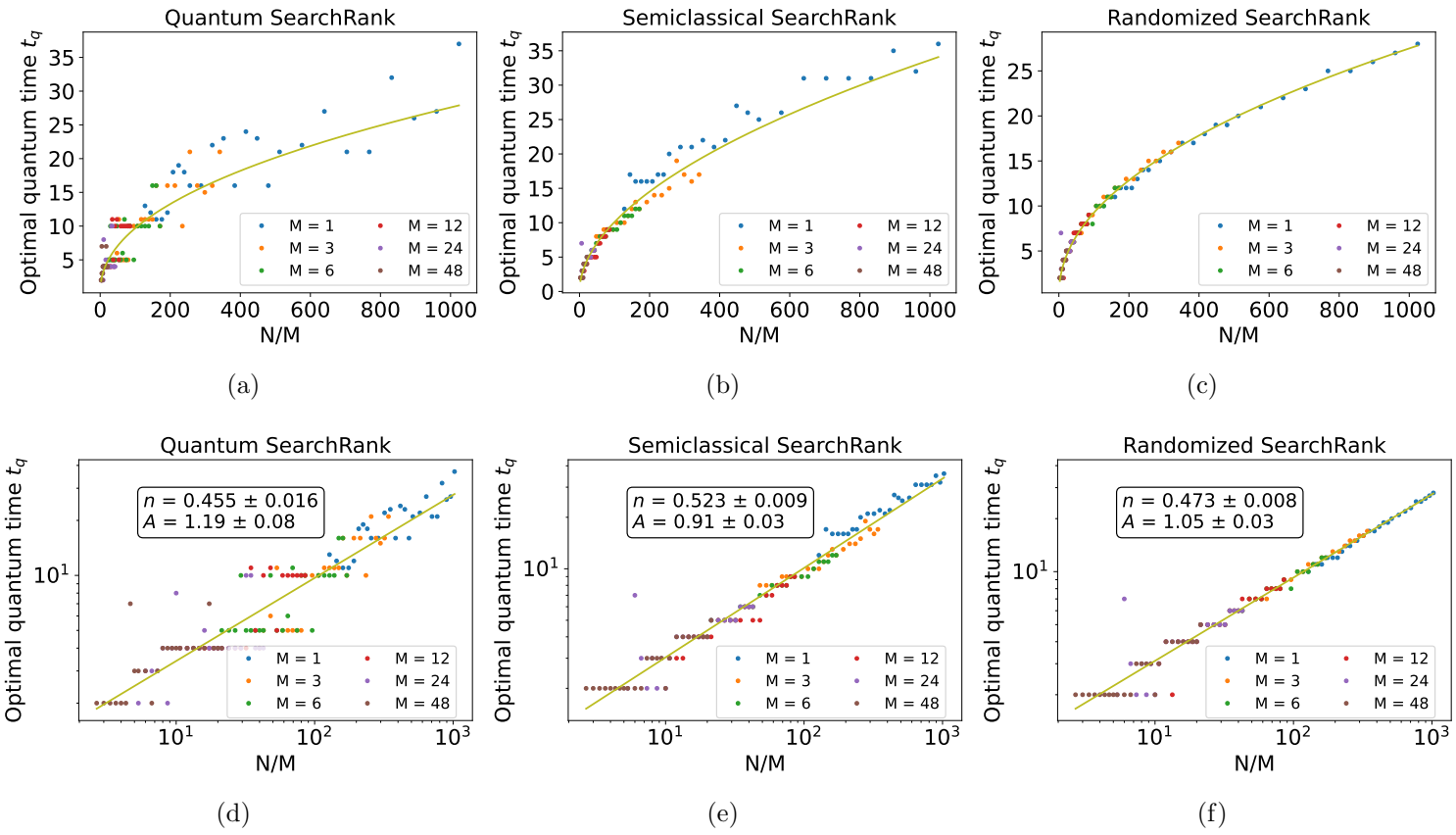


Figure 7.8: *Quantum time for which the maximum of probability occurs versus the relation N/M for (a) the quantum SearchRank, (b) the semiclassical SearchRank, and (c) the randomized SearchRank. The corresponding data for the linear fit is represented in (d)-(f) in logarithmic scale.*

7.3.3. Reference time of measurement

So far, we have observed that the semiclassical and randomized SearchRank algorithms are able to find the marked nodes with a high probability at maxima. However, to measure with this probability, we would need to know *a priori* how many quantum steps are needed to reach the maximum. Although we have seen that the optimal value of the quantum time is close to $\sqrt{N/M}$, we cannot be sure where the maximum is since it depends on the particular network.

As a reference, we can always measure to the nearest integer $t_q = \left\lfloor \sqrt{N/M} \right\rceil$, expecting to be close to the optimal point. Therefore, we need to analyze what is the real probability when measuring at this reference time. For this purpose, we have plotted in Figure 7.9 the probability of measuring the marked nodes at the reference time. The probability in the quantum algorithm falls as expected, with a relationship with N/M similar to the previous one. The parameters of the fit are summarized in Table 7.1. For the semiclassical and randomized algorithms the probability is very similar to the maximum. Therefore, we still measure the marked nodes with a very high probability. The average probability achieved by the semiclassical and randomized SearchRank algorithms is summarized in Table 7.2.

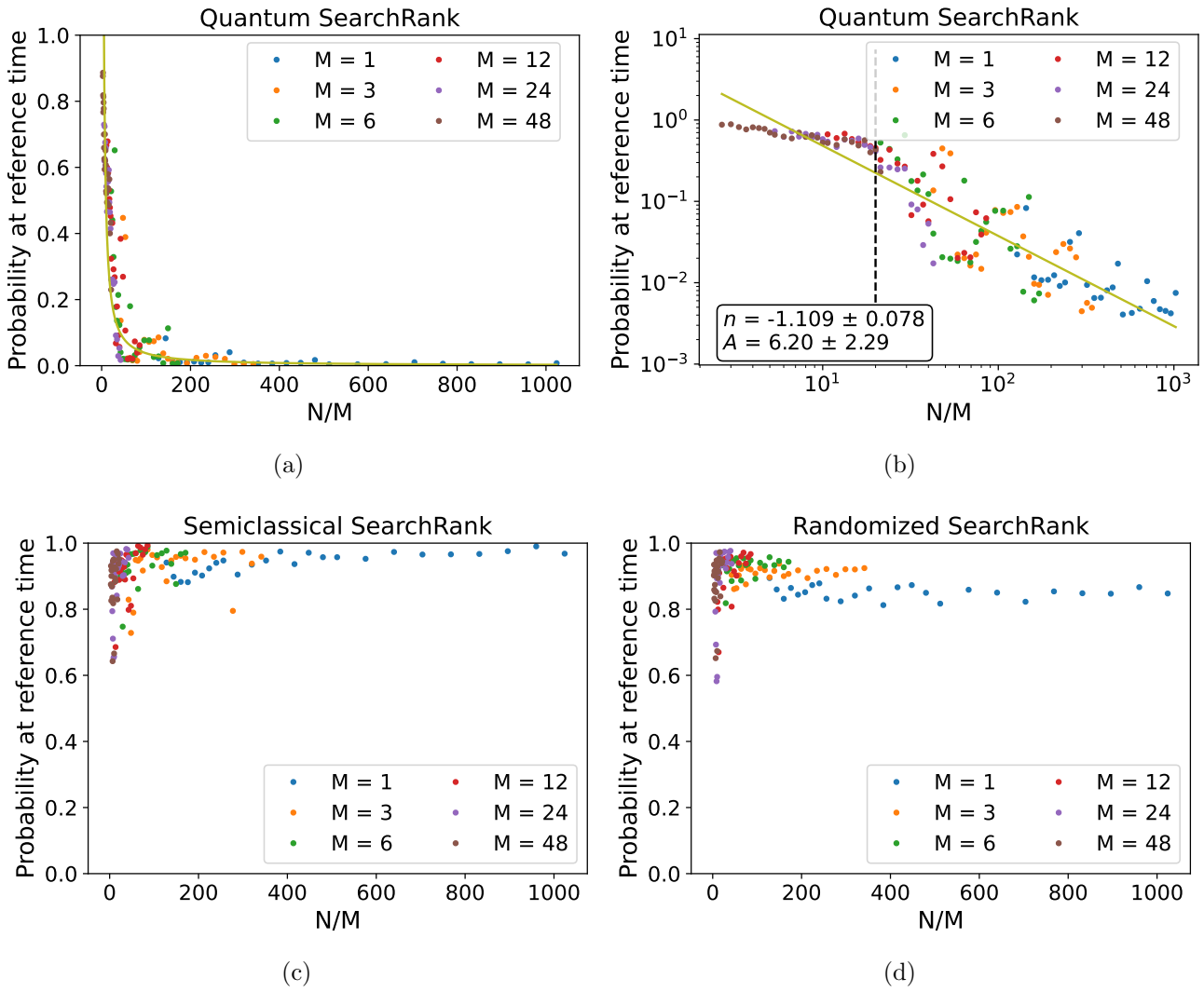


Figure 7.9: *Probability at the reference time of measurement* $t_q = \left\lfloor \sqrt{N/M} \right\rfloor$ *versus the relation* N/M *for (a) the quantum SearchRank, (b) the quantum SearchRank in logarithmic scale, (c) the semiclassical SearchRank, and (d) the randomized SearchRank. The vertical dashed line in (b) indicates the region from which the linear fit has been done.*

We have shown that the semiclassical SearchRank is an algorithm capable of performing effective quantum search on scale-free networks that the quantum algorithm does not perform. With simplification as a randomized quantum walk, we do not have to worry about the classical time to convergence, thereby the time complexity is similar to that of the quantum algorithm. Thus, we have devised an algorithm that can be useful for searching problems with a quadratic speedup with respect to classical algorithms, regardless of its ability to classify nodes.

7.4. Ranking Power of the SearchRank Algorithms

The second feature of the SearchRank algorithms is their ability to rank the marked nodes according to their importance. Since the randomized SearchRank is a type of semiclassical SearchRank, the

natural question arises as to which algorithm it resembles the most in terms of ranking, the classical or the quantum PageRank. In other words, whether it is more classical or more quantum with respect to this property. In this section, we are going to compare the ranking provided by the three SearchRank algorithms with the ranking given by both the classical and quantum PageRank. For this purpose we are going to use the Kendall coefficient [201]. It is used to measure the similarity between two ordered lists of items. The Kendall coefficient is 1 if both lists are equal, -1 if the order is totally reversed, 0 if there is a total absence of correlation, and takes intermediate values depending on the partial correlation.

First, let us look at the value of the Kendall coefficient when $M = 48$ nodes are marked. We have plotted this metric for all graphs of different size N in Figure 7.10(a) for comparison with the classical PageRank, and in Figure 7.10(b) for the quantum PageRank. The first thing we notice is that the results do not depend on the size of the network N. Compared to the classical PageRank, the three SearchRank algorithms have a Kendall coefficient of around 0.6. Since this coefficient is in the interval $[-1, 1]$, it means that there is good agreement in the ranking of the marked nodes compared to the ranking of the classical PageRank. However, when compared with the quantum PageRank, this coefficient has a small value, around 0.15. Therefore, the correlation with the quantum PageRank, although positive, is very weak. As for the best matching SearchRank algorithm, in both cases it seems to be the randomized SearchRank. Nevertheless, the differences between the three algorithms are practically negligible.

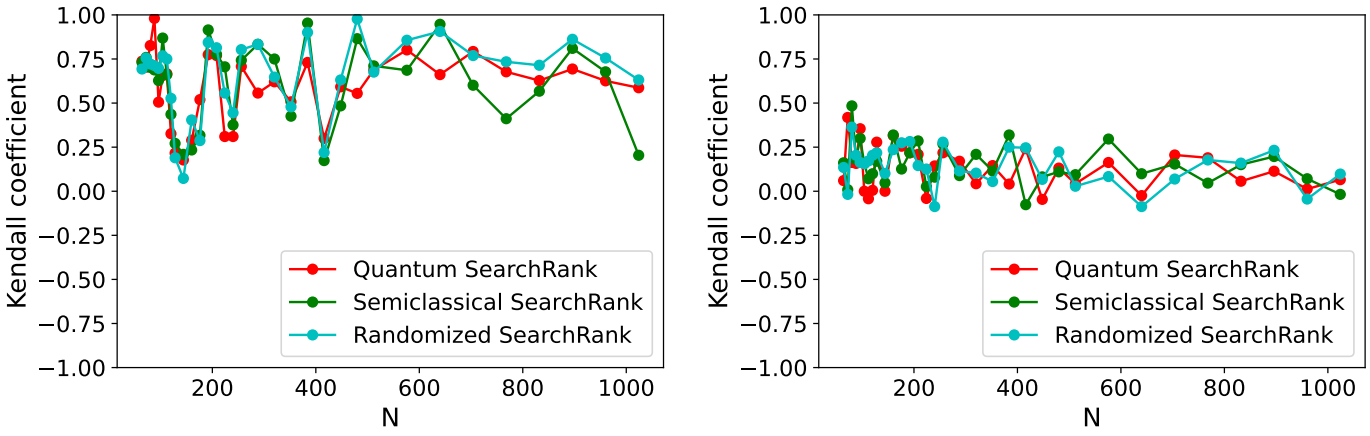


Figure 7.10: *Kendall coefficient versus graph size N for a ranking of $M = 48$ marked nodes by the three SearchRank algorithms compared to (a) the classical PageRank and (b) the quantum PageRank.*

Now, we want to ensure that these results hold for any number M of marked nodes. We have averaged the Kendall coefficient for all networks of different size N and plotted it for each value of M, in Figure 7.11(a) for the comparison with the classical PageRank, and in Figure 7.11(b) for the quantum PageRank. As expected, in all cases the Kendall coefficient is larger in the comparison with the classical PageRank, and there is little correlation with the quantum PageRank for all numbers of marked nodes. This may be due to the fluctuations introduced by the quantum PageRank and SearchRank algorithms, so that nodes with similar importance easily change their ranking between the different algorithms. Let us now look at the results for different values of M. For $M = 12$, 24, and 48 there is little difference between the three SearchRank algorithms in the comparison with the

classical PageRank, and the results are similar for the three values of M. Nonetheless, for $M = 3$ and $M = 6$ larger differences are observed, with higher values of the coefficient for the randomized SearchRank. In the comparison with the quantum PageRank all the results are almost similar.

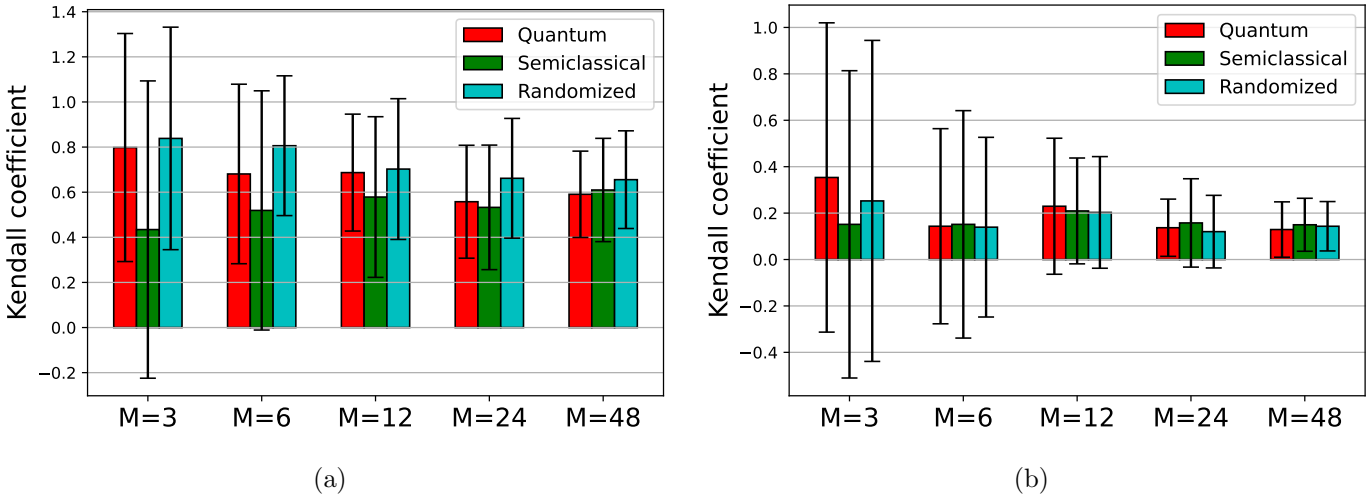


Figure 7.11: *Mean Kendall coefficient for all graphs of different size N in the comparison of the ranking of different number M of marked nodes by the three SearchRank algorithms with (a) the classical PageRank and (b) the quantum PageRank. Error bars correspond to one standard deviation.*

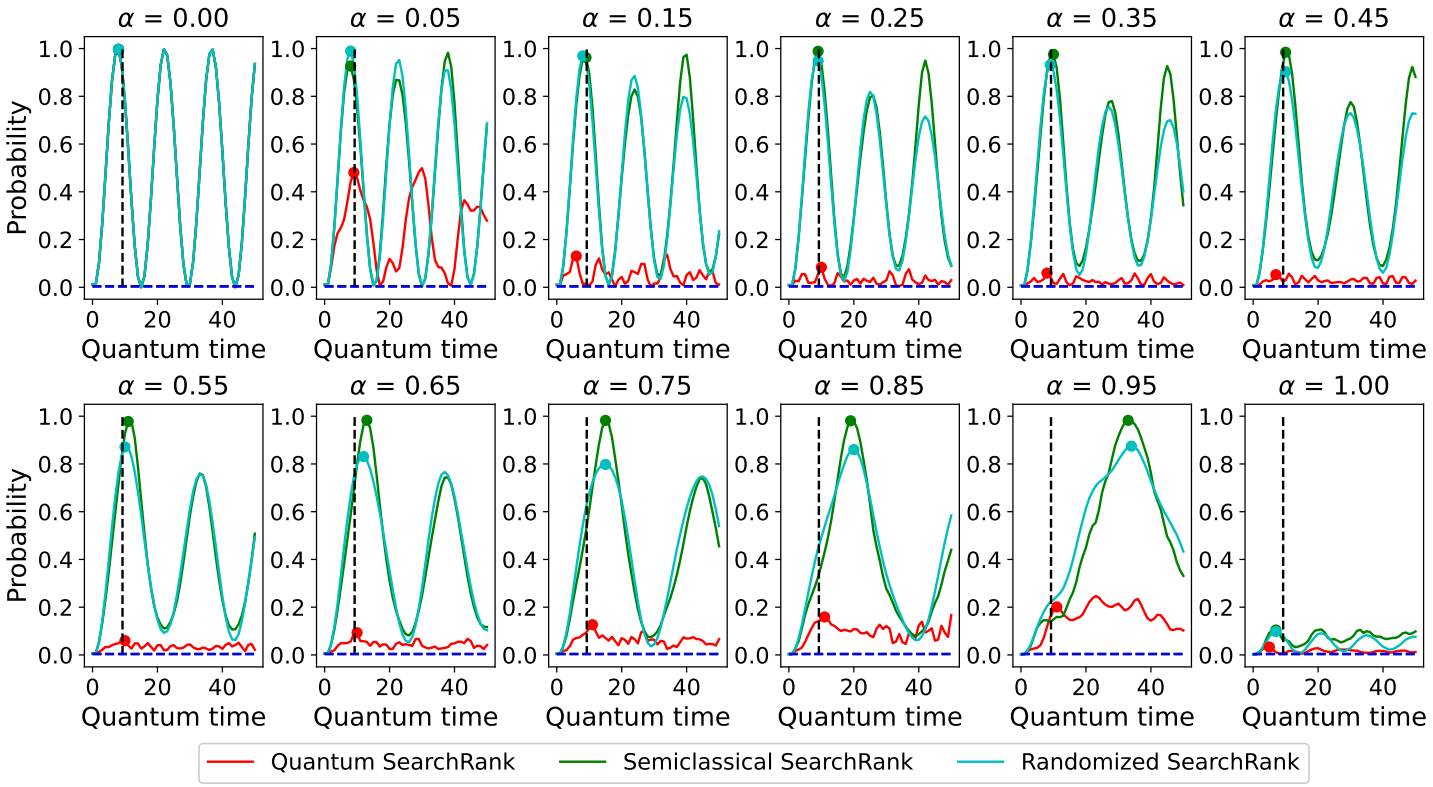
It turns out that the Kendall coefficient is quite unstable for small lists. For example, for $M = 3$ it is easy to get perfect agreement, since there are only 3 nodes. However, it is also easy to have perfect disagreement. Thus, due to the fluctuations introduced by the quantum evolution in the SearchRank algorithms, we have very different values for each network. This explains the large error bars for small values of M.

In summary, although our semiclassical and randomized SearchRank algorithms are not able to sample the quantum PageRank distribution of the marked nodes, they have good agreement with the classical PageRank. Therefore, these algorithms can be used to sample the marked nodes in a network from a probability distribution that is related to the classical PageRank, taking advantage of quadratic quantum acceleration.

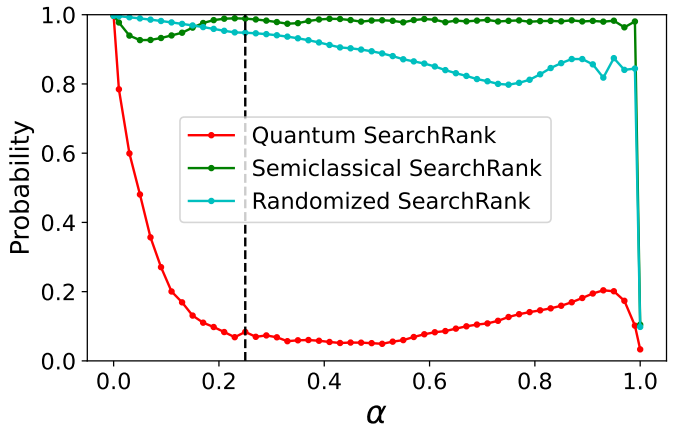
7.5. Dependence with the Damping Parameter

In Section 7.1.1 we stated that the quantum SearchRank used $\alpha = 0.25$ for the construction of the Google matrix in (7.2). However, the question arises as to what happens if we use another value of α. In this section we briefly show some simulation results of the SearchRank algorithms for different values of α. We have used a scale-free network with $N = 512$ nodes and $M = 6$ marked nodes.

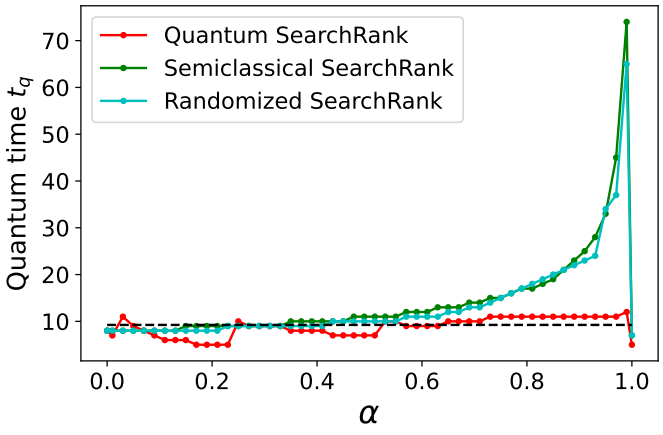
In Figure 7.12(a) we have plotted the probability of measuring the marked nodes versus the quantum time for different values of the parameter α from 0 to 1. Let us first analyze the probability at the first maximum, shown in Figure 7.12(b). For $\alpha = 0$, when the network is actually the fully connected uniform network given by matrix $\mathbf{1}$ in (7.2), the probability at maximum reaches 1 for the



(a)



(b)



(c)

Figure 7.12: *(a) Probability curves of measuring one of the marked nodes versus the quantum time for different values of the parameter α for the three SearchRank algorithms applied on a graph with $N = 512$ nodes and $M = 6$ marked nodes. The first maximum of each curve is marked with a dot. The vertical dashed line represents the value $\sqrt{N/M}$. The horizontal dashed lines represent the probability of the marked nodes in the classical (black) and quantum (blue) PageRank distributions. (b) Probability achieved at the maximum versus the parameter α. The vertical dashed line represents the value $\alpha = 0.25$. (c) Optimal quantum time for the measurement versus the parameter α. The horizontal dashed line represents the value $\sqrt{N/M}$.*

three SearchRank algorithms. Nevertheless, as soon as α takes a non-zero value, the probability in the quantum SearchRank starts to drop dramatically. There is a subtle recovery from $\alpha = 0.5$, but the probability remains below 0.2 and eventually drops to 0.03. In the semiclassical SearchRank, the probability at the maximum remains close to 1 for all α except $\alpha = 1$, when the network is just the scale-free network previous to the mixing made in (7.2). The randomized SearchRank also maintains a high probability, but drops slowly to 0.8 in the worst case. Finally, for $\alpha = 1$ the probability is almost zero for the quantum SearchRank, and around 0.1 for the semiclassical and randomized SearchRank algorithms.

It might seem that there could be better values than $\alpha = 0.25$ for the semiclassical algorithms, since a large α value means a closer resemblance of the Google G matrix to the original network. However, let us look at the quantum time required to reach the maximum, shown in Figure 7.12(c). For the quantum SearchRank, the maximum is reached at approximately the same time, which is compatible with the reference value $t_q = \sqrt{N/M}$. Nonetheless, for the semiclassical and randomized SearchRank, the optimal time increases dramatically from $\alpha = 0.6$. Thus, we cannot increase the value of α much for these algorithms even though the probability remains high. We have found similar results for different examples. Nevertheless, more rigorous analysis is needed in the future.

As previously discussed, for $\alpha = 0$, the Google matrix corresponds to the transition matrix of the Grover walk on the complete graph with loops. As shown in Section 3.9, with the addition of the oracle, the Grover search algorithm is recovered in both registers, as long as they are initialized in equal superpositions of the computational basis. Although in the semiclassical framework the system is initialized in a mixed state formed by the $|\psi_i\rangle$ states, each of them has in the second register the equal superposition $|s\rangle$ in the second register for $\alpha = 0$. Therefore, since in the SearchRank algorithms the second register is measured, the Grover search algorithm is also recovered for the semiclassical and randomized SearchRank. Thus, the three probability curves overlap, reaching practically a maximum probability of 1. For $0 < \alpha < 1$, the algorithm interpolates between the pure Grover search, and the search on the raw graph. Whereas the quantum algorithm loses its efficacy as soon as it separates from the Grover algorithm, the semiclassical approaches are more resistant to this effect. Nevertheless, the fact that when $\alpha = 1$ the probability of the semiclassical SearchRank drops so drastically suggest that mixing the graphs is essential, needing at least a residual contribution of the pure Grover algorithm. This highlights that the construction of the Google matrix is a valuable technique not only for the PageRank algorithm, but also for search algorithms with quantum computers. This could lead to new quantum search algorithms inspired by PageRank.

7.6. Summary of Results and Conclusions

- The quantum SearchRank algorithm is able to expand the probability of measuring a given set of marked nodes, while providing a ranking of these nodes by importance. However, the probability of measuring the marked nodes decreases as the size of the graph increases, so the algorithm loses its utility. In order to resolve this issue, we have proposed a semiclassical approach, resulting in the semiclassical SearchRank algorithm. Although this algorithm solves the probability problem, it has a longer running time than the quantum SearchRank due to a combination of quantum and classical steps until it converges. Thus, we have proposed a simplification with a single classical step, called randomized SearchRank, since the underlying

walk is equivalent to the quantum walk but with a mixed initial state. Therefore, this new algorithm maintains a similar running time to the original quantum SearchRank.

- We have analyzed the performance of the three SearchRank algorithms on a relatively small scale-free network. We found that the SearchRank algorithms are able to amplify the probability of the marked nodes, so that there is a high probability of measuring one of these nodes each time the algorithm is run. Since the graph is small enough, the quantum SearchRank does not suffer from probability depletion. Furthermore, the probability of each of the marked nodes produces a ranking with good agreement with the classical and quantum PageRank distributions. However, there is a violation in the ranking of the residual nodes due to the fluctuations introduced by the quantumness of the algorithm, which drastically affect the ranking of nodes with a very small difference in importance. From this example network, we have been able to visualize how the semiclassical SearchRank works. In this case, the underlying semiclassical walk behaves like a classical walk on a graph in which the information flow is redirected to the marked nodes, so that the probability of these nodes is amplified in the asymptotic distribution.

- To obtain statistical results on the SearchRank algorithms, we have simulated them on a large set of graphs of increasing sizes and different sets of randomly marked nodes. On the one hand, we have checked how the probability of measuring marked nodes in the quantum SearchRank collapses as the size of the graph N increases and/or the number of marked nodes M decreases, with an asymptotic scaling of approximately $\mathcal{O}(M/N)$. This depletion means that the quantum SearchRank loses the ability to amplify the amplitude of the marked nodes, so it is not a successful search algorithm. Nevertheless, we have also shown that in the semiclassical SearchRank and the randomized SearchRank this problem is solved, so that the probability does not decrease, remaining at a high value above 0.9. On the other hand, we have studied the time complexity of the SearchRank algorithms in terms of the quantum time t_q. In all cases we have obtained a scaling law approximately compatible with $\mathcal{O}(\sqrt{N/M})$. Since we cannot know *a priori* what the exact value of the optimal quantum time is, we have decided to take $t_q = \left\lfloor \sqrt{N/M} \right\rceil$ as a reference value for the measurement, having shown that the probability in the semiclassical and randomized SearchRank remains at a high value around 0.9 despite not being optimal.

- Regarding the ranking capability of the SearchRank algorithms, we used the Kendall coefficient to measure the similarity between the rankings provided by the PageRank and SearchRank distributions. We have observed that the three SearchRank algorithms yield similar results, obtaining a fairly good agreement with the classical PageRank. Nonetheless, the agreement with the quantum PageRank is very low, so there seems to be a large lack of correlation. This may be due to the fluctuations introduced by the quantum PageRank, and also by the SearchRank algorithms, so that nodes that are similar in importance can easily have their order changed, and thus the correlation is lost.

- Finally, we have studied the dependence of the SearchRank algorithms on the damping parameter α. In the case of the quantum SearchRank, the probability of measuring the marked nodes collapses rapidly as soon as α takes a non-zero value. This explains why in the quantum SearchRank a value of $\alpha = 0.25$ was taken [102] instead of the value of $\alpha = 0.85$ used in the PageRank algorithms [31]. In the case of the semiclassical SearchRank, the probability remains close to 1, except for α values close to 1. The same is true for the randomized SearchRank,

although the probability drops a little while maintaining a value above 0.8. Although the probability is high in these last two algorithms, the maximum probability shifts to the right as α increases, so the execution time becomes much higher. Therefore, we cannot increase the value of α arbitrarily, and $\alpha = 0.25$ seems to be a good value for the new SearchRank algorithms.

- Taking all the results together, the randomized SearchRank solves the probability problem of the quantum SearchRank while maintaining the same time complexity. This algorithm is able to provide one of the marked nodes with a quadratic speedup and a probability that is directly related to the classical PageRank. Thus, it can be used to sample this distribution without the need to calculate it exactly. Moreover, like the quantum SearchRank, it is not necessary to average the results at different time steps of the walk, so it is much faster to implement than the quantum PageRank. It therefore constitutes a further step towards a quantum search engine. Even though the novel randomized SearchRank stems from the semiclassical SearchRank, formally the only difference with the quantum SearchRank is that the initial state is a mixed state rather than a quantum superposition of all the $|\psi_i\rangle$ states. It is interesting to see how the introduction of this mixedness at the beginning of the algorithm allows the probability to be amplified appropriately. Furthermore, blending with the complete graph seems to be crucial for the search functionality, so that the algorithm has some proportion of a pure Grover search. These two intriguing features need further research, and could be used as a base tool for future quantum search algorithms on arbitrary graphs.

- In the future, it would be interesting to study other formulations of the unitary Szegedy quantum walk operator with oracles [101] in the context of the quantum SearchRank algorithm, or extensions with arbitrary phase rotations, as done for the quantum PageRank algorithm [1]. In addition, there are some issues that deserve further research. One of them is the fact that we are assuming that we know the number M of marked nodes to search, so that we can estimate the optimal time for the measurement. However, in a real scenario we would not know how many nodes satisfy the search conditions. A possible solution could be to introduce a quantum counting algorithm [103, 113–115] to estimate the number of marked nodes. Another issue is that we would like to be able to sample from the quantum PageRank distribution as well, since that algorithm is expected to provide better results in quantum networks. In all, our randomized SearchRank algorithm is already a valuable technique to quantum speed up fundamental properties of classical networks.

Chapter 8

Quantum Signature Validation in Blockchain

Blockchain technology was proposed in 2008 [202] as a decentralized system operating on a peer-to-peer (P2P) network, enabling direct transactions between users without needing centralized intermediaries or financial institutions. Thus, transactions in the blockchain are validated through a distributed consensus mechanism within the P2P network. Additionally, cryptographic techniques are used to ensure transaction security and privacy. Once transactions are validated, they become verifiable, immutable, and secure on the blockchain [203]. The first operational blockchain system was launched in 2009 with the creation of Bitcoin. Since then, numerous blockchain systems have emerged, including Ethereum (2015) [204] and Polkadot (2020) [205]. The rapid development of blockchain technology has driven innovation across diverse fields, such as medical data management [206], logistics and supply chains [207], and art property management [208].

Despite its rapid deployment, blockchain faces critical challenges in scalability, efficiency, and security [209]. Therefore, quantum computing could be used to improve blockchain systems. Although most of the works are focused on protecting blockchain systems from future quantum attacks to classical cryptography [210, 211], there are also works which aim to use quantum algorithms for speeding up the implementation. For example, the use of the Grover algorithm to optimize the block mining process [212], or quantum algorithms for efficient quantum consensus mechanism [213, 214].

In this thesis, we propose a quantum algorithm for detecting fraudulent activity, which we denote as Quantum Signature Validation Algorithm (QSVA) [6]. This protocol is based on the SearchRank algorithm of Chapter 7, so that can leverage a graph representation of the blockchain system to extract further information than a conventional quantum search algorithm. Our simulation results using a real Bitcoin dataset of transactions show that the QSVA can efficiently detect all the fraudulent activity, providing a speedup with respect to a classical search.

This chapter is structured as follows. In Section 8.1 we review the fundamentals of blockchain technology. In Section 8.2 we present the Quantum Signature Validation Algorithm, designed for the efficient detection of tampered transactions. In Section 8.3 we evaluate the QSVA with classical simulations. Finally, we summarize and conclude in Section 8.4.

8.1. Blockchain

In this section, we review the functioning of blockchain. We start explaining briefly the main fundamentals of blockchain technology [215], and then we focus on digital signatures [216] and the graph structure of transactions [217], which are the basics of our quantum algorithm.

8.1.1. Blockchain technology

In blockchain technology there is a record-keeping system that stores the transaction, known as ledger. In a P2P network, this ledger is shared among all the participants, so that the information of the transactions, such as the amount transferred and the transaction identifiers, is visible to anyone. However, the identities of the participants are typically pseudonymous, linked only to cryptographic addresses. This transparency allows operations on the network to be verified by multiple participants, ensuring trust without relying on a central authority. The ledger is divided into blocks of transactions, as shown in Figure 8.1. Furthermore, these blocks are cryptographically linked, so that the ledger is tamper-evident, i.e., any signs of unauthorized manipulation are immediately revealed [215].

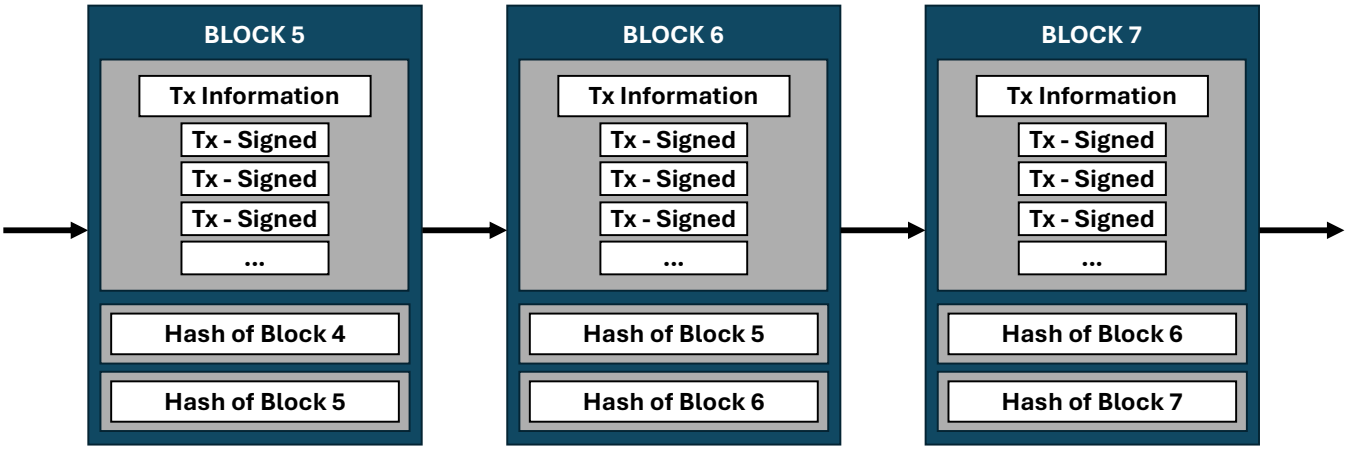


Figure 8.1: *Schematic representation of the blockchain structure. Each block consists of a collection of digitally signed transactions, according to the protocol described in Section 8.1.2, as well as the hash derived from the solved proof-of-work required to add the block to the ledger, and the hash of the proof-of-work from the preceding block.*

In order to chain blocks to the ledger, the system broadcasts each transaction in the P2P network to the participants, who maintain a personal copy of the ledger. Among all the types of participants, we focus on those who validate the transactions, referred to as miners in systems like Bitcoin [202]. Miners receive a list of digitally signed transactions, and store them in a pending register, known as the mempool, prior to their verification. Miners then assess the validity of each transaction in the mempool by employing digital signature processes. If the verification fails, it indicates that the transaction may have been altered or tampered with, potentially by an attacker. In such cases, the miner removes the fraudulent transaction from its mempool, and follows with the next one.

Once the miner has a long enough set of valid transactions, constructs a block to be chained. Since

there are multiple miners creating blocks, there could be problems of double-spending. Moreover, a particular miner could be manipulating the transactions and chain them as if they were valid. Therefore, there must be a consensus mechanism for deciding which blocks are appended to the ledger. There are different mechanism [215], as the proof-of-work used in Bitcoin [202], or the proof-of-stake used in Ethereum [204].

For the sake of simplicity, in this thesis we only review the proof-of-work method. It consists of expending computing resources for solving a difficult task, so that only those blocks for which the problem has been solved can be chained. Honest miners work in groups to create chains of new blocks, and if there are different groups creating different chains, in the end the consensus protocol decides to take the longest chain, discarding the rest. The reason for the proof-of-work is that a malicious miner could create multiple fake identities, and create a chain of blocks with tampered transactions on its own. This is known as sybil attack. Nevertheless, it needs the computational power necessary to solve the proof-of-work of all the blocks, which is expensive and prohibitive for a single party. Therefore, the honest miners forming groups dominate the ledger, as they possess more computational power. Moreover, upon successfully adding blocks to the blockchain, miners receive an economic reward, so that it may be more profitable to work honestly in a common chain than trying to create a different tampered chain [202].

The proof-of-work problem usually consists of calculating a hash of the block. In this sense, a cryptographic hash function [218] is used. A hash function takes some data as input, which can be even as large as a document, and gives a fixed length string of bits. For example, the Bitcoin blockchain uses the SHA-256 hash function [202]. This means that, for (almost) every data given as an entry in the SHA-256, the function returns a string of 256 bits [219]. Furthermore, these functions have properties of preimage resistance and collision resistance [218, 220]. Preimage resistance ensures that, for nearly all specified outputs, it is computationally unfeasible to find any input that produces that output when hashed. In contrast, collision resistance implies that it is computationally unfeasible to find two distinct inputs that generate the same hash output. For the proof-of-work, a numeric parameter in the block is continuously modified until the output of the hash function starts with a determinate number of zeros, so that new blocks are added every 10 minutes approximately [202].

Given the properties of hash functions, if a transaction in the ledger is tampered with, the new hash of the block is different, and it is highly likely that it will not solve the proof-of-work problem, so that the attack is quickly detected. Additionally, to ensure greater security, each block in the ledger is linked to the previous one by the hash associated with the proof-of-work, as shown in Figure 8.1. This means that an attacker would have to redo all the computational work of the blocks that follow the altered block to benefit from their modifications, since a little change in a block produces a change in the hash of all the successive blocks.

To sum up, a diagram representing the blockchain protocol is shown in Figure 8.2.

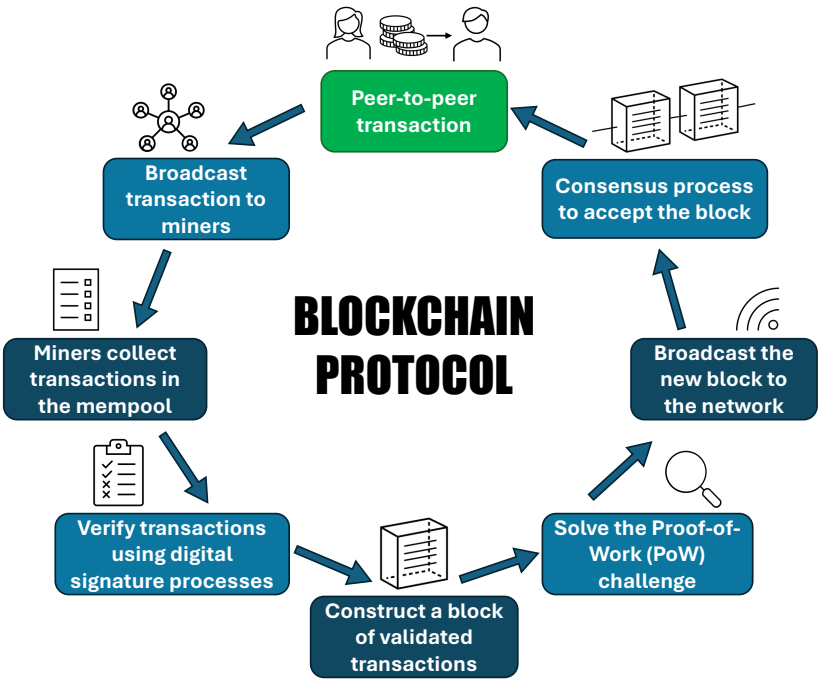


Figure 8.2: *Schematic representation of the blockchain protocol. The starting point is highlighted in green. Each new transaction on the peer-to-peer network is broadcast to all the miners. Each miner maintains a copy of the ledger, which is organized into blocks containing signed transactions. Miners collect incoming transactions in the mempool, verify their validity using a digital signature validation process, and construct a block of validated transactions by solving the proof-of-work challenge. The newly constructed block is then broadcast to the network, where it is accepted through a consensus process. Miners then proceed to work on building the next block in the chain, incorporating the hash of the accepted block as the previous hash in the new block under construction. Each step ensures the integrity, security, and decentralization of the distributed ledger system.*

8.1.2. Digital signature

In order to validate a transaction for adding it to a block, digital signature processes are used [216, 220, 221]. In Figure 8.3 we show the main algorithms involved. First of all, each participant in the network must generate a pair formed by a private key and a public key. Whereas the public key can be known by anyone else, the private key must remain secret to ensure security. These keys are generated in such a way that one of them can encrypt some data, and the other can decrypt it, but none of them can be use solely for both tasks. For example, in asymmetric-key cryptography using the RSA algorithm [30], any sender can encrypt some data with the public key of a particular receiver, and only that receiver can decrypt the data using its private key. However, for digital signature validation, the keys work the other way around. The owner of the private key uses it to encrypt some data, along with some metadata like a timestamp and data of the signatory, generating the signature. The validator receives the data and the signature, and uses the public key to decrypt it. If and only if the decrypted signature matches the original data, then the signature is valid.

In the case of blockchain, the transaction data may be encrypted with the private key of the participant performing the operation. However, encrypting the raw data consumes too much time. For that reason, the data is hashed with a hash function to reduce its size prior to the signature

generation [222]. Recall that due to the properties of cryptographic hash functions, the output of the hash is practically unique for each data input. Therefore, it represents the original data and can be used to create the signature, as shown in Figure 8.3. To validate it, a validator obtains the hash of the raw data, and checks if it matches the signature after decrypting it with the public key.

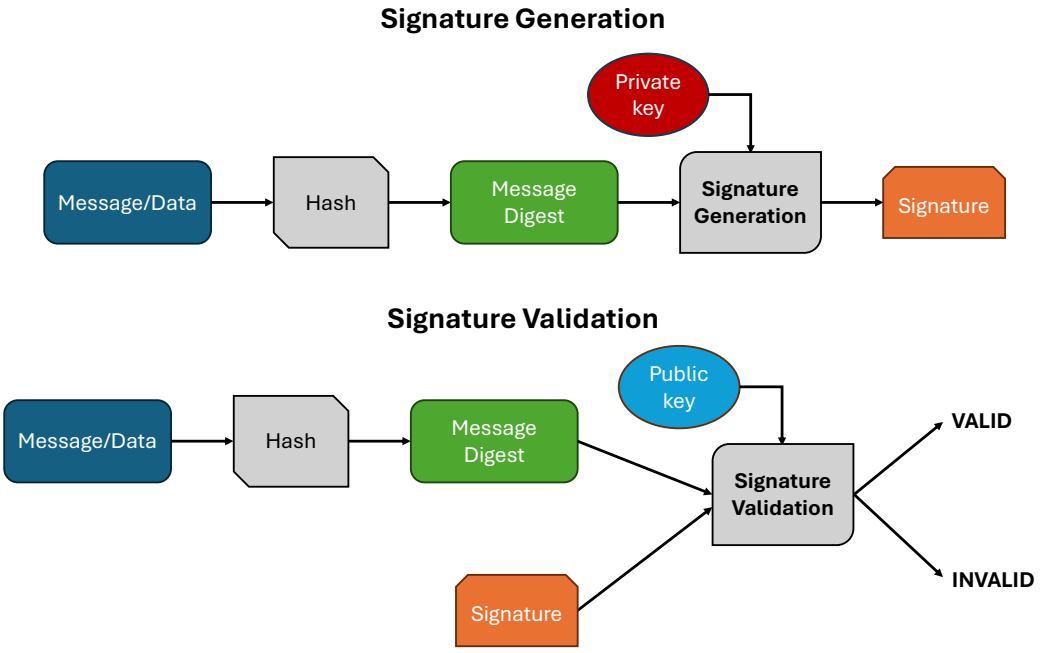


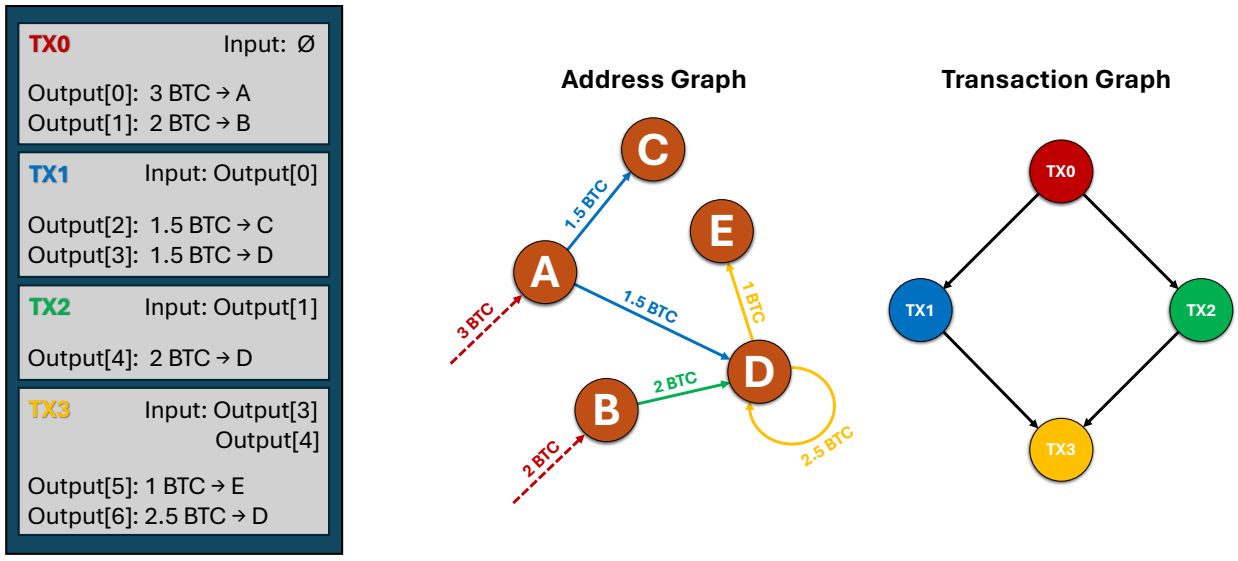
Figure 8.3: *Digital signature processes. In the signature generation process, a hash function is applied to a message or data to obtain a message digest or hashed message. Then, a signature generation algorithm uses the hashed message and the private key of the signatory as inputs to create a digital signature. For signature validation, the hashed message is reobtained and the digital signature is validated with the public key of the signatory, producing a boolean value indicating whether the message has been altered (invalid) or not (valid).*

A common issue about digital signatures is how to ensure that the public keys of the signatories have been transmitted properly to the validators. In this sense, public-key infrastructures may be necessary to ensure the proper distribution of public keys. Otherwise, a malicious participant could tamper the public key of a signatory, so that it matches a different private key that the malefactor owns, and therefore can tamper any transaction so that it will be valid. Nevertheless, this is beyond the scope of this thesis, and we refer to the literature for further information [220].

8.1.3. Graph representation

Given the connections between user addresses and flows of money, an effective approach for extracting useful information from the ledger is using graph representations. Various graph models can be found in the literature, each tailored to the specific type of information of interest [223]. Since our study focuses on analyzing individual transactions, we use the transaction graph representation, where each transaction corresponds to a node of the graph [217].

In order to construct a transaction graph, we use the UTXO (Unspent Transaction Output) model [224], which is used in transaction-centered blockchain systems as Bitcoin. There, each transaction



(a) (b) (c)

Figure 8.4: *(a) Schematic representation of a UTXO-based ledger. Outputs specify the amounts of BTC sent to specific addresses, while inputs reference the outputs of previous transactions. If an input is empty, it indicates the creation of new BTC, such as through the mining process of a block. (b) Address graph associated to the ledger in (a), where the flow of money between different addresses is represented. (c) Transaction graph associated to the ledger in (a), where each transaction corresponds to a node.*

has inputs and outputs. The outputs determine the amount of BTC (the currency of Bitcoin) to be transferred to other addresses, and the inputs are references to previous unspent transactions outputs. Examples of transactions are shown in Figure 8.4(a). The inputs are totally consumed by the transaction, meaning that the sum of the inputs equals the sum of the outputs (plus any change returned to the sender or a fee to a miner). Therefore, unlike account-based systems, there is no need to track all past transactions of an address from the beginning of the ledger to determine its balance. Instead, it is enough to verify that the referenced output in the transaction remains unspent [223, 224].

The flows of money could be represented with an address graph, where each address corresponds to a node, and the directed edges represent the transactions. An example is shown in Figure 8.4(b). Nevertheless, we are interested in the transaction graph, which can be constructed from the address graph. In this case, each transaction is associated with a node. These nodes are connected by directed edges that point from transaction whose output has not been spent, to transactions that use those funds as inputs. An example is shown in Figure 8.4(c). Transaction graphs are widely used for multiple purposes, such as analyzing money flows [225], detecting money laundering transactions [226], and conducting typical network analyses [223].

8.2. Quantum Signature Validation Algorithm

Early detection of tampered transactions in blockchain is crucial. The faster this occurs, the sooner valid transactions can be included in a block, then added to the ledger. In this sense, quantum

computing is a promising paradigm which could speed up the detection of fraudulent transactions, improving the efficiency of blockchain systems.

In order to solve scalability issues of blockchain, we propose a quantum protocol, which we denote as Quantum Signature Validation Algorithm (QSVA). This algorithm aims to enable the rapid detection of tampered transactions, using a quantum search subroutine.

From the transactions in the mempool waiting to be validated, we can construct a database of N elements, where there will be M tampered transactions which we want to detect. As we have seen along this thesis, quantum search algorithms make use of an oracle Q_f, which computes a function $f(x)$ and marks the quantum states of the elements of interest in a database flipping their phase, as explained in Section 2.3.6. In the context of blockchain, we are interested in a function that determines if the signature of a transaction is valid or not. In this sense, the function $f(x)$ would take the index x of the transaction in the database, and use it to read the information about the data of the transaction, the signature and the public key, to perform the signature verification algorithm as shown in Figure 8.3.

In order to perform a quantum search algorithm, we need to know the number M of marked elements. This can be achieved using the quantum counting algorithm [103, 113–115] explained in Section 2.4.1. After that, we could use the Grover algorithm [26, 27, 95, 103] explained in Section 2.4 for finding the tampered transactions. After detecting a marked element, we could construct an artificial oracle that flips the phase of only the states corresponding to the detected tampered transaction, and use it as a filter for the oracle Q_f, so that the phase of the already detected element is reinverted after the application of the oracle and therefore it behave as unmarked. Thus, we could repeat the Grover algorithm with the filtered oracle to find a different tampered transaction. Repeating this procedure $M - 1$ times we could detect all the remaining fraudulent transactions, leveraging the quadratic speedup of the Grover algorithm.

Although this approach would work, we can use other algorithms that leverage the graph structure of transactions to extract further information, and could enhance the search of tampered transactions. For example, the PageRank algorithm has been used in blockchain for both address graphs [227–229] and transaction graphs [217, 226]. The key idea behind the PageRank algorithm is that the importance of a node is determined not only by the number of nodes linking to it but also by the quality of these links (see Section 3.10). Specifically, links from nodes that have few outgoing links contribute more to the importance of the target node. As a result, nodes that receive many incoming links from relevant sources are more likely to rank higher in PageRank. From a transaction graph perspective, attacks that manipulate multiple transactions to redirect their outputs in order to accumulate large amounts of funds result in a node with multiple independent incoming connections in the graph. Consequently, such nodes are likely to receive a high PageRank score, facilitating their identification.

Since the PageRank can be helpful for detecting tampered transactions, we propose the use of an algorithm based on the quantum SearchRank introduced in Chapter 7, instead of the Grover algorithm. As we saw, the best performing algorithm is the randomized SearchRank, so we propose its utilization in the QSVA. Nevertheless, in the next section we compare the performance of the three SearchRank algorithms. Using this approach, the probability of measuring each of the tampered transactions depends on the PageRank distribution. Therefore, it is expected to measure first tampered transaction with many incoming flows from other transactions, which can cause a

significant and dispersed impact on legitimate transactions. Therefore, once a transaction of this nature is identified, it can be traced back to uncover other tampered transactions associated with it or to identify the transactions that have been impacted. This approach allows a classical search acting in parallel to be redirected in a way that enables faster detection of potential fraudulent or affected transactions, compared to an unranked algorithm such as Grover. As a result, it provides an efficient search strategy for detecting fraudulent transactions and accelerates the process of assessing the extent of the impact on other transactions.

Taking all this into account, we can define the steps involved in the QSVA as follows:

1) Construct a transaction graph from the N transactions in the mempool of a miner, where nodes represent transactions and directed edges the relationships between these transactions.

2) Apply a quantum counting algorithm to the database of transactions that constitute the mempool to obtain the total count of manipulated transactions in the database, M.

3) Perform a search algorithm based on the quantum SearchRank on the transaction graph for $t_q = \left\lfloor \sqrt{N/M^*} \right\rceil$ quantum steps, where M^* is the number of remaining tampered transactions, to obtain a tampered transaction depending on its connectivity.

4) Register the obtained tampered transaction as identified by adjusting the filter in the oracle of the SearchRank algorithm.

5) Evaluate the possibility of conducting a classical search on transactions linked to the identified fraudulent transaction to detect additional potentially tampered or affected transactions.

6) Repeat Steps 3, 4, and 5 at most $M - 1$ times to identify all the tampered transactions in the mempool.

8.3. Simulation Results

Given that current quantum computers lack the capacity and development necessary to effectively perform a simulation of our algorithm, we evaluate the functionality of the QSVA using the classical simulation algorithm for the Szegedy quantum walk explained in Chapter 9. Our simulator can emulate the action of the oracle Q_f on a given set of fraudulent transactions using the expression (2.61), therefore simulating the transaction validation process. In order to evaluate the performance of the algorithm, we have selected a dataset consisting of real transactions extracted from the Bitcoin blockchain [230, 231]. This dataset includes a subset of transactions that have been previously classified as licit or fraudulent. The primary objective of the algorithm is to accurately identify all the falsified transactions within this dataset, demonstrating its effectiveness in detecting fraudulent activity in blockchain networks.

The dataset comprises three distinct types of data: information related to each transaction, including the transaction identifier, the timestamp when the transaction was recorded, and the amount of BTC transferred; the classification of each transaction as either licit, fraudulent, or unclassified; and the origin and destination transactions associated with the flow of funds. We filtered the dataset

to include only licit and fraudulent transactions recorded within a specific time frame, simulating the mempool of transactions a miner would encounter at a given moment. From the filtered dataset of classified transactions, which includes the origin and destination transactions associated with BTC movements, we constructed the corresponding transaction graph, which is shown in Figure 8.5. There are $N = 398$ nodes, of which $M = 25$ are tampered.

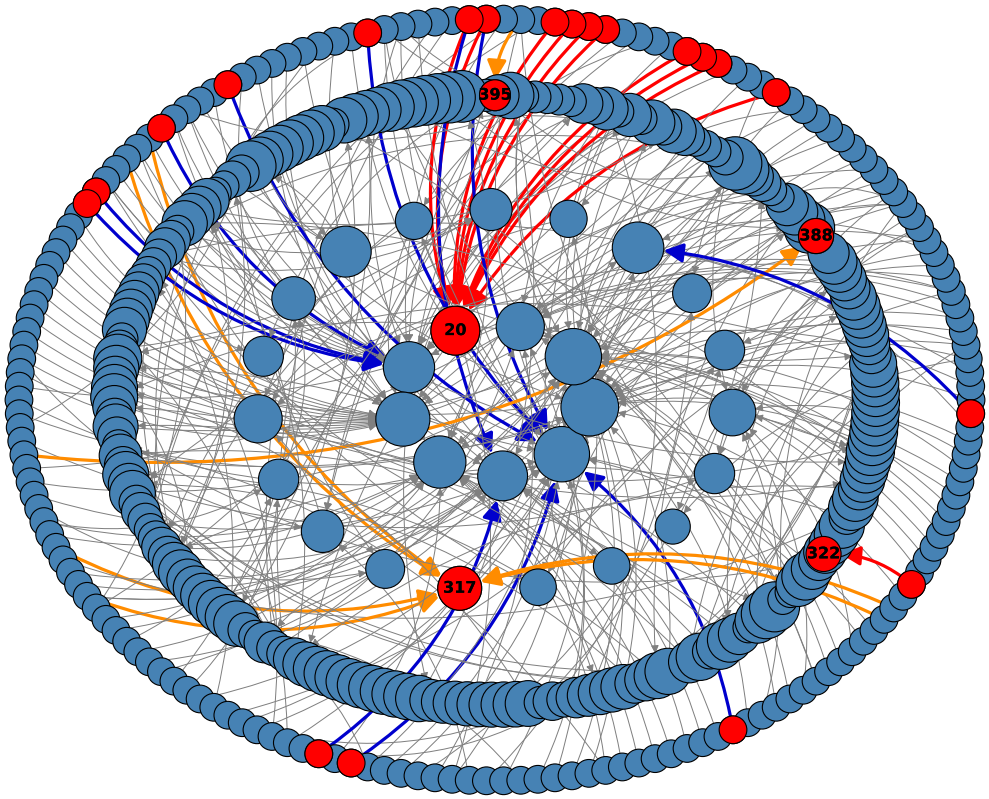


Figure 8.5: *Transaction graph of the filtered Bitcoin dataset for time frame* = 37*. Fraudulent transactions are represented in red, while licit transactions are shown in blue. The nodes in the graph are arranged in concentric circles based on their indegree. Nodes with an indegree of* 0 *are positioned in the outermost circle, nodes in the second and third circles have in-degrees of* 1 *and between* 2 *and* 9*, respectively, and the innermost circle consists of nodes with an indegree of* 10 *or higher. The size of each node is proportional to its classical PageRank value, scaled logarithmically. Edges are color-coded based on the type of transactions they connect: blue edges represent connections from a fraudulent transaction (origin) to a licit transaction (destination), orange edges represent connections from a licit transaction (origin) to a fraudulent transaction (destination), and red edges indicate connections where both the origin and destination are fraudulent transactions. The top five fraudulent transactions are labeled in the graph.*

We have tested the three SearchRank algorithms on the transaction graph to see their ability to find a tampered transactions. The probability of finding a marked node as a function of the quantum time t_q is shown in Figure 8.6(a). The probability reaches its first maximum at the reference time of measurement $t_q = \left\lfloor \sqrt{N/M} \right\rceil$ in the three algorithms. The quantum algorithm reaches a probability below 0.7, whereas in the randomized algorithm is close to 1. In order to see the classification of the fraudulent transactions when measuring at this reference time, we represent their isolated distributions in Figures 8.6(b)-8.6(d). Since in the previous chapter we saw that the ranking of the

SearchRank algorithms resemble more the classical PageRank than the quantum one, in this case we only compare with the classical distribution. Recall that we represent them on different scales, since in the SearchRank distributions the probabilities are amplified. Although the three algorithms agree with the most important node, only the randomized SearchRank provides a distribution resembling the classical PageRank ranking. The top 5 most important nodes, in descending order, are: 20, 317, 322, 388, and 395. The rest of marked nodes are degenerate in the classical distribution since they do not have any link pointing to them. Therefore, their order in the SearchRank algorithm is irrelevant, and since their differences due to quantum fluctuations are very small, we can also regard them as degenerate in this distribution. Therefore, the ranking in the randomized SearchRank perfectly agrees with the classical PageRank distribution.

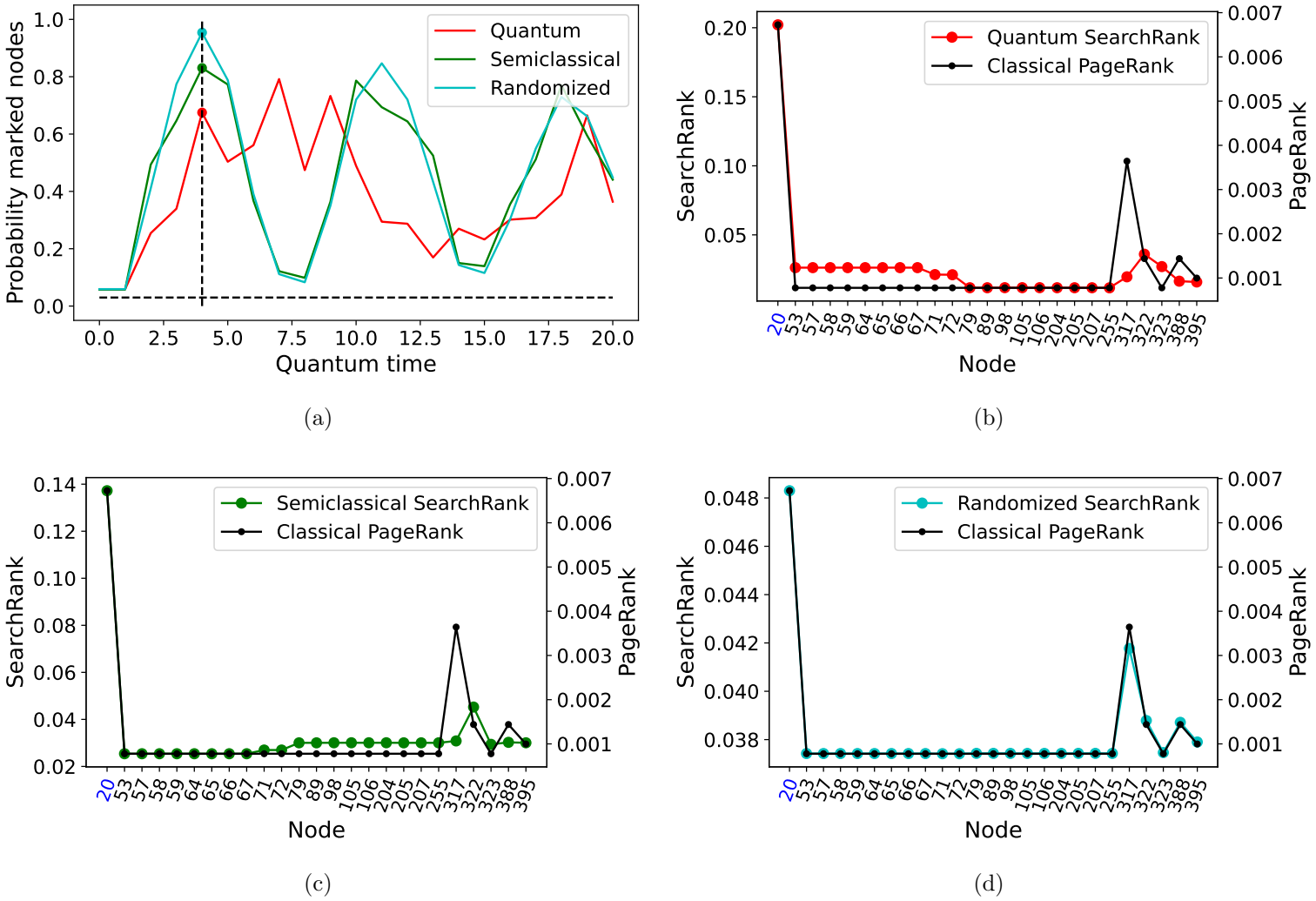


Figure 8.6: *Results for the first iteration of the QSVA. (a) Probability of measuring one of the marked nodes versus the quantum time for the three SearchRank algorithms. The vertical dashed line represents the value* $t_q = \left\lfloor \sqrt{N/M} \right\rceil$. *The horizontal dashed line represents the probability of the marked nodes in the classical PageRank distribution. (b)-(d) Comparison of the SearchRank distributions of the marked nodes with the classical PageRank distribution. The PageRank (right axis) is represented on a different scale from the SearchRank (left axis).*

Adhering to the QSVA procedure, the transaction most likely to be obtained after measurement

is the one associated with node 20. Based on this result, one can trace the transactions connected to it to uncover ten additional fraudulent transactions with a straightforward classical verification, as shown in Figure 8.5. Therefore, less iterations of the QSVA procedure would be needed in the end. In contrast, an unranked quantum search algorithm such as Grover, without any structured prioritization, would detect whatever fraudulent node first, so it would be helpless for a parallel classical search aiming to detect as soon as possible fraudulent activity.

For simulation purposes, we consider that the QSVA is performing alone, so we keep iterating the algorithm until finding the remaining $M - 1$ marked nodes. In this case, we only show the results for the randomized SearchRank, since is the one showing better performance. The probability distributions of the consecutive four iterations are shown in Figure 8.7. Given the closer alignment of the randomized SearchRank with the classical PageRank distribution, the most likely transactions expected to be obtained after measurement in each iteration of the QSVA coincide with the high-ranked transactions of the classical PageRank. The order of the rest of transactions is irrelevant as they are degenerate. All the detected transactions are shown in Table 8.1, along with their transaction identifiers in the dataset.

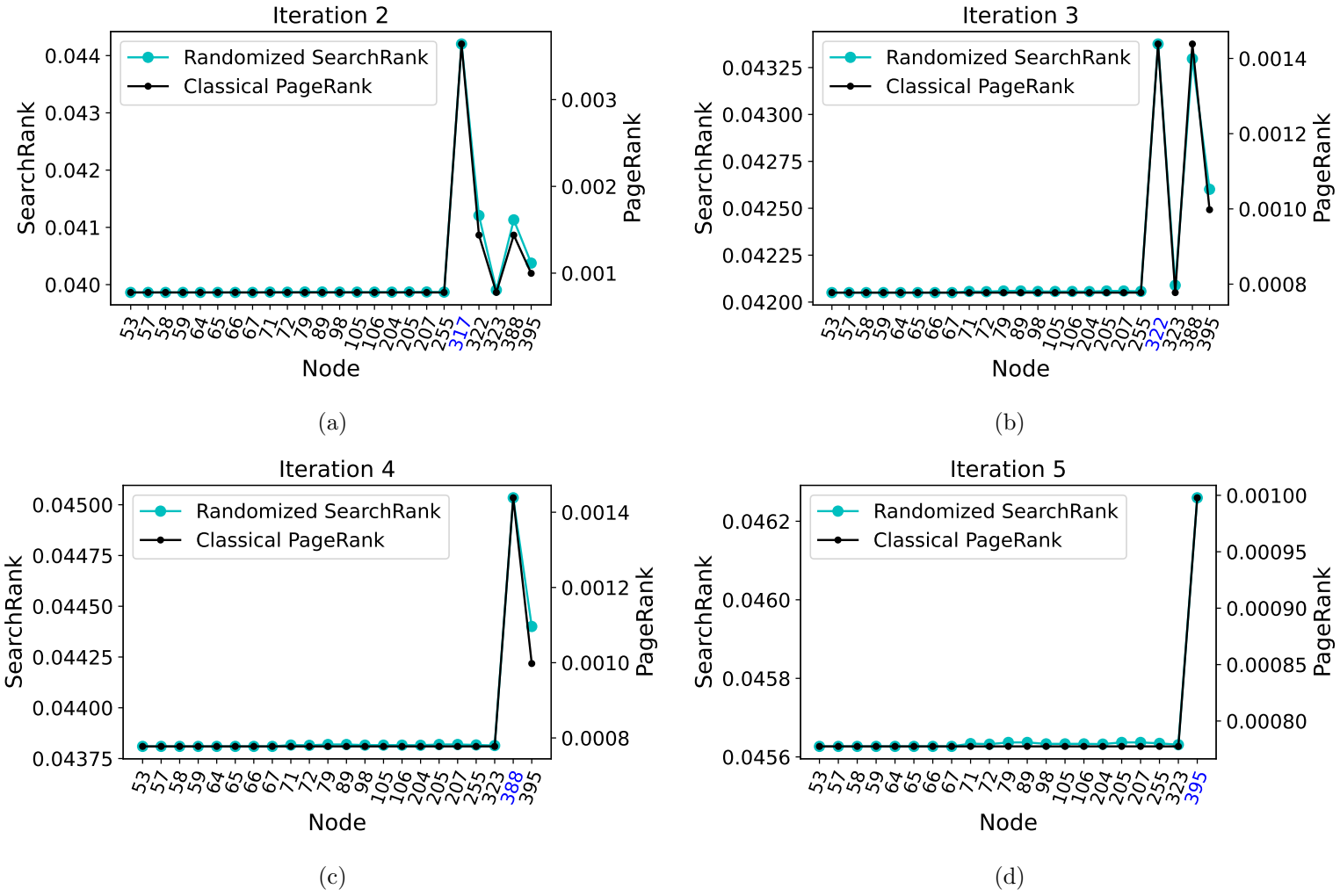
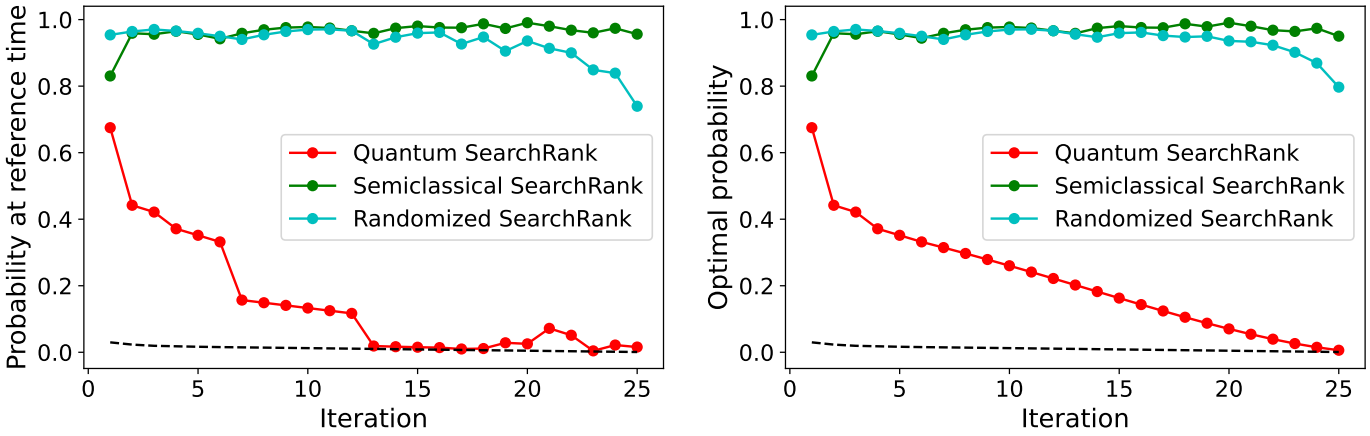


Figure 8.7: *Comparison of the randomized SearchRank distribution of the marked nodes with the classical PageRank distribution for the iterations* 2, 3, 4 *and* 5 *of the QSVA. The PageRank (right axis) is represented on a different scale from the SearchRank (left axis).*

Table 8.1: *Fraudulent transaction identifiers and their corresponding graph node indexes, listed in the order they are expected to be identified by the QSVA. The top five nodes are highlighted.*

Node	TxId	Node	TxId	Node	TxId	Node	TxId	Node	TxId
20	**13735016**	79	28953279	105	29133205	72	28907492	66	28900174
317	**30179316**	89	29035758	106	29133218	323	30204891	58	28878820
322	**30204549**	205	29611862	204	29581775	57	28877539	64	28898351
388	**39684200**	207	29615095	255	30061066	53	28862543	67	28900605
395	**39747107**	98	29132572	71	28903935	59	28879339	65	28898970

Finally, we analyze the probability of finding a marked node for each iteration of the QSVA. The results for the three SearchRank algorithms are shown in Figure 8.8(a) for the reference time of measurement. Moreover, for completeness, we also compare the probability at the optimal time in Figure 8.8(b). As expected, in the quantum algorithm the probability decreases. As discussed in Section 7.3, this sharp decline in the quantum SearchRank is caused by the ratio N/M, which increases as the number of marked elements is reduced when fraudulent transactions are identified. In contrast, the semiclassical and randomized SearchRank algorithms are largely independent of this ratio, allowing the probability to remain consistently high. Therefore, the randomized SearchRank is able to accurately identify all the fraudulent transactions in the network, which, combined with the previous results, makes it the best suitable algorithm for the QSVA.



(a) (b)

Figure 8.8: *Probability of measuring a fraudulent transaction at (a) the reference time* $t_q = \left\lfloor \sqrt{N/M} \right\rceil$ *and (b) the optimal time, as a function of the QSVA iterations. The dashed line represents the probability of the marked nodes in the classical PageRank distribution.*

8.4. Summary of Results and Conclusions

- We have reviewed the fundamentals of blockchain technology, where we have noted that an early detection of tampered transactions is crucial. The faster this occurs, the sooner valid transactions can be included in a block, then added to the ledger. Moreover, we have seen how

useful information can be extracted from graphs representations of the transactions. Therefore, quantum walks could improve the efficiency of blockchain systems.

- We have proposed a quantum protocol denoted as Quantum Signature Validation Algorithm (QSVA). At its core is a SearchRank algorithm, which leverages the transaction graph of blockchain for detecting fraudulent transactions giving priority to those that cause a great impact in the network. Furthermore, this algorithm can work with a parallel classical validation redirected to those transactions that are connected to the transactions detected by the QSVA, and therefore could detect sooner other tampered transactions, or analyze the impact of the attack on them. Thus, this algorithm combines both the quadratic speedup of the Grover algorithm and the information extracted by the PageRank algorithm.

- We have analyzed the performance of the QSVA on a real Bitcoin dataset using classical simulations. From the three SearchRank algorithms, the randomized SearchRank provides the best results as expected, perfectly agreeing with the classical PageRank distribution of the fraudulent transactions. Furthermore, we have proved that the algorithm is able to find all the tampered transactions properly when using the semiclassical and randomized SearchRank subroutines, although only the order of the randomized SearchRank is correct. In contrast, as expected, the naive quantum SearchRank looses its efficacy as soon as the ratio between the number N of nodes and the number M of marked nodes increases.

- Although the results obtained for the QSVA are promising, as is the case with most quantum algorithms proposed for blockchain, further research is needed to solve some issues of the algorithm. First, future work focused on the circuit design of the oracle used for transaction validation is required. The compilation of such a circuit, as well as circuits for the walk operators on general graphs, is currently an open problem. Second, for simulation purposes we have assumed that at each iteration of the QSVA we measure the node with higher probability. Nevertheless, in a real scenario we could measure other of the fraudulent transactions. Indeed, it may be more probable to measure any other of the marked nodes rather than the most important one, since the joint probability of the rest can be higher despite all of them being less important. So far, our algorithm is a proof of concept, and further optimization of the core search algorithm in needed, going beyond the capabilities of the randomized SearchRank. Nevertheless, this approach is promising and is currently able to detect all the tampered transactions.

- So far, we have developed the QSVA for blockchain technologies based on transaction graphs. In the future, it would interesting to study how our algorithm could be used for other systems based on address graphs like Ethereum [204]. Moreover, there is room for developing new quantum algorithms which could improve blockchain systems, or even different distributed ledger technologies (DLTs) like Hashgraph [232] and Holochain [233, 234], which offer unique architectures that could greatly benefit from quantum advancements.

Results III

SQUWALS

Chapter 9

Szegedy QUantum WALks Simulator

The last part of results of this thesis focuses on the classical simulation of the Szegedy quantum walk. It is important to have a classical form of simulating it in order to check the properties of the quantum algorithms based on this quantum walk, since these classical simulations are ideal thereby absent of errors.

A naive approach for simulating the Szegedy walk would require constructing the corresponding unitary operator. As we will see later, the memory resources needed to store it grow as $\mathcal{O}(N^4)$, where N is the number of nodes in the graph. However, using a sparse representation of these matrices, this dependence can be reduced to $\mathcal{O}(N^3)$. Other method based on the spectral decomposition of the operator was proposed [31], also needing memory resources scaling as $\mathcal{O}(N^3)$. At the beginning of this thesis, the spectral method allowed the simulation on networks up to 256 nodes on a computer with 16 GB of RAM memory, so we could perform the numerical calculations for the quantum PageRank in Chapter 5. However, it resulted prohibitive for larger networks, as the ones used for the SearchRank algorithm in Chapter 7.

Therefore, we needed to devise a novel simulation algorithm saving memory resources. In this thesis, we propose a method for simulating the Szegedy quantum walk with memory resources scaling as $\mathcal{O}(N^2)$ [3]. Furthermore, this algorithm also improves the time complexity, speeding up the simulations. Moreover, we show how this algorithm can be implemented to simulate the semiclassical Szegedy walk of Chapter 6, and the quantum walk on mixed states.

There exist some simulators for quantum walks in discrete time, as for example QWalk [235], Hiperwalk [236] and QuantumWalk [237]. This last simulator implements the Szegedy quantum walk in Julia language using sparse matrices for storing the unitary operators. However, as far as we are concerned, there is not an efficient simulator for the Szegedy quantum walk that implements our algorithm. For this reason, we have developed a Python package called SQUWALS (Szegedy QUantum WALks Simulator), which implements our memory-saving algorithm. Moreover, our simulator provides some alternative formulations of the Szegedy quantum walk, including oracles and the complex-phase extensions of Chapter 4.

This chapter is structured as follows. In Section 9.1 we review the traditional methods of simulating the Szegedy quantum walk. In Section 9.2 we show our efficient algorithm for simulating this quantum walk on a classical computer. In Section 9.3 we show how the semiclassical Szegedy walk

can be simulated with our algorithm. In section 9.4 we show an algorithm for simulating the evolution of mixed states. In section 9.5 we present SQUWALS, our Python simulator for the Szegedy quantum walk. Finally, we summarize and conclude in Section 9.6.

9.1. Traditional Simulation Methods

In this section we show the previous simulation algorithms for the Szegedy quantum walk. To understand the complexity of different simulation algorithms, we first need to clarify some concepts regarding the structure and representation of matrices:

Definition 9.1. *Depending on the structure, there are two types of matrices:*

- *Dense matrix: It is a matrix where most elements are different from* 0*.*
- *Sparse matrix: It is a matrix where most elements are* 0*.*

Definition 9.2. *There are two forms of representing a matrix in memory:*

- *Dense representation: This is the naive representation, so that all the elements, including those that are* 0*, are stored in memory.*
- *Sparse representation: In this case, only the non-null elements are stored in memory. However, this method can be less efficient if the matrix is dense enough.*

9.1.1. Naive unitary simulation

The naive method for simulating the Szegedy quantum walk consists of creating the unitary matrix of the single-step operator $U_s = S_w R$ in (3.34), and applying it to a vector state. Recall that the Hilbert space where the walk takes place is

$$\mathcal{H}_S := \text{span}\{|i\rangle_1 |j\rangle_2 , \ i, j = 0, 1, ..., N-1\} = \mathbb{C}^N \otimes \mathbb{C}^N, \tag{9.1}$$

whose dimension is N^2. Therefore, the matrix representing the unitary operator U_s has $N^2 \times N^2 = N^4$ elements, so to store it in memory with a dense representation we would need resources scaling as $\mathcal{O}(N^4)$. Nevertheless, since the reflection R has a block-diagonal structure, there are N blocks of size $N \times N$. Thus, for a dense transition matrix G, the reflection R has N^3 non-null elements. The swap S_w just permutes these elements, so the unitary operator U_s also has N^3 elements. Thus, if we used a sparse representation for the unitary matrices, the resources requirements would improve to $\mathcal{O}(N^3)$.

This scaling is still so much prohibitive, so that there is a limitation on simulating the quantum walk on graphs with dense transition matrices, as for example the ones used for the quantum PageRank algorithm [31, 32]. However, there are graphs with very sparse transition matrices, like regular lattices, so that the unitary operator can be efficiently stored with a sparse representation even for

thousands of nodes. Thus, this approach is feasible for sparse transition matrices, but not for dense ones.

If we wanted to simulate the action of an oracle operator Q_f, we would just create a diagonal operator according to the marked nodes, as explained in Section 2.3.6. With it, we can create an operator for the oracle acting on the first register as:

$$Q_1 := Q_f \otimes \mathbb{1}_N, \tag{9.2}$$

or on the second register as:

$$Q_2 := \mathbb{1}_N \otimes Q_f. \tag{9.3}$$

In both cases, the resulting operator is also diagonal, so that it does not change the number of non-null elements in the unitary evolution operator U_s if it is introduced. Therefore, the memory complexity remains the same after the introduction of an oracle.

For the time complexity, it is expected to scale with the size N the same as the memory requirements, since in the matrix-vector product each matrix element intervenes only once. With regard to the number of time steps of the walk t, the complexity is trivially linear, since we have to apply the operator U_s sequentially for each time step.

9.1.2. Spectral decomposition simulation

Other method proposed to simulate the Szegedy quantum walk is based on the spectral decomposition of the operator [1, 31]. There is an invariant subspace, known as dynamical subspace, where all the dynamics of the walk usually takes place. It is defined as:

$$\mathcal{H}_D := \text{span}\left\{|\psi_i\rangle, S_w|\psi_i\rangle\right\}. \tag{9.4}$$

In Appendix A.1 we prove that it is invariant. For an initial state formed as a superposition of the $|\psi_i\rangle$ states in (3.31), we can simulate its evolution if we decompose it as a linear combination of the eigenvectors in this subspace. Once obtained the coefficients of the combination, we just have to multiply them by the corresponding eigenvalues raised to the power of time t, and recompose the resulting state with the new coefficients. The dynamical subspace has a dimension of (at most) $2N$, and since each eigenvector has *a priori* N^2 non-null elements, we need memory resources scaling as $\mathcal{O}(N^3)$. A numerical form of obtaining the eigenvalues and eigenvectors from the transition matrix G is shown in Appendix A.2.

Again, for the time complexity with respect to the graph size N, a scaling as $\mathcal{O}(N^3)$ is expected. Nevertheless, an advantage with respect to the naive method is that the different time steps of the walk do not have to be simulated sequentially. We can obtain the result for whatever time step just raising the eigenvalues to the power of that particular time t. Moreover, we can perform the simulation of different time steps in parallel, speeding up the simulation.

Although this algorithm seems to improve the naive method, it has some limitations. On the one hand, there is no spectral decomposition for the operator including oracles. Furthermore, although we can provide a spectral decomposition after the introduction of some of the phase extensions studied in Chapter 4, currently there is no method for including local arbitrary phase rotations (APR), so

that we can only deal with global APR and link phases. On the other hand, it is not possible to simulate the single-step operator U_s acting on a state outside the dynamical subspace $\mathcal{H}_D$. As proved in Appendix A.1, the action of the single-step unitary operator U_s on the orthogonal complement is just $-S_w$, so the eigenvalues are ± 1. Therefore, the evolution of the perpendicular component of the initial state is not trivial, and we cannot calculate it. Nevertheless, we can do so for the double-step operator W_s, since in the orthogonal complement it acts as $(-S_w)^2 = \mathbb{1}$, and therefore the perpendicular component does not evolve. Anyway, such an initial state is unusual.

9.2. Optimized Classical Simulation

In this thesis we are interested in simulating the walk for a general transition matrix G, so that, without loss of generality, we are assuming that it is dense. Thus, in this section we explain a new method for simulating the Szegedy quantum walk with time and memory requirements scaling as $\mathcal{O}(N^2)$ for dense transition matrices, which is optimal. Any Szegedy unitary operator can be expressed in terms of the reflection R, the swap S_w, and the oracles Q_1 and Q_2. Therefore, in order to simulate this quantum walk, we provide algorithms for simulating each of the operators individually, so that they can be used as building blocks to simulate any unitary operator. These methods avoid the explicit construction of the corresponding matrices. Moreover, these algorithms are constructed in a vectorized manner, so that we avoid the explicit use of *for* loops. Thus, they are optimized for being used with the numerical Python library NumPy [238], which can parallelize calculations on a CPU, being faster when the operations are vectorized.

9.2.1. Matrix state representation

A generic state in the Szegedy Hilbert space $\mathcal{H}_S$ can be written as:

$$|\phi\rangle = \sum_{i,j=0}^{N-1} a_{ij} \, |i\rangle_1 \, |j\rangle_2 \, . \tag{9.5}$$

In order to simulate the Szegedy quantum walk with the least possible memory resources, we have to think of the quantum vector state as a matrix. Let us define the matrix Φ representing the vector state $|\phi\rangle$ as the one whose elements are

$$\Phi_{ij} = a_{ji}. \tag{9.6}$$

Note that using this notation, the column index represents the first register, whereas the row index represents the second register. This is so for convenience. If we divide the vector state into blocks corresponding to each state of the computational basis of the first register, then each block corresponds to each column of the matrix state. In Figure 9.1 we show an example for a network with $N = 3$ nodes.

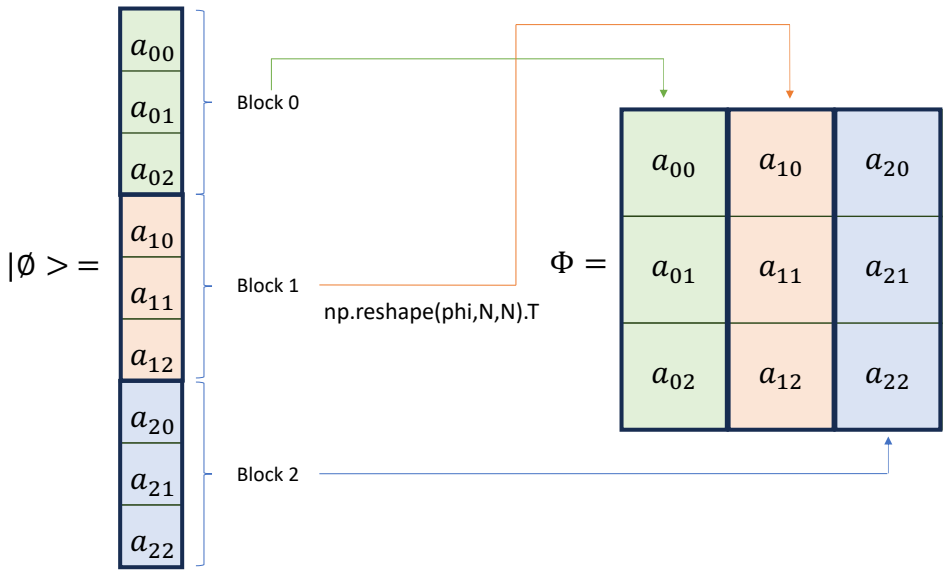


Figure 9.1: *How to represent a vector state by a matrix state for a network with $N = 3$ nodes. The blocks of the vector, indexed by the computational basis of the first register, correspond to the columns of the matrix. Beware that the indexes in the coefficients a_{ij} are swapped with respect to the ones of the usual matrix notation Φ_{ij}.*

9.2.2. Reflection operator

Let us recall the expression of the reflection operator:

$$R := 2\Pi - \mathbb{1}, \qquad \Pi := \sum_{i=0}^{N-1} |\psi_i\rangle \langle\psi_i| , \qquad |\psi_i\rangle := |i\rangle_1 \otimes \sum_{k=0}^{N-1} \sqrt{G_{ki}}\, |k\rangle_2 . \tag{9.7}$$

We need to store in memory the information of the $|\psi_i\rangle$ states. It turns out that each of these states has only N non-null elements. Therefore, we can create an $N \times N$ matrix Ψ, where each column i has the N non-null elements of $|\psi_i\rangle$. These states are then expressed as:

$$|\psi_i\rangle = \sum_{k=0}^{N-1} \Psi_{ki}\, |i\rangle_1 |k\rangle_2 . \tag{9.8}$$

An example of condensing the vectors $|\psi_i\rangle$ for a graph with $N = 3$ nodes is shown in Figure 9.2.

The matrix Ψ corresponds to the element-wise square root of the transition matrix G:

$$\Psi_{ij} = \sqrt{G_{ij}}. \tag{9.9}$$

In order to simulate the reflection operator in (9.7), first we need to obtain the action of the projector operator Π. The projection gives us the parallel component of the vector $|\phi\rangle$ to the space spanned by the vectors $|\psi_i\rangle$:

$$\Pi\, |\phi\rangle = |\phi\rangle_{\parallel} = \sum_{i=0}^{N-1} C_i\, |\psi_i\rangle , \tag{9.10}$$

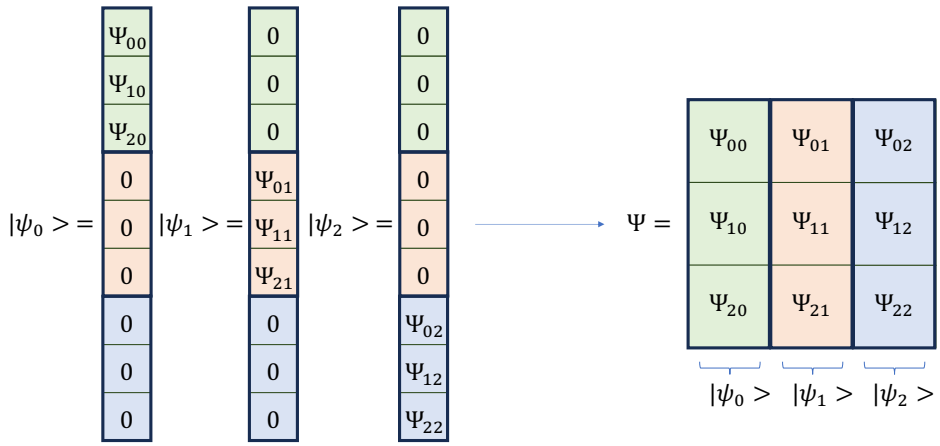


Figure 9.2: *How to condense all the information of the $|\psi_i\rangle$ states in a matrix for a network with $N = 3$ nodes. Each column i of the matrix Ψ corresponds to the non-null block of each $|\psi_i\rangle$ state.*

where the coefficients C_i are obtained as:

$$C_i = \langle\psi_i|\phi\rangle = \sum_{k=0}^{N-1} a_{ik}\Psi_{ki}. \tag{9.11}$$

The vector C with the coefficients can be obtained by multiplying element-wise the matrix state Φ with the matrix Ψ, and adding over the rows of the resulting matrix. In Figure 9.3 we show an example for a network with $N = 3$ nodes.

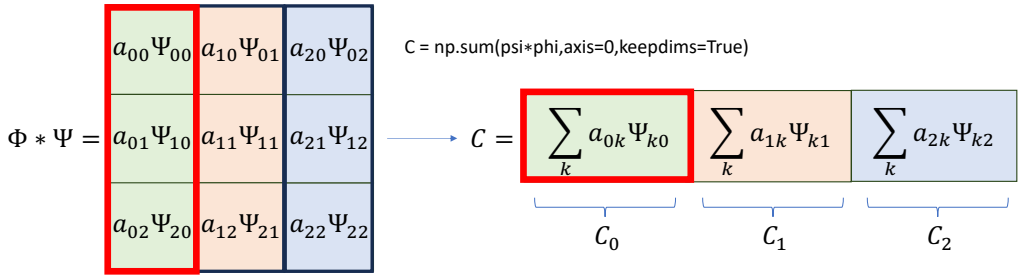


Figure 9.3: *How to obtain the vector C in (9.11) for a network with $N = 3$ nodes. The element-wise multiplication between Φ and Ψ results in a matrix whose i-th column has the product of the non-null elements of $|\psi_i\rangle$ and the elements of $|\phi\rangle$. Thus, summing the elements of each column, the coefficients C_i are obtained.*

Once the coefficients are obtained, we can use the broadcasting feature of NumPy to multiply element-wise the row vector C with the matrix Ψ. The resulting matrix has in the i-th column the non-null elements of $|\psi_i\rangle$ multiplied by the coefficient C_i. Thus, it results in the matrix representing the parallel component of our state, $|\phi\rangle_{\|}$. An example is shown in Figure 9.4.

Finally, the reflection of the state is obtained straightforwardly as

$$R\,|\phi\rangle = 2\,|\phi\rangle_{\|} - |\phi\rangle\,. \tag{9.12}$$

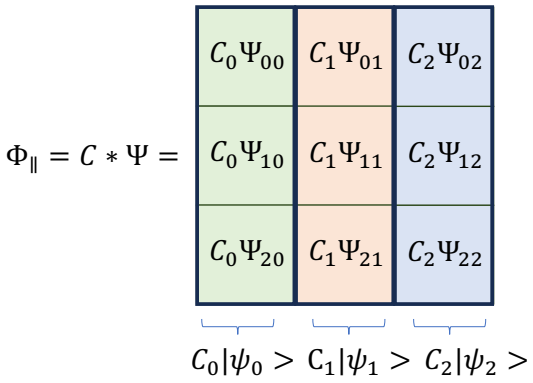


Figure 9.4: *How to obtain the parallel component of the state $|\phi\rangle$ for a network with $N = 3$ nodes. The element-wise product of C and Ψ results in a matrix where each column represents the non-null elements of $C_i |\psi_i\rangle$, which ends up representing the vector $|\phi\rangle_{\|}$.*

So far, we have devised an algorithm to simulate the standard reflection operator. However, it can be extended introducing the complex phases of Chapter 4. Next, we show how to implement these extensions in the simulation.

9.2.2.1. Link phases

The link phases are introduced by modifying the $|\psi_i\rangle$ states as follows [149, 167]:

$$|\psi_i(\varphi)\rangle := \sum_{k=0}^{N-1} e^{\mathrm{i}\varphi_{ik}} \sqrt{G_{ki}} \, |i\rangle_1 \, |k\rangle_2 \, . \tag{9.13}$$

The original reflection operator is recovered when all the phases φ_{ij} are equal to 0.

In order to introduce this modification in our algorithm, we have to substitute the matrix Ψ in (9.9) by

$$\Psi_{ij} = e^{\mathrm{i}\varphi_{ji}} \sqrt{G_{ij}}, \tag{9.14}$$

which can be done easily creating a matrix φ with the phases, taking the element-wise exponential, and multiplying its transpose element-wise with the element-wise square root of the transition matrix G. Moreover, since the matrix Ψ is complex, its elements must be conjugated in equation (9.11) for obtaining of the coefficients C_i, and so in Figure 9.3.

9.2.2.2. Arbitrary phase rotations

Let us first see the case of a global APR. It consists of transforming the reflection operator into

$$R(\theta) := (1 - e^{\mathrm{i}\theta})\Pi - \mathbb{1}. \tag{9.15}$$

The original reflection is recovered when the phase θ is equal to π. To simulate it, we just have to modify expression (9.12) as:

$$R(\theta) \, |\phi\rangle = (1 - e^{\mathrm{i}\theta}) \, |\phi\rangle_{\|} - |\phi\rangle \, , \tag{9.16}$$

in the last step of the algorithm.

In the case of local APR phases, we have the following operators:

$$R(\vec{\theta}) := 2\Sigma(\vec{\theta}) - \mathbb{1}, \quad \Sigma(\vec{\theta}) := \frac{1}{2}\sum_{i=0}^{N-1}(1 - e^{\mathrm{i}\theta_i})\,|\psi_i(\varphi)\rangle\,\langle\psi_i(\varphi)|\,. \tag{9.17}$$

Let us denote $|\phi\rangle_\Sigma$ as the result of acting $\Sigma(\vec{\theta})$ on the state $|\phi\rangle$. We need to obtain this vector for the simulation. Indeed, since the factor 2 in $R(\vec{\theta})$ is going to cancel the factor $1/2$ in $\Sigma(\vec{\theta})$, we calculate directly the state $2\,|\phi\rangle_\Sigma$:

$$2\,|\phi\rangle_\Sigma = 2\Sigma(\vec{\theta})\,|\phi\rangle = \sum_{i=0}^{N-1}(1 - e^{\mathrm{i}\theta_i})C_i\,|\psi_i\rangle\,, \tag{9.18}$$

where the coefficients C_i are obtained the same as for the standard case, as in equation (9.11) and Figure 9.3. Therefore, so far, the algorithm remains the same. Now, we define the modified coefficients $\widetilde{C}_i$ as:

$$\widetilde{C}_i = (1 - e^{\mathrm{i}\theta_i})C_i, \tag{9.19}$$

so that equation (9.18) can be rewritten as:

$$2\,|\phi\rangle_\Sigma = \sum_{i=0}^{N-1}\widetilde{C}_i\,|\psi_i\rangle\,. \tag{9.20}$$

The vector $\widetilde{C}$ representing the new coefficients is obtained creating a vector with the factors $1 - e^{\mathrm{i}\theta_i}$ and multiplying it element-wise with the vector C. Again, we can use the broadcasting feature of NumPy to multiply element-wise the row vector $\widetilde{C}$ with the matrix Ψ, to calculate the matrix representing the vector $2\,|\phi\rangle_\Sigma$ in a similar manner as in Figure 9.4. The action of $R(\vec{\theta})$ is obtained by an element-wise matrix subtraction of the initial state:

$$R(\vec{\theta})\,|\phi\rangle = 2\,|\phi\rangle_\Sigma - |\phi\rangle\,. \tag{9.21}$$

9.2.3. Swap operator

The action of the swap operator on a vector of the computational basis is

$$S_w\,|i\rangle_1\,|j\rangle_2 = |j\rangle_1\,|i\rangle_2\,. \tag{9.22}$$

Thus, it corresponds to exchanging the coefficients a_{ij} of the vector state (9.5) with the coefficients a_{ji}. Given the matrix form of expressing the vector state, this operation corresponds straightforwardly to the transposition of the matrix Φ.

In the case of the twisted swap operator $S_w(\Omega)$ [167] in (4.2), we discussed that it corresponds to a normal swap S_w, followed by the application of a relative phase to each computational basis state. To simulate it, we would just element-wise exponentiate the matrix Ω, multiply it element-wise with the matrix state Φ, and then transpose it.

9.2.4. Oracle operators

We have defined two oracle operators, Q_1 in (9.2), and Q_2 in (9.3), depending on which register is used for marking nodes. They are formed by the phase-flip oracle Q_f, whose action is emulated knowing beforehand the nodes that are marked, as discussed in Section 2.3.6. Recall that its action on a state of the computational basis is

$$Q_f |i\rangle := \begin{cases} -|i\rangle & \text{if } i \in \mathcal{M}, \\ |i\rangle & \text{otherwise,} \end{cases} \tag{9.23}$$

where $\mathcal{M}$ is the set of marked nodes.

On the one hand, in order to mark node i in the first register, we have to invert the sign of the coefficients a_{ij} $\forall$ j of the vector state in (9.5). Given the matrix form Φ representing this vector, the index i in a_{ij} refers to the columns. Thus, we have to invert the sign of the elements along the i-th column. Having a list of marked nodes, we can multiply the columns corresponding to the marked nodes in a vectorized form in Python. On the other hand, if we wanted to mark the nodes in the second register, we could proceed in a similar way but inverting the sing of the elements along the corresponding rows. An example is shown in Figure 9.5 for a network with $N = 3$ nodes, where nodes 0 and 2 are being marked.

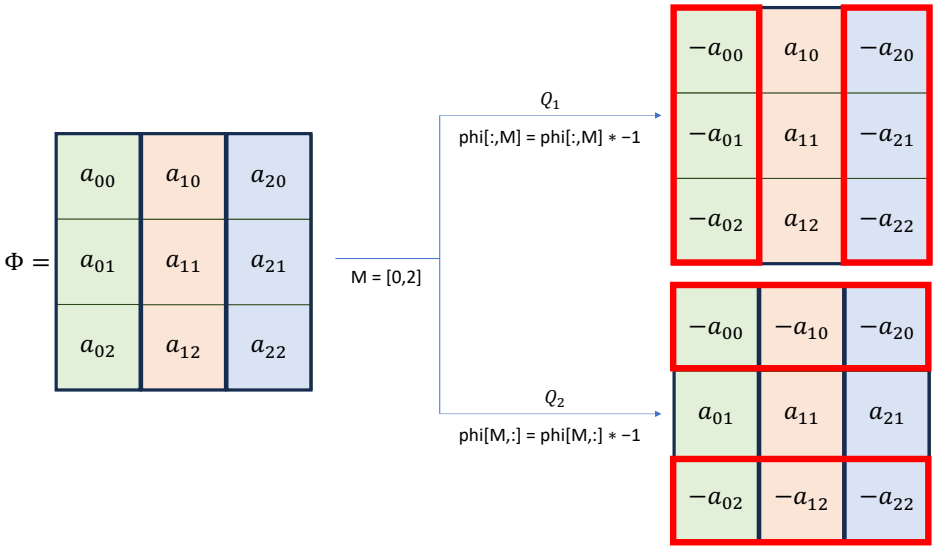


Figure 9.5: *How to apply the oracle operators for a network with $N = 3$ nodes. In this case, nodes 0 and 2 are being marked. In order to mark them in the first (second) register, the corresponding columns (rows) are multiplied by -1.*

The same as the reflection operator, the oracle can be extended with an arbitrary phase rotation as:

$$Q_f(\theta) := \mathbb{1}_N - (1 - e^{i\theta}) \sum_{k \in M} |k\rangle \langle k| , \tag{9.24}$$

whose action on the computational basis is

$$Q_f(\theta) |i\rangle = \begin{cases} e^{i\theta} |i\rangle & \text{if } i \in \mathcal{M}, \\ |i\rangle & \text{otherwise.} \end{cases} \tag{9.25}$$

Thus, to introduce the phase in the algorithm, we just have to multiply the elements by $e^{i\theta}$ instead of by -1.

9.2.5. Measurement

The probability of measuring node i after measuring the first register is given by

$$(p_1)_i = ||\,{}_1\langle i|\phi\rangle\,||^2 = \left|\left|\sum_{k,l=0}^{N-1} a_{kl}\;\; {}_1\langle i|k\rangle_1\,|l\rangle_2\right|\right|^2 = \left|\left|\sum_{l=0}^{N-1} a_{il}\,|l\rangle_2\right|\right|^2 = \sum_{l=0}^{N-1}|a_{il}|^2, \tag{9.26}$$

where we have used in the last step that the computational basis is orthonormal. Analogously, the probability of measuring node i if we were measuring the second register is given by

$$(p_2)_i = ||\,{}_2\langle i|\phi\rangle\,||^2 = \left|\left|\sum_{k,l=0}^{N-1} a_{kl}\,|k\rangle_1\;\; {}_2\langle i|l\rangle_2\right|\right|^2 = \left|\left|\sum_{k=0}^{N-1} a_{ki}\,|l\rangle_2\right|\right|^2 = \sum_{k=0}^{N-1}|a_{ki}|^2. \tag{9.27}$$

To obtain the probabilities of measuring each of the nodes, first we need to take the squared modulus of all the elements in the matrix state Φ. After that, the probability of measuring node i in the first (second) register is obtained by adding the elements of the i-th column (row), according to (9.6), (9.26) and (9.27). A vectorized way of doing this with NumPy is shown in Figure 9.6.

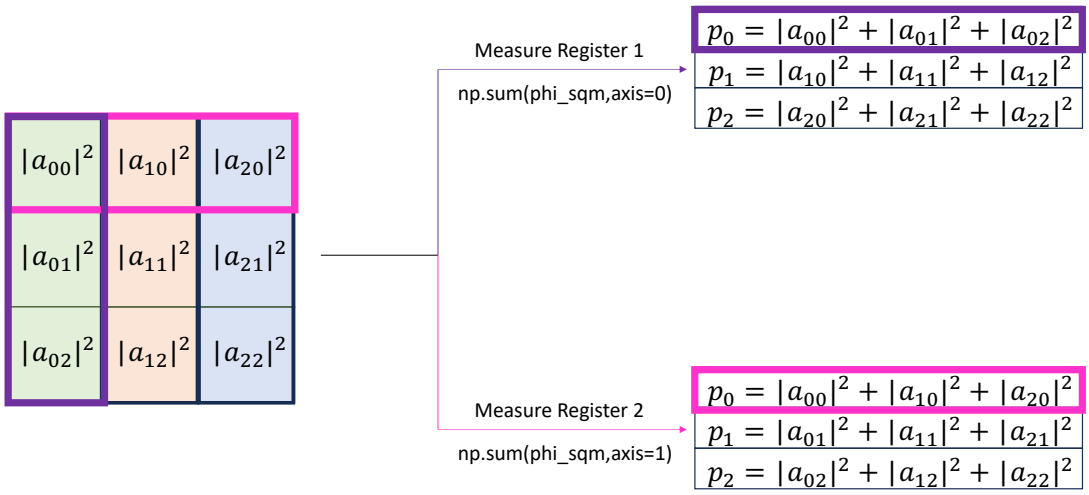


Figure 9.6: *How to obtain the probability distributions of the walker for a network with $N = 3$ nodes. First, the squared modulus of all the elements in Φ is taken. To obtain the probabilities after measuring in the first (second) register, the elements of each column (row) are summed.*

9.2.6. Initial state

Usually, the initial state of a Szegedy quantum walk is a linear combination of the $|\psi_i\rangle$ states:

$$|\phi\rangle = \sum_{i=0}^{N-1} a_i\,|\psi_i\rangle\,. \tag{9.28}$$

It can be seen that the matrix Ψ corresponds to the sum of the matrices representing the $|\psi_i\rangle$ states. Thus, the matrix representing this initial state is obtained by multiplying each column i by the corresponding amplitude a_i. In the case of an equal superposition, we just have to divide the matrix Ψ by $\sqrt{N}$.

9.2.7. Memory and time complexity

Given the the procedures described for simulating the action of the different operators, we can formulate the following lemmas about our algorithm.

Lemma 9.1 (Memory)**.** *The memory storage of the SQUWALS algorithm scales as* $\mathcal{O}(N^2)$.

In all the algorithms that we have seen in this section, the objects are dense $N \times N$ matrices. Thus, the number of elements that we have to store for each application of the unitary operator scales as $\mathcal{O}(N^2)$. For a dense classical matrix we just have N^2 elements, so that this scaling of the memory requirements is the minimum achievable, and then is optimal. Moreover, since we do not need the quantum states at each time step, but the probabilities, we can just measure each time we apply the algorithm for simulating the unitary evolution without saving the intermediate quantum states. Note that at each step we have N probability elements, so for big enough networks the memory needed to store them is negligible with respect to the cost of storing the quantum state. Therefore, the memory requirements would not increase with the number of time steps, and only depend on the size of the graph as $\mathcal{O}(N^2)$.

Lemma 9.2 (Time Complexity)**.** *The time complexity of the SQUWALS algorithm scales as* $\mathcal{O}(N^2)$.

Regarding the time complexity of the algorithm, the most costly operations are the multiplications. As we have seen, all the multiplications are done element-wise, so that each element of the $N \times N$ matrices intervene only once for each of the operations. Thus, the number of times that an element is used does not depend on N. Since in each element-wise multiplication we have N^2 elements intervening, then the time needed by the algorithm to run will scale theoretically as $\mathcal{O}(N^2)$, the same as the memory requirements. Finally, since we need to calculate all the time steps sequentially, the time required by the algorithm will scale linearly with the number of time steps.

These lemmas are valid for dense representations of matrices, where even null elements are being stored. However, if the transition matrix G only has very few non-null elements, then we could use a sparse representation to reduce the requirements of our algorithm even further. For example, for a cyclic 2D lattice the number of non-null elements in G is only $4N$, since there are N nodes with four connections each. Then, the dimension of the reduced subspace $\mathcal{H}_S^R$, where the walk takes place (see Section 3.5), grows as $\mathcal{O}(N)$, and so the resources needed for the simulation.

9.3. Classical Simulation of the Semiclassical Szegedy Walk

In Chapter 6, we proposed a new kind of walk algorithm that mixes quantum and classical properties, called semiclassical Szegedy walk [2]. Recall that there are two parameters to describe a semiclassical

walk. The first one is the quantum time t_q, which is the number of times we apply the unitary evolution between measurements. The second one is the classical time t_c, which is the number of times that the scheme of quantum evolution and measurement is repeated.

In order to simulate these walks on a classical computer, we need to obtain the semiclassical matrices. There are two classes of walks depending on what register is measured to obtain the classical position:

$$ {}_1G_{ji}^{(t_q)} := \left|\left|\, {}_1\langle j|\, U^{t_q}\, |\psi_i\rangle \right|\right|^2, \qquad {}_2G_{ji}^{(t_q)} := \left|\left|\, {}_2\langle j|\, U^{t_q}\, |\psi_i\rangle \right|\right|^2. \tag{9.29} $$

The left-subscript in the semiclassical matrices denotes the class of the walk, and the quantum time t_q characterizes the particular semiclassical walk of the family.

Since a classical walk is easy to simulate given a transition matrix, the semiclassical walk simulation reduces to calculate these semiclassical matrices. For this, we have to simulate the quantum evolution of all the $|\psi_i\rangle$ states. A naive approach would be to use our simulation algorithm N times with a *for* loop, taking each of the $|\psi_i\rangle$ states as initial state. In this case, the memory resources would still scale as $\mathcal{O}(N^2)$ since only one quantum evolution is being performed at a time, and the time needed would be multiplied by N, scaling as $\mathcal{O}(N^3)$.

If the memory resources of the computer are permissive enough, we can save time vectorizing the operations with NumPy. We can stack different initial matrix states along a third dimension, so that the broadcasting feature allows to perform the quantum evolution on all of them with the same operations that we have seen in the previous section. An example of a vectorized initial state with all the $|\psi_i\rangle$ states is shown in Figure 9.7 for a network with $N = 3$ nodes.

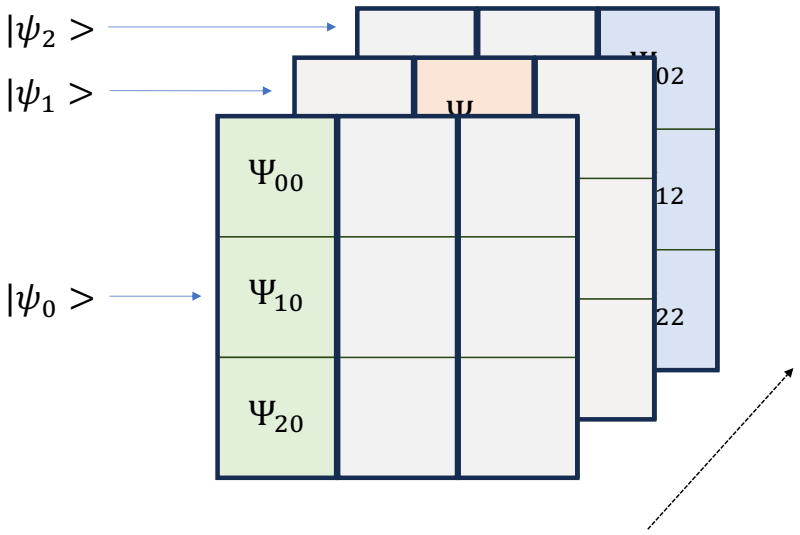


Figure 9.7: *How to stack different initial states for a network with $N = 3$ nodes. The different matrix states are stacked along a third dimension, forming a 3D tensor. In this case, the three $|\psi_i\rangle$ states are stacked in a batch.*

In the case of vectorizing all the operations, the memory requirements would scale now as $\mathcal{O}(N^3)$. Note that the time complexity remains as $\mathcal{O}(N^3)$, since it only depends on the number of operations, although these can be done faster thanks to the vectorization and how they could be parallelized. Depending on the memory resources of the computer, we could make a batch with all the $|\psi_i\rangle$ states,

or use mini-batches composed of a limited number of states to save memory. In the limiting case of one state at a time, the scaling of the memory as $\mathcal{O}(N^2)$ is recovered.

9.4. Classical Simulation with Mixed States

Since the Hilbert space $\mathcal{H}_S$ is of dimension N^2, a naive approach for simulating the evolution of a mixed state would require to store an $N^2 \times N^2$ matrix, which is prohibitive and would need memory resources scaling as $\mathcal{O}(N^4)$. Moreover, depending on the state, it can also be prohibitive if we used a sparse representation of this matrix. In this section, we show a method for simulating the Szegedy quantum walk on a mixed state that is expressed as a weighted sum of a particular set of pure states. In contrast to the simulation algorithm on pure states, which can be used to obtain all the complex amplitudes of the state, the algorithm that we provide for mixed states does not calculate the matrix elements of the density operator, but only the probabilities of measuring each of the nodes.

Let us denote $|\beta_i\rangle$ to the vectors of a set that are in the Szegedy Hilbert space $\mathcal{H}_S$. The mixed state is then expressed as:

$$\rho = \sum_{i=0}^{M-1} c_i \left|\beta_i\right\rangle \left\langle\beta_i\right|, \tag{9.30}$$

where c_i are non-negative real numbers adding 1, and M is the size of the set. Note that this size does not have to be N^2 in principle.

Without loss of generality, let us suppose that the system evolves with the single-step unitary operator U_s. The mixed state after the evolution is

$$U_s \rho U_s^\dagger = \sum_{i=0}^{M-1} c_i \; U_s \left|\beta_i\right\rangle \left\langle\beta_i\right| U_s^\dagger. \tag{9.31}$$

Let us suppose that we want to measure the first register of the system. The probability of measuring node j is given by

$$(p_1)_j = \mathrm{Tr}\left[{}_1\langle j| \, U_s \rho U_s^\dagger \, |j\rangle_1 \right] = \mathrm{Tr}\left[\sum_{i=0}^{M-1} c_i \; {}_1\langle j| \, U_s \left|\beta_i\right\rangle \left\langle\beta_i\right| U_s^\dagger \, |j\rangle_1 \right]. \tag{9.32}$$

The same as the product of a vector state with a vector of the computational basis of the first register results in a vector in the second subspace, in this case the sandwich results in a density matrix in the second subspace. Thus, we have to trace to obtain the final probability. Since the product ${}_1\langle j| \, U_s \left|\beta_i\right\rangle$ results in a vector in the second register, we can express the trace as:

$$\mathrm{Tr}\left[\sum_{i=0}^{M-1} c_i \; {}_1\langle j| \, U_s \left|\beta_i\right\rangle \left\langle\beta_i\right| U_s^\dagger \, |j\rangle_1 \right] = \sum_{i=0}^{M-1} c_i \left|\left| \, {}_1\langle j| \, U_s \left|\beta_i\right\rangle \right|\right|^2. \tag{9.33}$$

The term $\left|\left| \, {}_1\langle j| \, U_s \left|\beta_i\right\rangle \right|\right|^2$ corresponds to the probability of measuring node j after the quantum walk evolution of the state $|\beta_i\rangle$. Thus, the probability of measuring node j starting from the mixed state is a weighted mean of the results for each of the states that form the mixed state, where the weights are given by the coefficients c_i.

If we were measuring the second register instead, a similar result would be obtained. Thus, in order to simulate the probabilities of the walker from a mixed state, we need to simulate the quantum walk for each of the states $|\beta_i\rangle$ that comprise it. The same as in the semiclassical walk simulation, we can stack them in a batch to simulate them in a vectorized form, as long as the memory resources allow it. However, we can simulate one at a time, so that the memory requirements again scale as $\mathcal{O}(N^2)$.

9.4.1. Relation with the semiclassical walk

We have previously mentioned that the common initial state for the Szegedy quantum walk in (9.28) is formed by a superposition of the $|\psi_i\rangle$ states. Thus, these states could be an appropriate set to form an initial mixed state as:

$$\rho = \sum_{i=0}^{N-1} c_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{9.34}$$

Generalizing the evolution for a quantum time step t_q and using (9.32), (9.33) and (9.29), we obtain a relation between the probabilities measuring the first register and the semiclassical matrices of class I:

$$(p_1)_j = \mathrm{Tr}\left[{}_1\langle j| U_s^{t_q} \rho U_s^{t_q\dagger} |j\rangle_1 \right] = \sum_{i=0}^{N-1} {}_1G_{ji}^{(t_q)} c_i. \tag{9.35}$$

If c_i are the coefficients of a column probability vector, then the quantum walk on the mixed state in (9.34) during t_q steps corresponds to the first classical step of the corresponding semiclassical walk on the probability distribution given by the coefficients c_i. The same discussion is valid for measuring in the second register, using the semiclassical walks of class II. From the point of view of the simulation on a classical computer, this means that the simulation of the semiclassical walks by means of the semiclassical matrices implies solving the problem for any mixed state of the form in (9.34).

9.5. SQUWALS

Given the algorithms presented above, we have developed a Python library called SQUWALS, which performs the operations in a vectorized form using NumPy. This software allows the user to simulate Szegedy quantum walks in a simple manner, without a deep knowledge of the algorithm itself. Moreover, the user does not have to transform the quantum states into matrices, so that they can be provided just as normal NumPy vectors.

In this section, we describe the code fundamentals for using our simulator, and show that the scaling in both the time and memory needed to run is compatible with $\mathcal{O}(N^2)$, as theoretically predicted. Finally, we show some applications based on the Szegedy quantum walk that are available in our library.

9.5.1. Simulator fundamentals

In order to simulate a Szegedy quantum walk, the user needs to provide an initial state vector $|\phi\rangle$ and a classical transition matrix G. This matrix must be column-stochastic, so that each column adds up to 1. In case that the user wants to use the initial quantum superposition state in (9.28), we provide a function that creates it from the transition matrix.

Our library provides three classes for the three main operators, which are the reflection R, the swap S_w and the oracle Q_f. On the one hand, the reflections are created providing the transition matrix G, and optionally, the phase extensions mentioned in 9.2.2. On the other hand, the oracle is created providing a list with the marked nodes, the subspace where the marking is having place, and an optional phase. Once the necessary operators have been instantiated, the user must use the unitary class to create a unitary operator concatenating the building-block operators in the desired manner. The unitary object can then be fed with a quantum state vector, and returns the result of the quantum evolution. After that, in order to obtain the probability distributions, the measurement class must be used, choosing what register is being measured.

With the unitary and measurement classes, we could simulate the Szegedy quantum walk from any initial quantum state. However, this would result in a low-level implementation, since the user must code a loop for the different time steps, and manage to store the results properly. For that reason, we provide a higher-level simulator that does it for us. Our quantum simulator needs as input parameters the unitary evolution operator and the initial state $|\phi\rangle$, as well as the number of time steps and the register to measure. This simulator performs all the time steps while measuring the probabilities, and it only provides the probability distributions for all the time steps. Thus, the intermediate quantum states are not being stored, and the memory requirements do not scale with the number of steps.

Finally, in case we want to simulate the quantum evolution on different initial quantum states, our simulator allows the user to introduce a batch of quantum states, so that their evolutions are done in a vectorized form. An example is the case of calculating the evolution on a mixed state, where we need to simulate the evolution of all the states that comprise it. Moreover, once the simulation on this set of states is performed, the result can be fed to another utility that receives the coefficients in (9.30) and returns the probability distributions for the evolution of the mixed state.

9.5.2. Time and memory complexity

In Section 9.2.7, we proved that the theoretical scaling of both the memory requirements and time needed for the our algorithm to run scales as $\mathcal{O}(N^2)$. In order to check that our implementation follows that scaling, we have made some simulations for random dense transition matrices of growing sizes, from $N = 100$ to $N = 16000$. We have used an AMD Ryzen 9 5950X 16-Core Processor for all the simulations.

First, we have measured the time it takes to run the algorithm. We want to fit the data to the following scaling law:

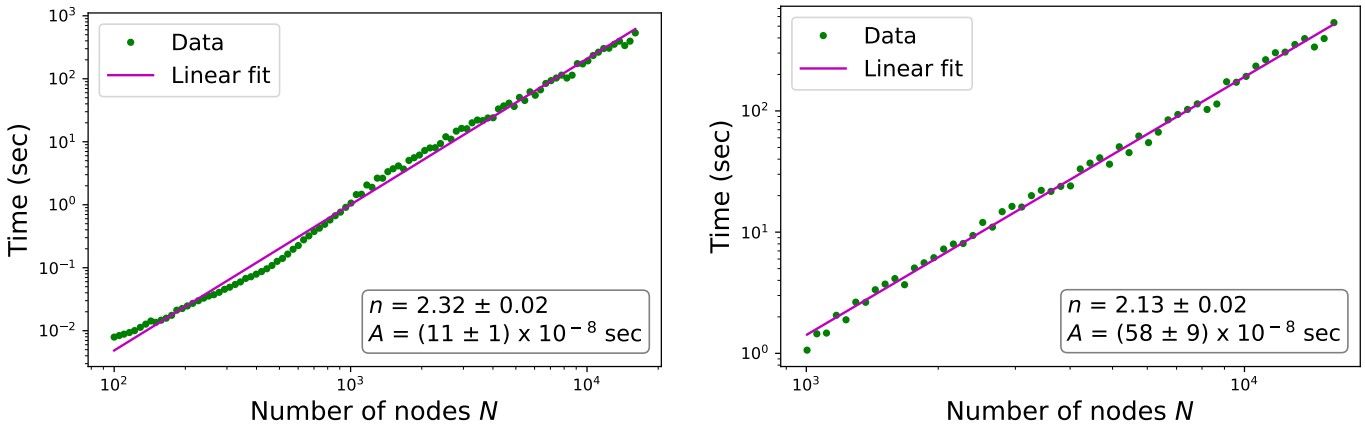
$$f(N) = AN^n, \tag{9.36}$$

so that the coefficient n determines the scaling complexity. To do so, we take logarithms in (9.36)

to linearize the expression:

$$\log f(N) = \log A + n \log N. \tag{9.37}$$

The results of the linear fit in logarithmic scale are shown in Figure 9.8(a). As we can see, the exponent n is significantly greater than 2. However, note that the linear behavior starts from $N = 1000$ approximately. Recall that the theoretical scaling was discussed in the asymptotic limit of large N. This can explain this behavior, and for this reason, we have made the fit in the linear region from $N = 1000$ in Figure 9.8(b). In this case now we can see a clear linear trend in logarithmic scale, with a smaller exponent of $n = 2.13$. Although this is slightly greater than 2, it seems that our implementation has a time complexity approximately similar to the theoretical one of $\mathcal{O}(N^2)$.



(a) (b)

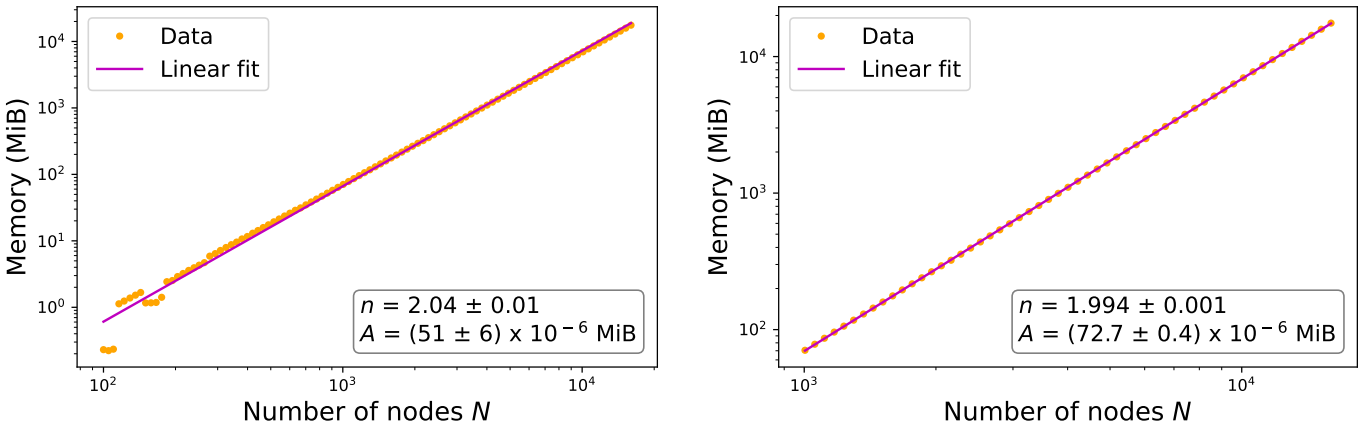
Figure 9.8: *Linear fit in logarithmic scale of the time needed to run the simulation algorithm versus the number of nodes of the graph. (a) From $N = 100$ to $N = 16000$. (b) From $N = 1000$ to $N = 16000$. All the simulations have been performed for 100 time steps, using the double unitary operator $W_s = S_w R S_w R$.*

For the scaling of the memory requirements, we have followed a similar procedure measuring the memory consumed by the algorithm. The results are shown in Figure 9.9(a). In this case, we have again finite size effects for small values of N. Thus, in Figure 9.9(b) we show the fit for larger values of N, where the asymptotic behavior occurs. The exponent $n \approx 2$ in this case, so that our simulator has the expected theoretical scaling with the memory resources as $\mathcal{O}(N^2)$.

9.5.3. High-level applications

With the main simulator described above, we can simulate any algorithm based on the Szegedy quantum walk. Nevertheless, we aim to provide higher-level functionalities in our library, so that end users can apply different Szegedy-based algorithms easier.

An application implemented in our library is the semiclassical Szegedy walk. We provide a function that manages to create the initial $|\psi_i\rangle$ states from a transition matrix G, and simulate the evolution on them using a unitary operator, to return the semiclassical matrices in (9.29) directly. Moreover, this function accepts a batch size parameter, so that the user can define the size of the batch of states that are vectorized at a time. Later, to simulate the walk, we also provide a classical



(a) (b)

Figure 9.9: *Linear fit in logarithmic scale of the memory consumed by the simulation algorithm versus the number of nodes of the graph. (a) From $N = 100$ to $N = 16000$. (b) From $N = 1000$ to $N = 16000$. All the simulations have been performed for 100 time steps, using the double unitary operator $W_s = S_w R S_w R$.*

walk simulator, so that it calculates a semiclassical walk using the desired semiclassical transition matrix.

Another application is the quantum PageRank algorithm described in Section 3.10. This is valuable not only as a quantum algorithm implementable on a quantum computer, but also as a quantum-inspired classical algorithm thought to be run on a classical computer [31,32]. Moreover, the transition matrix has no null elements. For these reasons, having an efficient classical implementation is crucial. In this case, the algorithm starts from the initial quantum superposition in (3.60), uses the double-step Szegedy unitary $W_s = S_w R S_w R$, and measures in the second register. After that, the probability distributions are averaged in time to yield the quantum PageRank distribution. Our function only requires the Google matrix G to perform all the calculations. Moreover, it allows the input of two arbitrary phase rotations, as introduced in Chapter 5.

Finally, there are applications for simulating the different SearchRank algorithms of Chapter 7. The user must provide the Google matrix G, and a list with the marked nodes. The applications calculate the reference time of measurement, apply the quantum or semiclassical walks for the unitary operator $W_Q = S_w Q_1 R S_w Q_1 R$, and return the SearchRank distributions from the second register.

9.6. Summary of Results and Conclusions

- Any Szegedy unitary operator can be decomposed on three building blocks: reflections, swaps and oracles. These operators are of size $N^2 \times N^2$, where N is the number of nodes of the graph, so that a naive approach for simulating this quantum walk would need memory resources that scale as $\mathcal{O}(N^4)$. There exist other methods that only requires $\mathcal{O}(N^3)$ memory resources when the transition matrix is dense, as the use of a sparse representation for the unitary operators, or the spectral decomposition method. However, for a general transition matrix this scaling is so much prohibitive for big enough networks.

- In order to save memory resources even if the transition matrix is dense, we have devised an algorithm that reduces the memory and time requirements further to $\mathcal{O}(N^2)$, and allows several modifications of the Szegedy unitary evolution as the introduction of oracles and complex phases extensions. Moreover, this scaling is optimal for dense transition matrices.

- We have also reviewed the semiclassical Szegedy walk and devised a method for simulating the quantum walk on mixed states. Both algorithms require simulating the walk on a set of different initial states, and we have shown how that procedure can be vectorized by constructing batches of quantum states. So far, for simulating on mixed states, our algorithm requires that they are diagonal in a given set, so that there are no coherence matrix elements. This is an open problem that needs a further research in the future.

- Finally, we have presented SQUWALS, a Python package for simulating Szegedy quantum walks using our optimized algorithm. This is a NumPy-based simulator that leverages vectorization techniques to speed up the calculations. We have coded a high-level functionality that only requires the initial state and the transition matrix to be provided by the user, and performs all the needed calculations to obtain the probability distributions at each time step of the quantum walk. Moreover, we have shown that both the time and memory requirements of our simulator scale as $\mathcal{O}(N^2)$, as theoretically predicted.

- There are instances of other classes of quantum walks on graphs that can be cast into a Szegedy quantum walk. For example, instances of coined quantum walks (see Section 3.5), or staggered quantum walks [149]. Thus,our simulator is also useful for simulating other kinds of quantum walks, whenever this casting is possible.

- In the future, we expect our algorithm to be parallelized with specialized hardware for that task, as for example GPU cards. Moreover, we aim to continue adding more higher-level algorithms to our library, having as yet algorithms for PageRank, SearchRank, and semiclassical walks simulation.

Conclusions

Chapter 10

Global Conclusions

In this thesis, we have explored new directions in quantum walk algorithms, which are the quantum equivalent of Markov chains. This opens a promising field for the development of algorithms that can beat the current classical approaches. Our objectives are divided in three blocks, for which we summarize the following results:

Phase Extensions

- We have introduced complex-phase extensions in the Szegedy quantum walk as new degrees of freedom, giving rise to the graph-phased Szegedy quantum walk. Our extended model increases the compatibility with the coined quantum walk and provides new strategies for search algorithms on graphs based on arbitrary phase rotations. Moreover, we have analyzed the implementation of this extended walk, modifying slightly the quantum circuits of the standard Szegedy walk.
- In the context of classifying information in future quantum networks, we have developed new quantum PageRank algorithms using arbitrary phase rotations. This approach shows improved results, as a restoring of the degeneracy of residual nodes in scale-free graphs. Moreover, it retains the good properties of the quantum PageRank, as for example an enhanced stability with respect to the damping parameter inherent to the algorithm. Therefore, it results in a novel paradigm for ranking nodes in a graph.

The Semiclassical Framework

- We have introduced a new type of walk algorithm that combines classical and quantum behaviors, which we denote as semiclassical walk. In particular, we have studied a semiclassical Szegedy walk, which has been proved to be a promising paradigm for algorithms beating both classical and quantum walks. As an example, we have analyzed its application for a new PageRank metric that can rank the nodes of symmetric weighted graphs, where both classical and quantum approaches fail. Furthermore, we have performed experiments on an actual quantum computer to prove its feasibility in a real scenario.
- We have applied the semiclassical framework to the quantum SearchRank algorithm, which is able to find marked nodes on graphs at the same time that provides a classification for them.

Whereas the quantum SearchRank looses its search ability as soon as the size of the graph grows, our new algorithm, denoted as semiclassical SearchRank, is able to restore it. Moreover, we have developed another approach, denoted as randomized SearchRank, which is a simplified algorithm retaining the quantum speedup. Moreover, the randomized version also improves the quality of the rankings, so that represents an important step towards a quantum search engine.

- In the context of blockchain technologies, we have developed the Quantum Signature Validation Algorithm for fraud detection. This protocol has in its core the randomized SearchRank, and therefore can leverage the graph structure of transactions for finding the most relevant fraudulent activity in the network. Our simulation results using a real Bitcoin dataset show that our algorithm is able to find all the tampered transactions, in an order that agrees with the classical PageRank distribution. Therefore, it can be helpful for solving scalability issues of blockchain and other distributed ledger technologies.

SQUWALS

- We have devised a novel classical simulation algorithm for the Szegedy quantum walk. Whereas previous simulation methods needed resources scaling as $\mathcal{O}(N^3)$ with the size N of the graph, our approach improves it to $\mathcal{O}(N^2)$. This allows the simulation of the walk on graphs with thousands of nodes, as the ones used in this thesis. Furthermore, we have developed SQUWALS, a Python package implementing our algorithm, including phase extensions and the semiclassical Szegedy walk. Apart from its usefulness at a research level, it can also be used for implementing quantum-inspired classical algorithms based on the Szegedy quantum walk.

Despite the positive results obtained in this thesis, it is obvious that more research is needed until a completely functional quantum algorithm is released. For example, our algorithms need further research focused on the construction of quantum circuits for oracles and walk operators. This thesis is just a step towards the advancement of quantum walks and their applications in tasks such as classification and search. We hope our findings pave the way for future developments bridging theoretical quantum computing research with practical implementations.

Appendices

Appendix A

Szegedy Dynamical Subspace and Spectral Decomposition

In this appendix, we introduce the dynamical subspace of the Szegedy quantum walk, and show the spectral decomposition of the unitary evolution operator on this subspace, useful for the simulation on a classical computer.

A.1. Dynamical Subspace

Without loss of generality, let us consider the single-step operator $U_s(\vec{\theta},\varphi)$ of the graph-phased Szegedy quantum walk introduced in Chapter 4, which contains link and APR phases:

$$U_s(\vec{\theta},\varphi) := S_w(2\Sigma(\vec{\theta},\varphi) - \mathbb{1}), \quad \Sigma(\vec{\theta},\varphi) := \frac{1}{2}\sum_{i=0}^{N-1}(1-e^{\mathrm{i}\theta_i})\,|\psi_i(\varphi)\rangle\,\langle\psi_i(\varphi)|\,, \tag{A.1}$$

with the $|\psi_i(\varphi)\rangle$ states given in (4.1). Recall that the standard Szegedy walk is recovered for $\theta_i = \pi$ and $\varphi_{ij} = 0$.

The dynamical subspace [31] is defined as:

$$\mathcal{H}_D(\varphi) := \mathrm{span}\left\{|\psi_i(\varphi)\rangle\,, S_w\,|\psi_i(\varphi)\rangle\right\}, \tag{A.2}$$

which depends on the link phases φ, but not on the APR phases $\vec{\theta}$. This subspace is invariant under the action of the evolution operator $U_s(\vec{\theta},\varphi)$, which acts as $-S_w$ in the orthogonal complement $\mathcal{H}_D^{\perp}(\varphi)$.

To prove that it is an invariant subspace, since the operator is unitary, it suffices to prove that $\mathcal{H}_D^{\perp}(\varphi)$ is invariant. Let us take a vector $|\phi\rangle \in \mathcal{H}_D^{\perp}$. Therefore this vector is perpendicular to the $|\psi_i(\varphi)\rangle$ states, so that $\Sigma(\vec{\theta},\varphi)\,|\psi_i(\varphi)\rangle = 0$. Thus:

$$U_s(\vec{\theta},\varphi)\,|\phi\rangle = -S_w\,|\phi\rangle\,. \tag{A.3}$$

By definition, $|\phi\rangle$ is also perpendicular to the $S_w|\psi_i(\varphi)\rangle$ states, so $S_w|\phi\rangle$ is perpendicular to the $|\psi_i(\varphi)\rangle$ states, and therefore also belongs to $\mathcal{H}_D^\perp(\varphi)$. Thus, the action of $U_s(\vec{\theta},\varphi)$ lets the dynamical subspace invariant, and effectively acts as $-S_w$ on the orthogonal complement.

Note that since the action in the orthogonal complement $\mathcal{H}_D^\perp(\varphi)$ is $-S_w$, the evolution in this subspace is not trivial and there is also some dynamics. The reason why the subspace $\mathcal{H}_D(\varphi)$ is named "dynamical" is historical, and comes from the use of the original double-step operator W_s. If we square the single-step operator, and apply it to a state in the orthogonal complement, we have

$$W_s(\vec{\theta},\varphi)|\phi\rangle = U_s^2(\vec{\theta},\varphi)|\phi\rangle = -U_s S_w|\phi\rangle = +S_w^2|\phi\rangle = |\phi\rangle\,, \tag{A.4}$$

so the action of $W_s(\vec{\theta},\varphi)$ is trivial and there is no dynamics. Moreover, even if we use the single operator $U_s(\vec{\theta},\varphi)$, the initial state is usually composed as a linear combination of the $|\psi_i(\varphi)\rangle$ states, so all the dynamics occurs in the dynamical subspace.

So far, we have studied a double-step operator $W_s(\vec{\theta},\varphi)$ coming from the square of a single-step operator $U_s(\vec{\theta},\varphi)$. Nevertheless, we also defined a double-step operator $W_s(\vec{\theta}_1,\varphi_1,\vec{\theta}_2,\varphi_2)$ coming from the product of operators with different phases. Since the dynamical subspace in (A.2) depends on the link phases φ, we must consider this case separately. The dynamical subspace for this operator is

$$\mathcal{H}_D(\varphi_1,\varphi_2) := \text{span}\left\{|\psi_i(\varphi_1)\rangle\,, S_w|\psi_i(\varphi_2)\rangle\right\}, \tag{A.5}$$

which again only depends on the link phases.

To prove it, we can get rid of the APR phases, since they do not intervene. Let us decompose the double-step operator as $W_s(\varphi_1,\varphi_2) = U_s(\varphi_2)U_s(\varphi_1)$. Since $|\phi\rangle \in \mathcal{H}_D^\perp(\varphi_1,\varphi_2)$ is perpendicular to the $|\psi_i(\varphi_1)\rangle$ states, then

$$U_s(\varphi_1)|\phi\rangle = -S_w|\phi\rangle\,. \tag{A.6}$$

Since $|\phi\rangle$ is also perpendicular to $S_w|\psi_i(\varphi_2)\rangle$, then $S_w|\phi\rangle$ is perpendicular to $|\psi_i(\varphi_2)\rangle$, so that

$$W_s(\varphi_1,\varphi_2)|\phi\rangle = -U_s(\varphi_2)S_w|\phi\rangle = +S_w^2|\phi\rangle = |\phi\rangle\,, \tag{A.7}$$

which proves that the dynamical subspace $\mathcal{H}_D(\varphi_1,\varphi_2)$ is invariant, and that the action on the orthogonal complement is trivial.

Note that the dimension of the dynamical subspace does not have to be $2N$. There can be cases where not all the vectors that generate it are linearly independent. For example, if there is a sink node, its corresponding $|\psi_i\rangle$ state is equal to its swapped version. Other cases occur when the graph has certain symmetries. Therefore, the dimension is at most $2N$, but can be smaller.

A.2. Spectral Decomposition

Since the action of any single-step operator $U_s(\vec{\theta},\varphi)$ in the orthogonal complement of the dynamical subspace is $-S_w$, its eigenvalues in this subspace are ± 1. Moreover, the action of any double-step $W_s(\vec{\theta}_1,\varphi_1,\vec{\theta}_2,\varphi_2)$ operator is trivial, so that its eigenvalues are $+1$. In our case we are interested in the spectral decomposition in the dynamical subspace $\mathcal{H}_D$, where the initial state of the quantum walk is usually, and where the evolution is not trivial. We show a method to obtain the eigenvalues

and eigenvectors from a decomposition of the discriminant matrix [31, 45, 90], which is an $N \times N$ matrix related to the transition matrix G. Thus, it can be decomposed numerically, and we can compute the spectral decomposition of the Szegedy evolution operator, avoiding the construction of the corresponding $N^2 \times N^2$ matrix.

The method to follow is different for each operator, so we treat them separately. Moreover, as we will see, these methods do not work in the case the operators have local APR phases, so we only consider the case of global APR phases.

A.2.1. Single-step Szegedy operator $U_s(\theta, \varphi)$

For the single-step operator we generalize a method based on the spectral decomposition of the discriminant matrix, which can be found in the literature for the standard Szegedy quantum walk [31, 90]. The evolution operator is

$$U_s(\theta,\varphi) := S_w(\left[1 - e^{\mathrm{i}\theta}\right]\Pi(\varphi) - \mathbb{1}), \quad \Pi(\varphi) := \sum_{i=0}^{N-1} |\psi_i(\varphi)\rangle \langle\psi_i(\varphi)| . \tag{A.8}$$

Let us define the discriminant matrix $D(\varphi)$ as an $N \times N$ matrix whose entries are

$$D_{ij}(\varphi) := e^{\mathrm{i}\varphi_{ji}} e^{-\mathrm{i}\varphi_{ij}} \sqrt{G_{ij}G_{ji}}, \tag{A.9}$$

We also define the operator $A(\varphi)$, which relates each vertex state to its corresponding $|\psi_i(\varphi)\rangle$ state:

$$A(\varphi) := \sum_{i=0}^{N-1} |\psi_i(\varphi)\rangle \langle i| . \tag{A.10}$$

From now own, for the sake of simplicity, we do not show the explicit dependence of these matrices on the complex phases, so that $\Pi \equiv \Pi(\varphi)$, $D \equiv D(\varphi)$, and $A \equiv A(\varphi)$. It is easy to check that these matrices satisfy the following properties:

$$AA^\dagger = \Pi, \quad A^\dagger A = \mathbb{1}_N, \quad D = A^\dagger S_w A. \tag{A.11}$$

The matrix D is Hermitian, so that it can be diagonalized, yielding N eigenvectors $|\lambda\rangle$ with eigenvalues λ, so that $D|\lambda\rangle = \lambda|\lambda\rangle$. With them, we define the vectors $\left|\widetilde{\lambda}\right\rangle := A|\lambda\rangle$, which are linear combinations of the $|\psi_i(\varphi)\rangle$ states by construction. We need to apply $U_s(\theta,\varphi)$ to these vectors. For convenience, let us calculate before the action of the projector. Using the properties in (A.11), we obtain:

$$\Pi\left|\widetilde{\lambda}\right\rangle = AA^\dagger\left|\widetilde{\lambda}\right\rangle = AA^\dagger A|\lambda\rangle = A\mathbb{1}_N|\lambda\rangle = \left|\widetilde{\lambda}\right\rangle . \tag{A.12}$$

Using this, we can apply the evolution operator:

$$U_s(\theta,\varphi)\left|\widetilde{\lambda}\right\rangle = -e^{\mathrm{i}\theta} S_w \left|\widetilde{\lambda}\right\rangle . \tag{A.13}$$

Now, we need to apply the evolution to the states $S_w\left|\widetilde{\lambda}\right\rangle$. We apply first the projector:

$$\Pi S_w\left|\widetilde{\lambda}\right\rangle = AA^\dagger S_w\left|\widetilde{\lambda}\right\rangle = AA^\dagger S_w A|\lambda\rangle = AD|\lambda\rangle = A\lambda|\lambda\rangle = \lambda\left|\widetilde{\lambda}\right\rangle , \tag{A.14}$$

and therefore we can calculate the unitary evolution:

$$U_s(\theta,\varphi)S_w\left|\widetilde{\lambda}\right\rangle = -\left|\widetilde{\lambda}\right\rangle + \left[1-e^{\mathrm{i}\theta}\right]\lambda S_w\left|\widetilde{\lambda}\right\rangle. \tag{A.15}$$

Note that the action on these vectors yield vectors in the subspace formed by them. Therefore, each pair $\left\{\left|\widetilde{\lambda}\right\rangle, S_w\left|\widetilde{\lambda}\right\rangle\right\}$ generates a 2-dimensional invariant subspace. Since the vectors $\left|\widetilde{\lambda}\right\rangle = A\left|\lambda\right\rangle$ are linear combinations of the $|\psi_i(\varphi)\rangle$ states, then the combination of all the vectors $\left|\widetilde{\lambda}\right\rangle$ and $S_w\left|\widetilde{\lambda}\right\rangle$ generate the dynamical subspace $\mathcal{H}_D(\varphi)$ in (A.2). Thus, these 2-dimensional invariant subspaces factorize the dynamical subspace. To find the spectral decomposition in $\mathcal{H}_D(\varphi)$, we just have to find the eigenvalues and eigenvectors in the subspaces generated by each pair $\left\{\left|\widetilde{\lambda}\right\rangle, S_w\left|\widetilde{\lambda}\right\rangle\right\}$. For each pair we have two eigenvectors, which we can form as linear combinations of the two vectors of the pair. Due to the fact that the product of an eigenvector by an scalar produces a parallel eigenvector, we can fix a coefficient to 1, and let the other be arbitrary. We make the following ansatz for the eigenvectors of $U_s(\theta,\varphi)$ in each 2-dimensional subspace:

$$\left|\mu\right\rangle = \left|\widetilde{\lambda}\right\rangle - aS_w\left|\widetilde{\lambda}\right\rangle, \tag{A.16}$$

where μ is the eigenvalue of the eigenvector $|\mu\rangle$, and a is a complex amplitude to determinate. The relative minus sign is chosen for convenience. We apply $U_s(\theta,\varphi)$ to this eigenvector, and using (A.13) and (A.15), we obtain:

$$U_s(\theta,\varphi)\left|\mu\right\rangle = a\left|\widetilde{\lambda}\right\rangle - \left(e^{\mathrm{i}\theta} + \left[1-e^{\mathrm{i}\theta}\right]a\lambda\right)S_w\left|\widetilde{\lambda}\right\rangle. \tag{A.17}$$

By definition of eigenvector we also have that

$$U_s(\theta,\varphi)\left|\mu\right\rangle = \mu\left|\widetilde{\lambda}\right\rangle - \mu a S_w\left|\widetilde{\lambda}\right\rangle. \tag{A.18}$$

Equaling both expressions, we obtain that $a=\mu$, and an equation for the eigenvalues:

$$-\mu^2 = -e^{\mathrm{i}\theta} - \left[1-e^{\mathrm{i}\theta}\right]\mu\lambda, \tag{A.19}$$

whose solution is

$$\mu = \frac{\left[1-e^{\mathrm{i}\theta}\right]\lambda \pm \sqrt{\left[1-e^{\mathrm{i}\theta}\right]^2\lambda^2 + 4e^{\mathrm{i}\theta}}}{2}. \tag{A.20}$$

This effectively yields (at most) two eigenvalues for each 2-dimensional invariant subspace, and we can calculate their corresponding eigenvectors.

Note that the vector $\left|\widetilde{\lambda}\right\rangle$ is trivially an eigenvector of the projector Π because it is a linear combination of the $|\psi_i(\varphi)\rangle$ states. For this reason, these states are not connected to each other. However, if we had local APR phases, the state $\left|\widetilde{\lambda}\right\rangle$ would not be an eigenvector of the pseudoprojector $\Sigma(\vec{\theta},\varphi)$, and the application of $U_s(\vec{\theta},\varphi)$ would result in a linear combination of all the $\left|\widetilde{\lambda}\right\rangle$ states. Thus, the dynamical subspace would not factorize in 2-dimensional invariant subspaces and we could not decompose the evolution operator using this method.

A.2.2. Double-step Szegedy operator $W_s(\theta_1, \varphi_1, \theta_2, \varphi_2)$

For the double-step Szegedy operator, in the case that it comes from the square of a single-step operator, such that $W_s(\theta, \varphi) = U_s^2(\theta, \varphi)$, the eigenvectors would be the same as before, and the eigenvalues would just be the squared ones, i.e., μ^2. However, if it comes from the product of two operators with different phases, we need to make the calculations directly for the operator $W_s(\theta_1, \varphi_1, \theta_2, \varphi_2)$:

$$W_s(\theta_1, \varphi_1, \theta_2, \varphi_2) := S_w(\left[1 - e^{\mathrm{i}\theta_2}\right] \Pi_2 - \mathbb{1})S_w(\left[1 - e^{\mathrm{i}\theta_1}\right] \Pi_1 - \mathbb{1}), \tag{A.21}$$

where $\Pi_1 \equiv \Pi(\varphi_1)$ and $\Pi_2 \equiv \Pi(\varphi_2)$.

Let us define the discriminant matrix $D(\varphi_1, \varphi_2)$ as an $N \times N$ matrix whose entries are

$$D_{ij}(\varphi_1, \varphi_2) := e^{\mathrm{i}(\varphi_1)_{ji}} e^{-\mathrm{i}(\varphi_2)_{ij}} \sqrt{G_{ij} G_{ji}}. \tag{A.22}$$

Again, we get rid of the dependence on the phases, so that $D \equiv D(\varphi_1, \varphi_2)$. We also define the matrices $A_1 \equiv A(\varphi_1)$ and $A_2 \equiv A(\varphi_2)$. These matrices satisfy the following properties:

$$A_1 A_1^\dagger = \Pi_1, \quad A_2 A_2^\dagger = \Pi_2, \quad A_1^\dagger A_1 = A_2^\dagger A_2 = \mathbb{1}_N, \quad D = A_2^\dagger S_w A_1, \quad D^\dagger = A_1^\dagger S_w A_2. \tag{A.23}$$

In the general case where $\varphi_1 \neq \varphi_2$, the matrix D not only is not Hermitian, neither is normal. Therefore, D and $D^\dagger$ do not share eigenvectors. As we will see, $D^\dagger$ appears in the calculus, so that unless they share eigenvectors, the spectral decomposition of D is not useful in this case. For that reason, we use a generalization of the method proposed by Szegedy using the singular value decomposition of D [45]. The matrix D has N singular values σ, whose right-singular and left-singular vectors are $|\sigma_R\rangle$ and $|\sigma_L\rangle$, respectively. They satisfy that $D|\sigma_R\rangle = \sigma|\sigma_L\rangle$, and $D^\dagger|\sigma_L\rangle = \sigma|\sigma_R\rangle$. Note that the singular values are always real numbers. We define the vectors $|\widetilde{\sigma}_R^1\rangle := A_1|\sigma_R\rangle$ and $|\widetilde{\sigma}_L^2\rangle := A_2|\sigma_L\rangle$, and follow an analogue procedure to the one used before. First, we apply the evolution operator to the states $|\widetilde{\sigma}_R^1\rangle$. For convenience, we calculate the following applications of the projectors Π_1 and Π_2 using the properties in (A.23):

$$\Pi_1 |\widetilde{\sigma}_R^1\rangle = A_1 A_1^\dagger |\widetilde{\sigma}_R^1\rangle = A_1 A_1^\dagger A_1 |\sigma_R\rangle = A_1 \mathbb{1}_N |\sigma_R\rangle = |\widetilde{\sigma}_R^1\rangle, \tag{A.24}$$

$$\Pi_2 S_w |\widetilde{\sigma}_R^1\rangle = A_2 A_2^\dagger S_w |\widetilde{\sigma}_R^1\rangle = A_2 A_2^\dagger S_w A_1 |\sigma_R\rangle = A_2 D |\sigma_R\rangle = A_2 \sigma |\sigma_L\rangle = \sigma |\widetilde{\sigma}_L^2\rangle. \tag{A.25}$$

With this, we can apply the evolution operator:

$$W_s(\theta_1, \varphi_1, \theta_2, \varphi_2) |\widetilde{\sigma}_R^1\rangle = e^{\mathrm{i}\theta_1} |\widetilde{\sigma}_R^1\rangle - e^{\mathrm{i}\theta_1} \left[1 - e^{\mathrm{i}\theta_2}\right] \sigma S_w |\widetilde{\sigma}_L^2\rangle. \tag{A.26}$$

After the application, the states $S_w|\widetilde{\sigma}_L^2\rangle$ appear. Therefore, we need to apply the evolution operator on them. Again, we precalculate some applications of the projectors:

$$\Pi_1 S_w |\widetilde{\sigma}_L^2\rangle = A_1 A_1^\dagger S_w |\widetilde{\sigma}_L^2\rangle = A_1 A_1^\dagger S_w A_2 |\sigma_L\rangle = A_1 D^\dagger |\sigma_L\rangle = A_1 \sigma |\sigma_R\rangle = \sigma |\widetilde{\sigma}_R^1\rangle, \tag{A.27}$$

$$\Pi_2 |\widetilde{\sigma}_L^2\rangle = A_2 A_2^\dagger |\widetilde{\sigma}_L^2\rangle = A_2 A_2^\dagger A_2 |\sigma_L\rangle = A_2 \mathbb{1}_N |\sigma_L\rangle = |\widetilde{\sigma}_L^2\rangle. \tag{A.28}$$

Using them, the evolution of the states results in

$$W_s(\theta_1, \varphi_1, \theta_2, \varphi_2) S_w |\widetilde{\sigma}_L^2\rangle = -\left[1 - e^{\mathrm{i}\theta_1}\right] \sigma |\widetilde{\sigma}_R^1\rangle + \left(\left[1 - e^{\mathrm{i}\theta_1}\right] \left[1 - e^{\mathrm{i}\theta_2}\right] \sigma^2 + e^{\mathrm{i}\theta_2}\right) S_w |\widetilde{\sigma}_L^2\rangle. \tag{A.29}$$

The same as before, each pair $\{|\widetilde{\sigma}_R^1\rangle, S_w|\widetilde{\sigma}_L^2\rangle\}$ generates a 2-dimensional invariant subspace, which factorize the dynamical subspace $\mathcal{H}_D(\varphi_1,\varphi_2)$ in (A.5). This is so because the vectors $|\widetilde{\sigma}_R^1\rangle = A_1|\sigma_R\rangle$ are linear combinations of the $|\psi_i(\varphi_1)\rangle$ states, and the vectors $|\widetilde{\sigma}_L^2\rangle = A_2|\sigma_L\rangle$ are linear combinations of the $|\psi_i(\varphi_2)\rangle$ states, so that the combination of all the vectors $|\widetilde{\sigma}_R^1\rangle$ and $S_w|\widetilde{\sigma}_L^2\rangle$ generate the dynamical subspace $\mathcal{H}_D(\varphi_1,\varphi_2)$. We make the following ansatz for the eigenvectors of $W_s(\theta_1,\varphi_1,\theta_2,\varphi_2)$ in each 2-dimensional subspace:

$$|\nu\rangle = \left|\widetilde{\sigma}_R^1\right\rangle - aS_w\left|\widetilde{\sigma}_L^2\right\rangle, \tag{A.30}$$

where ν is the eigenvalue of the eigenvector $|\nu\rangle$, and a is again a complex amplitude to determinate. We apply $W_s(\theta_1,\varphi_1,\theta_2,\varphi_2)$ to this eigenvector, and using (A.26) and (A.29), we obtain:

$$W_s(\theta_1,\varphi_1,\theta_2,\varphi_2)|\nu\rangle = \left(e^{\mathrm{i}\theta_1} + C_1a\sigma\right)\left|\widetilde{\sigma}_R^1\right\rangle - \left(ae^{\mathrm{i}\theta_2} + \left[e^{\mathrm{i}\theta_1} + C_1a\sigma\right]C_2\sigma\right)S_w\left|\widetilde{\sigma}_L^2\right\rangle, \tag{A.31}$$

where $C_k = \left[1 - e^{\mathrm{i}\theta_k}\right]$. Using the definition of eigenvector, we also have that

$$W_s(\theta_1,\theta_2)|\nu\rangle = \nu\left|\widetilde{\sigma}_R^1\right\rangle - \nu aS_w\left|\widetilde{\sigma}_L^2\right\rangle. \tag{A.32}$$

We obtain a system of two equations with two variables, a and ν:

$$\nu = e^{\mathrm{i}\theta_1} + C_1a\sigma, \tag{A.33}$$

$$\nu a = ae^{\mathrm{i}\theta_2} + \left[e^{\mathrm{i}\theta_1} + C_1a\sigma\right]C_2\sigma. \tag{A.34}$$

After substituting the first equation into the second one, we finally have a second-order equation for a:

$$C_1\sigma a^2 + \left[e^{\mathrm{i}\theta_1} - e^{\mathrm{i}\theta_2} - C_1C_2\sigma^2\right]a - e^{\mathrm{i}\theta_1}C_2\sigma = 0, \tag{A.35}$$

whose solution is

$$a = \frac{-\left[e^{\mathrm{i}\theta_1} - e^{\mathrm{i}\theta_2} - C_1C_2\sigma^2\right] \pm \sqrt{\left[e^{\mathrm{i}\theta_1} - e^{\mathrm{i}\theta_2} - C_1C_2\sigma^2\right]^2 + 4e^{\mathrm{i}\theta_1}C_1C_2\sigma^2}}{2C_1\sigma}. \tag{A.36}$$

Using (A.33) we obtain (at most) two eigenvalues for each subspace, and we can calculate the eigenvalues and eigenvectors of the operator $W_s(\theta_1,\varphi_1,\theta_2,\varphi_2)$ in the dynamical subspace $\mathcal{H}_D(\varphi_1,\varphi_2)$.

Finally, in the case that $\varphi_1 = \varphi_2$, the matrix D is Hermitian, and we could use its spectral decomposition. We would just have to take $\sigma = \lambda$, and $|\sigma_R\rangle = |\sigma_L\rangle = |\lambda\rangle$.

Bibliography

Bibliography

[1] S. A. Ortega and M. A. Martin-Delgado. Generalized Quantum PageRank Algorithm with Arbitrary Phase Rotations. *Physical Review Research*, 5:013061, 2023.

[2] S. A. Ortega and M. A. Martin-Delgado. Discrete-Time Semiclassical Szegedy Quantum Walks. *Physica A*, 625:129021, 2023.

[3] S. A. Ortega and M. A. Martin-Delgado. SQUWALS: A Szegedy QUantum WALks Simulator. *Advanced Quantum Technologies*, 7:2400022, 2024.

[4] S. A. Ortega and M. A. Martin-Delgado. Randomized SearchRank: A Semiclassical Approach to a Quantum Search Engine. *Physical Review Research*, 6:043014, 2024.

[5] S. A. Ortega and M. A. Martin-Delgado. Complex-Phase Extensions of the Szegedy Quantum Walk on Graphs. *Physical Review A*, 111:032216, 2025.

[6] J. Torres, S. A. Ortega, and M. A. Martin-Delgado. Quantum Signature Validation Algorithm for Efficient Detection of Tampered Transactions in Blockchain. *arXiv:2502.15023*, 2025.

[7] S. A. Ortega, P. Fernández, and M. A. Martin-Delgado. Implementing Semiclassical Szegedy Walks in Classical-Quantum Circuits for Homomorphic Encryption. *arXiv:2412.01966*, 2024.

[8] J. D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9:90–95, 2007.

[9] OriginLab Corporation. Origin(Pro), Version 2025, 1991.

[10] A. Hagberg, D. S. Chult, and P. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, 2008.

[11] A. Kay. Tutorial on the Quantikz Package. *arXiv:1809.03842*, 2018.

[12] Microsoft Corporation. Microsoft PowerPoint 2025, 1990.

[13] S. Rosen. Electronic Computers: A Historical Survey. *ACM Computing Surveys (CSUR)*, 1:7–36, 1969.

[14] R. Kaur, P. Kumar, and R. P. Singh. A Journey of Digital Storage From Punch Cards to Cloud. *IOSR Journal of Engineering*, 4:36–41, 2014.

[15] M. Riordan, L. Hoddeson, and C. Herring. The Invention of the Transistor. *Reviews of Modern Physics*, 71:S336, 1999.

[16] W. F. Brinkman, D. E. Haggan, and W. W. Troutman. A History of the Invention of the Transistor and Where It Will Lead Us. *IEEE Journal of Solid-State Circuits*, 32:1858–1865, 1997.

[17] R. K. Goyal. Exploring Quantum Materials and Applications: A Review. *Journal of Materials Science: Materials in Engineering*, 20:4, 2025.

[18] G. E. Moore. Cramming More Components Onto Integrated Circuits. *Electronics*, 38:114, 1965.

[19] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[20] J. R. Powell. The Quantum Limit to Moore's Law. *Proceedings of the IEEE*, 96:1247–1248, 2008.

[21] B. Schumacher. Quantum Coding. *Physical Review A*, 51:2738, 1995.

[22] R. P. Feynman. Simulating Physics with Computers. In *Feynman and Computation*, pages 133–153, 2018.

[23] R. P. Feynman. Quantum Mechanical Computers. *Foundations of Physics*, 16:507–532, 1986.

[24] D. Deutsch. Quantum Theory, the Church–Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 400:97–117, 1985.

[25] D. Deutsch and R. Jozsa. Rapid Solution of Problems by Quantum Computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439:553–558, 1992.

[26] L. K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 212–219, 1996.

[27] L. K. Grover. Quantum Mechanics Helps in Searching for a Needle in a Haystack. *Physical Review Letters*, 79:325–328, 1997.

[28] P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science, IEEE*, pages 124–134, 1994.

[29] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, 41:303–332, 1999.

[30] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[31] G. D. Paparo and M. A. Martin-Delgado. Google in a Quantum Network. *Scientific Reports*, 2:444, 2012.

[32] G. D. Paparo, Müller M., F. Comellas, and M. A. Martin-Delgado. Quantum Google in a Complex Network. *Scientific Reports*, 3:2773, 2013.

[33] IBM Quantum. https://quantum.ibm.com, 2021.

[34] F. Arute *et al.* Quantum Supremacy Using a Programmable Superconducting Processor. *Nature*, 574:505–510, 2019.

[35] J. Preskill. Quantum Computing in the NISQ Era and Beyond. *Quantum*, 2:79, 2018.

[36] D. Gottesman. An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation. In *Quantum Information Science and Its Contributions to Mathematics, Proceedings of Symposia in Applied Mathematics*, volume 68, pages 13–58, 2010.

[37] P. W. Shor. Scheme for Reducing Decoherence in Quantum Computer Memory. *Physical Review A*, 52:R2493, 1995.

[38] A. Berthiaume, D. Deutsch, and R. Jozsa. The Stabilisation of Quantum Computations. In *Proceedings Workshop on Physics and Computation. PhysComp'94, IEEE*, pages 60–62, 1994.

[39] Y. Aharonov, L. Davidovich, and N. Zagury. Quantum Random Walks. *Physical Review A*, 48:1687, 1993.

[40] R. Portugal. *Quantum Walks and Search Algorithms*. Springer, New York, 2013.

[41] A. A. Markov. Rasprostranenie Zakona Bol'shikh Chisel na Velichiny, Zavisyashchiye Drug ot Druga. *Izvestiya Fiziko-Matematicheskogo Obschestva Pri Kazanskom Universitete*, 15:18, 1906.

[42] F. Magniez, M. Santha, and M. Szegedy. Quantum Algorithms for the Triangle Problem. *SIAM Journal on Computing*, 37:413–424, 2007.

[43] A. Ambainis. Quantum Walk Algorithm for Element Distinctness. *SIAM Journal on Computing*, 37:210–239, 2007.

[44] N. Shenvi, J. Kempe, and K. B. Whaley. Quantum Random-Walk Search Algorithm. *Physical Review A*, 67:052307, 2003.

[45] M. Szegedy. Quantum Speed-up of Markov Chain Based Algorithms. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–41, 2004.

[46] J. Lemieux, B. Heim, D. Poulin, K. Svore, and M. Troyer. Efficient Quantum Walk Circuits for Metropolis-Hastings Algorithm. *Quantum*, 4:287, 2020.

[47] P. A. M. Casares, R. Campos, and M. A. Martin-Delgado. QFold: Quantum Walks and Deep Learning to Solve Protein Folding. *Quantum Science and Technology*, 7:025013, 2022.

[48] R. Campos, P. A. M. Casares, and M. A. Martin-Delgado. Quantum Metropolis Solver: A Quantum Walks Approach to Optimization Problems. *Quantum Machine Intelligence*, 5:28, 2023.

[49] G. Escrig, R. Campos, P. A. M. Casares, and M. A. Martin-Delgado. Parameter Estimation of Gravitational Waves with a Quantum Metropolis Algorithm. *Classical and Quantum Gravity*, 40:045001, 2023.

[50] G. Escrig, R. Campos, H. Qi, and M. A. Martin-Delgado. Quantum Bayesian Inference with Renormalization for Gravitational Waves. *The Astrophysical Journal Letters*, 979:L36, 2025.

[51] S. Giordano and M. A. Martin-Delgado. Quantum Algorithm for Testing Graph Completeness. *arXiv:2407.20069*, 2024.

[52] G. D. Paparo, V. Dunjko, A. Makmal, M. A. Martin-Delgado, and H. J. Briegel. Quantum Speedup for Active Learning Agents. *Physical Review X*, 4:031002, 2014.

[53] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum Computation by Adiabatic Evolution. *arXiv:quant-Ph/0001106*, 2000.

[54] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev. Adiabatic Quantum Computation Is Equivalent to Standard Quantum Computation. *SIAM Review*, 50:755–787, 2008.

[55] D. E. Deutsch. Quantum Computational Networks. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 425:73–90, 1989.

[56] A. C. C. Yao. Quantum Circuit Complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361, 1993.

[57] P. A. M. Dirac. *The Principles of Quantum Mechanics*. Oxford University Press, 1930.

[58] E. Schrödinger. An Undulatory Theory of the Mechanics of Atoms and Molecules. *Physical Review*, 28:1049, 1926.

[59] J. J. Sakurai and J. Napolitano. *Modern Quantum Mechanics*. Cambridge University Press, 2020.

[60] Y. Cao *et al*. Quantum Chemistry in the Age of Quantum Computing. *Chemical Reviews*, 119:10856–10915, 2019.

[61] J. von Neuman. Wahrscheinlichkeitstheoretischer Aufbau der Quantenmechanik. *Mathematisch-Physikalische Klasse*, 1927:245–272, 1927.

[62] D. Cohen. On Holy Wars and a Plea for Peace. *Computer*, 14:48–54, 1981.

[63] H. L. Huang, D. Wu, D. Fan, and X. Zhu. Superconducting Quantum Computing: A Review. *Science China Information Sciences*, 63:1–32, 2020.

[64] M. H. Devoret and J. M. Martinis. Implementing Qubits with Superconducting Integrated Circuits. *Experimental Aspects of Quantum Computing*, pages 163–203, 2005.

[65] J. L. O'brien. Optical Quantum Computing. *Science*, 318:1567–1570, 2007.

[66] P. Kok, W. J. Munro, K. Nemoto, T. C. Ralph, J. P. Dowling, and G. J. Milburn. Linear Optical Quantum Computing with Photonic Qubits. *Reviews of Modern Physics*, 79:135–174, 2007.

[67] J. I. Cirac and P. Zoller. Quantum Computations with Cold Trapped Ions. *Physical Review Letters*, 74:4091, 1995.

[68] L. Henriet *et al.* Quantum Computing with Neutral Atoms. *Quantum*, 4:327, 2020.

[69] A. Barenco *et al.* Elementary Gates for Quantum Computation. *Physical Review A*, 52:3457, 1995.

[70] A. Galindo and M. A. Martin-Delgado. Information and Computation: Classical and Quantum Aspects. *Reviews of Modern Physics*, 74:347, 2002.

[71] A. Peruzzo *et al.* A Variational Eigenvalue Solver on a Photonic Quantum Processor. *Nature Communications*, 5:4213, 2014.

[72] Y. Kwak, W. J. Yun, S. Jung, and J. Kim. Quantum Neural Networks: Concepts, Applications, and Challenges. In *2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN). IEEE*, pages 413–416, 2021.

[73] B. Zindorf and S. Bose. Efficient Implementation of Multi-Controlled Quantum Gates. *arXiv:2404.02279*, 2024.

[74] T. Roy, L. Jiang, and D. I. Schuster. Deterministic Grover Search with a Restricted Oracle. *Physical Review Research*, 4:L022013, 2022.

[75] B. O'Gorman, W. J. Huggins, E. G. Rieffel, and K. B. Whaley. Generalized Swap Networks for Near-Term Quantum Computing. *arXiv:1905.05118*, 2019.

[76] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan. On Universal and Fault-Tolerant Quantum Computing: A Novel Basis and a New Constructive Proof of Universality for Shor's Basis. In *40th Annual Symposium on Foundations of Computer Science, IEEE*, pages 486–494, 1999.

[77] A. Y. Kitaev. Quantum Computations: Algorithms and Error Correction. *Russian Mathematical Surveys*, 52:1191, 1997.

[78] N. J. Ross and P. Selinger. Optimal Ancilla-Free Clifford+T Approximation of Z-Rotations. *Quantum Information and Computation*, 16:901–953, 2016.

[79] D. Gottesman. The Heisenberg Representation of Quantum Computers. In *Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, pages 32–43, 1998.

[80] D. Gottesman. A Theory of Fault-Tolerant Quantum Computation. *Physical Review A*, 57:127, 1998.

[81] C. Yuan and M. Carbin. The T-Complexity Costs of Error Correction for Control Flow in Quantum Computation. *Proceedings of the ACM on Programming Languages*, 8:492–517, 2024.

[82] P. A. M. Casares, R. Campos, and M. A. Martin-Delgado. TFermion: A Non-Clifford Gate Cost Assessment Library of Quantum Phase Estimation Algorithms for Quantum Chemistry. *Quantum*, 6:768, 2022.

[83] Cirq. 10.5281/zenodo.4062499, 2018.

[84] A. Javadi-Abhari *et al.* Quantum Computing with Qiskit. *arXiv:2405.08810*, 2024.

[85] F. Pan, H. Gu, L. Kuang, B. Liu, and P. Zhang. Eficient Quantum Circuit Simulation by Tensor Network Methods on Modern GPUs. *ACM Transactions on Quantum Computing*, 5:1–26, 2024.

[86] V. V. Shende, S. S. Bullock, and I. L. Markov. Synthesis of Quantum Logic Circuits. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, pages 272–275, 2005.

[87] G. P. He. Two Examples of Implementation of the If/then/else Conditional Statement with Quantum Computers. *Proceedings of the Romanian Academy, Series A: Mathematics, Physics, Technical Sciences, Information Science*, 24:275–278, 2023.

[88] V. Bergholm, J. J. Vartiainen, M. Möttönen, and M. M. Salomaa. Quantum Circuits with Uniformly Controlled One-Qubit Gates. *Physical Review A*, 71:052330, 2005.

[89] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa. Quantum Circuits for General Multiqubit Gates. *Physical Review Letters*, 93:130502, 2004.

[90] A. Childs. Quantum Algorithms: LECTURE 14. Discrete-Time Quantum Walk. *University of Waterloo*, 2008.

[91] T. Loke and J. B. Wang. Efficient Quantum Circuits for Szegedy Quantum Walks. *Annals of Physics*, 382:64–84, 2017.

[92] B. L. Douglas and J. B. Wang. Efficient Quantum Circuit Implementation of Quantum Walks. *Physical Review A*, 79:052335, 2009.

[93] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum Amplitude Amplification and Estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[94] C. F. Chiang, D. Nagaj, and P. Wocjan. Efficient Circuits for Quantum Walks. *Quantum Information and Computation*, 10:420–434, 2010.

[95] L. K. Grover. Quantum Computers Can Search Rapidly by Using Almost Any Transformation. *Physical Review Letters*, 80:4329, 1998.

[96] G. L. Long, Y. S. Li, W. L. Zhang, and L. Niu. Phase Matching in Quantum Searching. *Physics Letters A*, 262:27–34, 1999.

[97] U. Schöning and J. Torán. *The Satisfiability Problem: Algorithms and Analyses*. Lehmanns Media, Berlin, 2013.

[98] M. DeCross, E. Chertkov, M. Kohagen, and M. Foss-Feig. Qubit-Reuse Compilation with Mid-Circuit Measurement and Reset. *Physical Review X*, 13:041057, 2023.

[99] T. T. Wu. Quantum Memory: Write, Read, Reset and Decoherence. In *Quantum Information and Computation. SPIE*, volume 5105, pages 204–215, 2003.

[100] P. Nation and B. Johnson. How to Measure and Reset a Qubit in the Middle of a Circuit Execution. *IBM Research Blog*, 2021.

[101] R. A. Santos. Szegedy's Quantum Walk with Queries. *Quantum Information Processing*, 15:4461–4475, 2016.

[102] H. Wang, J. Wu, X. Yang, P. Chen, and X. Yi. An Enhanced Quantum PageRank Algorithm Integrated with Quantum Search. In *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IEEE*, pages 74–81, 2014.

[103] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight Bounds on Quantum Searching. *Fortschritte der Physik: Progress of Physics*, 46:493–505, 1998.

[104] D. P. DiVincenzo. Quantum Computation. *Science*, 270:255–261, 1995.

[105] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and Weaknesses of Quantum Computing. *SIAM Journal on Computing*, 26:1510–1523, 1997.

[106] A. Ambainis. Understanding Quantum Algorithms via Query Complexity. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3265–3285, 2018.

[107] A. Galindo and M. A. Martin-Delgado. Family of Grover's Quantum-Searching Algorithms. *Physical Review A*, 62:062303, 2000.

[108] G. L. Long, L. Xiao, and Y. Sun. Phase Matching Condition for Quantum Search with a Generalized Initial State. *Physics Letters A*, 294:143, 2002.

[109] G. L. Long. Grover Algorithm with Zero Theoretical Failure Rate. *Physical Review A*, 64:022307, 2001.

[110] P. Li and S. Li. Phase Matching in Grover's Algorithm. *Physics Letters A*, 366:42–46, 2007.

[111] F. M. Toyama, W. van Dijk, Y. Nogami, M. Tabuchi, and Y. Kimura. Multiphase Matching in the Grover Algorithm. *Physical Review A*, 77:042324, 2008.

[112] T. J. Yoder, G. H. Low, and I. L. Chuang. Fixed-Point Quantum Search with an Optimal Number of Queries. *Physical Review Letters*, 113:210501, 2014.

[113] G. Brassard, P. Høyer, and A. Tapp. Quantum Counting. In *Automata, Languages and Programming: 25th International Colloquium, ICALP'98 Aalborg*, pages 820–831, 1998.

[114] M. Mosca. Counting by Quantum Eigenvalue Estimation. *Theoretical Computer Science*, 264:139–153, 2001.

[115] M. Mosca. Quantum Searching, Counting and Amplitude Amplification by Eigenvector Analysis. In *MFCS'98 Workshop on Randomized Algorithms*, pages 90–100, 1998.

[116] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum Algorithms Revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454:339–354, 1998.

[117] K. Pearson. The Problem of the Random Walk. *Nature*, 72:342–342, 1905.

[118] P. Mörters and Y. Peres. *Brownian Motion*, volume 30. Cambridge University Press, 2010.

[119] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

[120] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57:97–109, 1970.

[121] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[122] E. Farhi and S. Gutmann. Quantum Computation and Decision Trees. *Physical Review A*, 58:915, 1998.

[123] R. P. Feynman and A. R. Hibbs. *Quantum Mechanics and Path Integrals*. McGraw-Hill, New York, 1965.

[124] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous. One-Dimensional Quantum Walks. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 37–49, 2001.

[125] J. Kempe. Quantum Random Walks Hit Exponentially Faster. *Probability Theory and Related Fields*, 133:215–235, 2005.

[126] A. M. Childs. Universal Computation by Quantum Walk. *Physical Review Letters*, 102:180501, 2009.

[127] N. B. Lovett, S. Cooper, M. Everitt, M. Trevers, and V. Kendon. Universal Quantum Computation Using the Discrete-Time Quantum Walk. *Physical Review A*, 81:042330, 2010.

[128] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani. Quantum Walks on Graphs. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 50–59, 2001.

[129] J. Watrous. Quantum Simulations of Classical Random Walks and Undirected Graph Connectivity. *Journal of Computer and System Sciences*, 62:376–391, 2001.

[130] M. Sandbichler. The Relationship Between Coined and Szegedy-Type Quantum Walks. Master's thesis, University of Innsbruck, 2016.

[131] T. G. Wong. Equivalence of Szegedy's and Coined Quantum Walks. *Quantum Information Processing*, 16:1–15, 2017.

[132] T. G. Wong. Coined Quantum Walks on Weighted Graphs. *Journal of Physics A: Mathematical and Theoretical*, 50:475301, 2017.

[133] M. Szegedy. Spectra of Quantized Walks and a $\sqrt{\delta\varepsilon}$-Rule. *arXiv:quant-Ph/0401053*, 2004.

[134] H. Krovi, F. Magniez, M. Ozols, and J. Roland. Finding Is as Easy as Detecting for Quantum Walks. In *Automata, Languages and Programming: 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010*, pages 540–551, 2010.

[135] R. A. M. Santos and R. Portugal. Quantum Hitting Time on the Complete Graph. *International Journal of Quantum Information*, 8:881–894, 2010.

[136] D. A. Levin and Y. Peres. *Markov Chains and Mixing Times*, volume 107. American Mathematical Society, 2017.

[137] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Society for Industrial and Applied Mathematics, 2023.

[138] D. König. *Theorie der Endlichen und Unendlichen Graphen*. Chelsea Publishing Company, New York, 1936.

[139] D. B. West. *Introduction to Graph Theory*. Upper Saddle River: Prentice Hall, 2001.

[140] A. P. Riascos and J. L. Mateos. Random Walks on Weighted Networks: A Survey of Local and Non-Local Dynamics. *Journal of Complex Networks*, 9:cnab032, 2021.

[141] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.

[142] S. Brin, R. Motwami, L. Page, and T. Winograd. What Can You Do with a Web in Your Pocket? *IEEE Data Engineering Bulletin*, 21:37–47, 1998.

[143] L. Page, S. Brin, R. Motwami, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *Stanford InfoLab*, 1998.

[144] A. N. Langville and C. D. Meyer. *Google's Pagerank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2011.

[145] P. Ren, T. Aleksić, D. Emms, R. C. Wilson, and E. R. Hancock. Quantum Walks, Ihara Zeta Functions and Cospectrality in Regular Graphs. *Quantum Information Processing*, 10:405–417, 2011.

[146] A. Ambainis, J. Kempe, and A. Rivosh. Coins Make Quantum Walks Faster. In *Proceedings of the Sixteenth Annual Acm-Siam Symposium on Discrete Algorithms*, pages 1099–1108, 2005.

[147] T. D. Mackay, S. D. Bartlett, L. T. Stephenson, and B. C. Sanders. Quantum Walks in Higher Dimensions. *Journal of Physics A: Mathematical and General*, 35:2745, 2002.

[148] M. A. Martin-Delgado. Alan Turing and the Origins of Complexity. *Arbor*, 189:n6006, 2013.

[149] R. Portugal, R. A. Santos, T. D. Fernandes, and D. N. Gonçalves. The Staggered Quantum Walk Model. *Quantum Information Processing*, 15:85–101, 2016.

[150] A. M. Childs, E. Farhi, and S. Gutmann. An Example of the Difference Between Quantum and Classical Random Walks. *Quantum Information Processing*, 1:35–43, 2002.

[151] U. Senanayake, M. Piraveenan, and A. Zomaya. The Pagerank-Index: Going Beyond Citation Counts in Quantifying Scientific Impact of Researchers. *PLOS ONE*, 10:e0134794, 2015.

[152] J. Bollen, M. A. Rodriguez, and H. Van de Sompel. Journal Status. *Scientometrics*, 69:669–687, 2006.

[153] C. Bravo and M. Óskarsdóttir. Evolution of Credit Risk Using a Personalized Pagerank Algorithm for Multilayer Networks. In *Proceedings of the 3rd Machine Learning in Finance Workshop, 2020 ACM Knowledge Discovery in Databases Conference*, 2020.

[154] C. Frainay *et al.* MetaboRank: Network-Based Recommendation System to Interpret and Enrich Metabolomics Results. *Bioinformatics*, 35:274–283, 2019.

[155] D. Bánky, G. Iván, and V. Grolmusz. Equal Opportunity for Low-Degree Network Nodes: A Pagerank-Based Method for Protein Target Identification in Metabolic Graphs. *PLOS ONE*, 8:e54204, 2013.

[156] G. Iván and V. Grolmusz. When the Web Meets the Cell: Using Personalized PageRank for Analyzing Protein Interaction Networks. *Bioinformatics*, 27:405–407, 2011.

[157] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. WTF: The Who to Follow Service at Twitter. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 505–514, 2013.

[158] H. J. Kimble. The Quantum Internet. *Nature*, 453:1023–1030, 2008.

[159] D. Lancho, J. Martinez, D. Elkouss, M. Soto, and V. Martin. QKD in Standard Optical Telecommunications Networks. In *International Conference on Quantum Comunication and Quantum Networking*, pages 142–149, 2009.

[160] R. Valivarthi *et al.* Teleportation Systems Toward a Quantum Internet. *PRX Quantum*, 1:020317, 2020.

[161] O. Perron. Zur Theorie der Matrizen. *Mathematische Annalen*, 64:248–263, 1907.

[162] G. Frobenius. Über Matrizen Aus Nicht Negativen Elementen. *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften*, 23:456–477, 1912.

[163] R. Balu, C. Liu, and S. E. Venegas-Andraca. Probability Distributions for Markov Chains Based Quantum Walks. *Journal of Physics A: Mathematical and Theoretical*, 51:035301, 2017.

[164] A.-L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, 1999.

[165] T. Loke, J. W. Tang, J. Rodriguez, M. Small, and J. B. Wang. Comparing Classical and Quantum PageRanks. *Quantum Information Processing*, 16:25, 2017.

[166] A. M. Childs. On the Relationship Between Continuous and Discrete-Time Quantum Walk. *Communications in Mathematical Physics*, 294:581–603, 294.

[167] Y. Higuchi, N. Konno, I. Sato, and E. Segawa. Spectral and Asymptotic Properties of Grover Walks on Crystal Lattices. *Journal of Functional Analysis*, 267:4197–4235, 2014.

[168] Y. Higuchi and E. Segawa. The Spreading Behavior of Quantum Walks Induced by Drifted Random Walks on Some Magnifier Graph. *Quantum Information and Computation*, 17:399–414, 2017.

[169] E. Segawa and A. Suzuki. Spectral Mapping Theorem of an Abstract Quantum Walk. *Quantum Information Processing*, 18:333, 2019.

[170] A. Chan and H. Zhan. Pretty Good State Transfer in Discrete-Time Quantum Walks. *Journal of Physics A: Mathematical and Theoretical*, 56:165305, 2023.

[171] E. Segawa and Y. Yoshie. Quantum Search of Matching on Signed Graphs. *Quantum Information Processing*, 20:182, 2021.

[172] Y. Yoshie and K. Yoshino. A Quantum Searching Model Finding One of the Edges of a Subgraph in a Complete Graph. *Quantum Information Processing*, 21:222, 2022.

[173] Y. Higuchi, N. Konno, I. Sato, and E. Segawa. A Note on the Discrete-Time Evolutions of Quantum Walk on a Graph. *Journal of Math-for-Industry*, 5:103–109, 2013.

[174] H. Krovi, F. Magniez, M. Ozols, and J. Roland. Quantum Walks Can Find a Marked Element on Any Graph. *Algorithmica*, 74:851–907, 2016.

[175] P. Chawla, R. Mangal, and C. M. Chandrashekar. Discrete-Time Quantum Walk Algorithm for Ranking Nodes on a Network. *Quantum Information Processing*, 19:158, 2020.

[176] E. Sánchez-Burillo, J. Duch, J. Gómez-Gardeñes, and D. Zueco. Quantum Navigation and Ranking in Complex Networks. *Scientific Reports*, 2:605, 2012.

[177] H. Tang *et al.* TensorFlow Solver for Quantum PageRank in Large-Scale Networks. *Science Bulletin*, 66:120–126, 2021.

[178] P. Boito and R. Grena. Ranking Nodes in Directed Networks Via Continuous-Time Quantum Walks. *Quantum Information Processing*, 22:246, 2023.

[179] S. Dutta. Discrete-Time Open Quantum Walks for Vertex Ranking in Graphs. *Physical Review E*, 111:034312, 2025.

[180] S. Garnerone, P. Zanardi, and D. A. Lidar. Variational Quantum PageRank. *Physical Review Letters*, 108:230506, 2012.

[181] K. Wang, Y. Shi, L. Xiao, J. Wang, Y. N. Joglekar, and P. Xue. Experimental Realization of Continuous-Time Quantum Walks on Directed Graphs and Their Application in PageRank. *Optica*, 7:1524–1530, 2020.

[182] A.-L. Barabási, R. Albert, and H. Jeong. Scale-Free Characteristics of Random Networks: The Topology of the World-Wide Web. *Physica A: Statistical Mechanics and Its Applications*, 281:69–77, 2000.

[183] V. M. Eguiluz, D. R. Chialvo, G. A. Cecchi, M. Baliki, and A. V. Apkarian. Scale-Free Brain Functional Networks. *Physical Review Letters*, 94:018102, 2005.

[184] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The Large-Scale Organization of Metabolic Networks. *Nature*, 407:651–654, 2000.

[185] H. S. R. Rajula, M. Mauri, and V. Fanos. Scale-Free Networks in Metabolomics. *Bioinformation*, 14:140–144, 2018.

[186] K. Soramäki, M. L. Bech, J. Arnold, R. J. Glass, and W. E. Beyeler. The Topology of Interbank Payment Flows. *Physica A: Statistical Mechanics and Its Applications*, 379:317–333, 2007.

[187] G. Pandurangan, P. Raghavan, and E. Upfal. Using PageRank to Characterize Web Structure. In *International Computing and Combinatorics Conference*, pages 330–339, 2002.

[188] B. Bollobás, C. Borgs, J. T. Chayes, and O. Riordan. Directed Scale-Free Graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 3, pages 132–139, 2003.

[189] B. Georgeot, O. Giraud, and D. L. Shepelyansky. Spectral Properties of the Google Matrix of the World Wide Web and Other Directed Networks. *Physical Review E*, 81:056109, 2010.

[190] P. Erdős and A. Rényi. On Random Graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[191] W. W. Zhang *et al.* Quantum Versatility in PageRank. *Physical Review Research*, 6:043163, 2024.

[192] J. D. Whitfield, C. A. Rodríguez-Rosario, and A. Aspuru-Guzik. Quantum Stochastic Walks: A Generalization of Classical Random Walks and Quantum Walks. *Physical Review A*, 81:022323, 2010.

[193] A. Didi and E. Barkai. Measurement-Induced Quantum Walks. *Physical Review E*, 105:054108, 2022.

[194] B. Tregenna, W. Flanagan, R. Maile, and V. Kendon. Controlling Discrete Quantum Walks: Coins and Intitial States. *New Journal of Physics*, 5:83, 2003.

[195] A. W. Marshall, I. Olkin, and B. C. Arnold. *Inequalities: Theory of Majorization and Its Applications*. Springer, New York, Dordrecht, Heidelberg, London, 1979.

[196] J. Yi, P. Talkner, and G. L. Ingold. Approaching Infinite Temperature Upon Repeated Measurements of a Quantum System. *Physical Review A*, 84:032121, 2011.

[197] F. Acasiete, F. P. Agostini, J. K. Moqadam, and R. Portugal. Implementation of Quantum Walks on IBM Quantum Computers. *Quantum Information Processing*, 19:1–20, 2020.

[198] S. Tornow and K. Ziegler. Measurement Induced Quantum Walks on an IBM Quantum Computer. *Physical Review Research*, 5:033089, 2023.

[199] E. Biham and D. Kenigsberg. Grover's Quantum Search Algorithm for an Arbitrary Initial Mixed State. *Physical Review A*, 66:062301, 2002.

[200] D. Shapira, Y. Shimoni, and O. Biham. Algebraic Analysis of Quantum Search with Pure and Mixed States. *Physical Review A*, 71:042320, 2005.

[201] M. G. Kendall. A New Measure of Rank Correlation. *Biometrika*, 30:81–93, 1938.

[202] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *White Paper*, pages 1–9, 2008.

[203] Y. Chen. Blockchain Tokens and the Potential Democratization of Entrepreneurship and Innovation. *Business Horizons*, 61:567–575, 2018.

[204] V. Buterin. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. *White Paper*, pages 1–36, 2014.

[205] G. Wood. Polkadot: Vision for a Heterogeneous Multi-Chain Framework. *White Paper*, pages 1–21, 2016.

[206] H. Taherdoost. The Role of Blockchain in Medical Data Sharing. *Cryptography*, 7:36, 2023.

[207] H. Wu, S. Jiang, and J. Cao. High-Efficiency Blockchain-Based Supply Chain Traceability. *IEEE Transactions on Intelligent Transportation Systems*, 24:3748–3758, 2023.

[208] Z. Tian. Art Management in the Digital Era: Challenges and Opportunities. *Frontiers in Art Research*, 6:56–61, 2024.

[209] A. Li, G. Song, and T. Zhu. A Miniscule Survey on Blockchain Scalability. *arXiv:2212.13353*, 2022.

[210] M. Gandhi *et al.* Quantum Blockchain: Trends, Technologies, and Future Directions. *IET Quantum Communication*, 5:516–542, 2024.

[211] V. Fernandez, A. B. Orue, and D. Arroyo. Securing Blockchain with Quantum Safe Cryptography: When and How? In *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020)*, pages 371–379, 2021.

[212] F. M. Ablayev, D. A. Bulychkov, D. A. Sapaev, A. V. Vasiliev, and M. T. Ziatdinov. Quantum-Assisted Blockchain. *Lobachevskii Journal of Mathematics*, 39:957–960, 2018.

[213] K. P. Kalinin and N. G. Berloff. Blockchain Platform with Proof-of-Work Based on Analog Hamiltonian Optimisers. *arXiv:1802.10091*, 2018.

[214] Q. Li, J. Wu, J. Quan, J. Shi, and S. Zhang. Efficient Quantum Blockchain With a Consensus Mechanism QDPoS. *IEEE Transactions on Information Forensics and Security*, 17:3264–3276, 2022.

[215] D. Yaga, P. Mell, N. Roby, and K. Scarfone. Blockchain Technology Overview. *arXiv:1906.11078*, 2019.

[216] National Institute of Standards and Technology (NIST). Digital Signature Standard. Technical Report 186-4, Federal Information Processing Standards (FIPS), July 2013.

[217] F. Reid and M. Harrigan. An Analysis of Anonymity in the Bitcoin System. In *Security and Privacy in Social Networks*, pages 197–223, 2012.

[218] I. B. Damgård. Collision Free Hash Functions and Public Key Signature Schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 203–216, 1987.

[219] National Institute of Standards and Technology (NIST). Secure Hash Standard (SHS). Technical Report 180-4, Federal Information Processing Standards (FIPS), August 2015.

[220] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC, 2nd edition, 2014.

[221] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions On Information Theory*, 22:644–654, 1976.

[222] R. C. Merkle. *Secrecy, Authentication, and Public Key Systems*. PhD thesis, Stanford University, 1979.

[223] J. Wu, J. Liu, Y. Zhao, and Z. Zheng. Analysis of Cryptocurrency Transactions From a Network Perspective: An Overview. *Journal of Network and Computer Applications*, 190:103139, 2021.

[224] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[225] G. Di Battista, V. Di Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia. Bitconeview: Visualization of Flows in the Bitcoin Transaction Graph. In *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8, 2015.

[226] Y. Hu, S. Seneviratne, K. Thilakarathna, K. Fukuda, and A. Seneviratne. Characterizing and Detecting Money Laundering Activities on the Bitcoin Network. *arXiv:1912.12060*, 2019.

[227] L. Ermann, K. M. Frahm, and D. L. Shepelyansky. Google Matrix of Bitcoin Network. *The European Physical Journal B*, 91:1–13, 2018.

[228] H. D. Do and T. Do. PageRank and HodgeRank on Ethereum Transaction. *International Journal of Software Innovation (IJSI)*, 11:1–13, 2023.

[229] Z. Wu, J. Liu, J. Wu, Z. Zheng, and T. Chen. TRacer: Scalable Graph-Based Transaction Tracing for Account-Based Blockchain Trading Systems. *IEEE Transactions on Information Forensics and Security*, 18:2609–2621, 2023.

[230] Elliptic and C. Bellei. Elliptic Data Set. https://www.kaggle.com/datasets/ellipticco/elliptic-data-set, 2019.

[231] Elliptic. Blockchain Analytics & Crypto Compliance Solutions. https://www.elliptic.co, 2013.

[232] L. Baird. The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance. Technical report, Swirlds, 2016.

[233] E. Harris-Braun, N. Luck, and A. Brock. Holochain: Scalable Agent-Centric Distributed Computing. Technical report, Holochain Foundation, 2018.

[234] E. Harris-Braun, A. Brock, and P. d'Aoust. Holochain: Distributed Coordination by Scaled Consent, Not Global Consensus. Technical report, Holochain Foundation, 2024.

[235] F. L. Marquezino and R. Portugal. The QWalk Simulator of Quantum Walks. *Computer Physics Communications*, 179:359–369, 2008.

[236] P. Lara, A. Leao, and R. Portugal. Simulation of Quantum Walks Using HPC. In *Proceedings of the 3rd Conference of Computational Interdisciplinary Sciences*, pages 230–242, 2015.

[237] A. Glos and J. A. Miszczak. iitis/QuantumWalk.jl, 2018.

[238] C. R. Harris *et al.* Array Programming with NumPy. *Nature*, 585:357–362, 2020.