Quantum circuit design for quantum walks

Thomas Loke Oon Han, BSc (Adv. Sc.) (Hons)

THE UNIVERSITY OF WESTERN AUSTRALIA

This thesis is presented for the degree of Doctor of Philosophy of the University of Western Australia

School of Physics

2017

Thesis declaration

I, Thomas Loke Oon Han, certify that:

- This thesis has been substantially accomplished during enrolment in the degree.
- This thesis does not contain material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution.
- No part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of The University of Western Australia and where applicable, any partner institution responsible for the joint-award of this degree.
- This thesis does not contain any material previously published or written by another person, except where due reference has been made in the text.
- The work(s) are not in any way a violation or infringement of any copyright, trademark, patent, or other rights whatsoever of any person.
- This thesis contains published work and/or work prepared for publication, some of which has been co-authored.

Signature: 

Date:

Abstract

Quantum walks, as the quantum analogue of classical walks, have markedly different properties that have been exploited in quantum algorithms to provide speedups relative to classical algorithms. However, the physical implementation of a quantum walk usually involves a reformulation in terms of the quantum circuit model, in order for it to be realized efficiently. This thesis focuses on applying analytical and numerical techniques to the problem of designing quantum circuits for quantum walks of different types. We first present a review of the quantum circuit model and three different types of quantum walks (namely the coined quantum walk model, the Szegedy quantum walk model and the continuous-time quantum walk model), followed by a precise definition of efficiency for quantum circuit implementations of quantum walks. We then develop efficient quantum circuit implementations of continuous-time quantum walks using diagonalization of the Hamiltonian. Specifically, we find that circulant graphs and some classes of composite graphs can be implemented efficiently in this way. Next, for the Szegedy quantum walk model, we provide a theoretical framework for the development of efficient quantum circuits. We then apply the formalism to a wide variety of classes of graphs, including cyclic permutations, complete bipartite graphs, tensor product graphs, and weighted interdependent networks, and to the implementation of the quantum Pagerank algorithm. We also investigate the application and optimisation of numerical techniques (specifically, the cosine-sine decomposition) to the more general problem of implementing a unitary matrix. Lastly, we discuss design principles for the development of efficient quantum circuit implementations of quantum walks of different types, providing some intuition to how such implementations are derived.

Acknowledgements

Though it has been ~ 3.5 years since I began my doctoral studies, it feels like it has passed in the blink of an eye—so I suppose the old adage of 'time flies when you're having fun' is a true one. It certainly would not have been as enjoyable (or even possible) without the support of many individuals throughout this journey, so I owe each of them a debt which I cannot possibly repay, save with my gratitude.

Many thanks to my supervisor Prof. Jingbo Wang, with whom I've been working for ~ 7 years now. Her invaluable guidance, coupled together with her inexhaustible patience and near-constant availability for consultation, has made my studies a very fruitful one. Thanks also to my fellow doctoral candidate Josh Izaac, who has been of immense help in developing my research ideas, and for keeping my sanity in check with a lot of food and fellowship. Shoutouts also to our collaborators at the Centre for Quantum Photonics in the University of Bristol (particularly Xiaogang Qiang) for their hard work in experimentally realizing some of our proposals. In helping to keep me afloat financially during my studies, credit goes to an International Postgraduate Research Scholarship, Australian Postgraduate Award and the Bruce and Betty Green Postgraduate Research Top-Up Scholarship.

For a lot of love and support, my gratitude goes out to my family—in particular, to my parents in Penang, to my older brother Thad and his wife Naoko in Perth, to my younger sister Tania in Selangor, and to Aunt Ruby and Uncle Neville also in Perth. My thanks also go out to the wider family I have in Jesus Christ: to my small group (you guys are awesome!), to Ernest (for Tuesdays and more Tuesdays), to my friends (turning my frown upside down) and to the staff (you all do a great job!) at Subiaco Church of Christ, and to the friends that I still have around the world (you know who you are!). Thanks also to everyone else that I haven't named/categorized here but have been a part of my journey thus far. May the Lord keep you all.

Soli Deo Gloria

Authorship declaration

This thesis contains work that has been published or prepared for publication.

1. T. Loke, J.B. Wang, and Y.H. Chen, "*OptQC*: An optimised parallel quantum compiler", *Computer Physics Communications* 185.12 (2014), pp. 3307–3316

    - Location in thesis: Chapter 5
    - Student contribution to work: Wrote most of the content, excluding minor edits.

2. T. Loke, and J.B. Wang, "*OptQC* v1.3: An (updated) optimised parallel quantum compiler", *Computer Physics Communications* 207 (2016), pp. 531–532

    - Location in thesis: Chapter 5
    - Student contribution to work: Wrote most of the content, excluding minor edits.

3. X. Qiang[1], T. Loke[1], A. Montanaro, K. Aungskunsiri, X. Zhou, J.L. O'Brien, J.B. Wang, and J.C.F. Matthews, "Efficient Quantum Walk on a Quantum Processor", *Nature Communications* 7 (2016), p. 11511

    - Location in thesis: Chapter 3
    - Student contribution to work: Equal contribution with first author; one of the main contributors to the conception of the experiment, wrote the theory section on Quantum circuit for CTQW on circulant graph with contributions to the introduction and discussion sections of the paper.

4. T. Loke, and J.B. Wang, "Efficient quantum circuits for continuous-time quantum walks on composite graphs", *Journal of Physics A: Mathematical And Theoretical* 50 (2017), p. 055303

[1]Contributed equally to this work

- Location in thesis: Chapter 3
- Student contribution to work: Wrote most of the content, excluding minor edits.

5. T. Loke, and J.B. Wang, "Efficient quantum circuits for Szegedy quantum walks" (2017). Accepted for publication in *Annals of Physics*. Available at arXiv:1609.00173

    - Location in thesis: Chapter 4
    - Student contribution to work: Wrote most of the content, excluding minor edits.

The following paper is not directly related to the subject of the thesis, but was produced during candidature.

1. G.R. Carson, T. Loke, and J.B. Wang, "Entanglement dynamics of two-particle quantum walks", *Quantum Information Processing* 14.9 (2015), pp. 3193–3210

    - Location in thesis: Appendix A
    - Student contribution to work: Did the original study on which the first author expanded upon.

Student signature: 

Date:

I, Prof. Jingbo Wang, certify that the student statements regarding their contribution to each of the works listed above are correct.

Coordinating supervisor signature:

Date:

Contents

List of Figures

Chapter 1

Introduction

The discovery of Shor's algorithm for factoring large numbers in 1994 [1] has generated much interest in the field of quantum computation, as it holds the promise of solving problems that are considered intractable on a classical computer. Since then, applications of quantum computing in cryptography [2–4], linear algebra [5–7], graph theory [8–10], quantum simulation [11–16], and many other fields have been found. However, quantum algorithms are only useful in practice if we are able to realise them efficiently on a quantum computer.

Firstly, in order to define a computational process on a quantum computer, we need to choose a fixed model of computation that is universal, namely it can be used to express any given computational process in terms of basic elements. There are a myriad of different models of quantum computation, each useful for solving certain kind of problems and for ease-of-implementation on certain physical platforms. Some examples are the quantum circuit model [17–20], the adiabatic model [21–23], the topological model [24–26], quantum Turing machines [27, 28], and quantum walks [29–34], with the quantum circuit model being the most widely used [35].

Quantum walks have been useful for designing quantum algorithms that exhibit speedups over the best classical algorithms to address a wide array of problems, e.g. spatial searching [36], the graph isomorphism problem [8], traversal of a graph [32, 37], and network analysis [9, 10]. However, a fundamental issue with adopting quantum walks directly as a model of quantum computation is that the vertices in a quantum walk form the basis of the Hilbert space. This implies that as the size of the Hilbert space scales exponentially, the number of physical vertices required for the quantum walk also scales exponentially. This lack of scalability makes the

quantum walk model not directly useful as a physical model [35]. However, this can be avoided by reformulating a given quantum walk in terms of a different model of quantum computation, most notably the quantum circuit model [32, 38–44], which is used as a bridge to implement quantum walk-based algorithms.

There are two approaches to quantum circuit design for a given quantum walk (or quantum algorithms in general): analytical methods [32, 38–41, 43, 44] and numerical methods (more commonly referred to as quantum compilation) [42, 45–51]. Analytical methods provide an efficient quantum circuit implementation for a specific class of quantum walks. In contrast, numerical quantum compilation provides a quantum circuit implementation for any quantum walk, but it is rarely as efficient as analytical methods when applied to the same problem. Hence, there exists a trade-off between scope and efficiency between analytical methods and quantum compilation.

For analytical methods, the methodology (and definition of efficiency) tends to differ depending on the type of quantum walk being studied. There are two broad classes of quantum walks: discrete-time quantum walks (DTQWs) [29, 34, 52] and continuous-time quantum walks (CTQWs) [30], both taking place in a discrete position space. The CTQW, introduced by Farhi and Gutmann [30], is based on evolving a quantum state according to the Schrödinger equation. There are several different models used to define a DTQW. The most well-studied one is the coined quantum walk model introduced by Y. Aharonov *et al* [29], which utilises the tensor product of two Hilbert spaces (the vertex space and the coin space) as the state space. An alternative to the coined quantum walk model is the Szegedy quantum walk model introduced by M. Szegedy [33, 34], which under some conditions can be viewed as equivalent to the coined quantum walk model [53, 54]. A DTQW model (called the scattering quantum walk) based on analogy to optical inferometers has also been proposed by M. Hillery *et al* [52]. All of the models discussed have been applied to a wide variety of problems–for example, the CTQW model has been applied to the graph isomorphism problem [55, 56], whereas the Szegedy model for the discrete-time quantum walk has been used to determine the relative importance of nodes in a graph [9, 10]. In this thesis, we primarily study the discrete-time Szegedy walk model and the CTQW model.

A substantial amount of work has been done on implementing CTQWs efficiently using quantum circuits, typically considered under the more general prob-

lem of Hamiltonian simulation. Henceforth, except where mentioned, we will treat the problem of implementing CTQWs on graphs as being synonymous to Hamiltonian simulation, to avoid a mixture of terminology. Two classes of graphs are considered separately: sparse graphs [32, 57–63] and dense graphs [58, 64]. Most of the literature has focused on lowering the complexity of sparse graph implementation, although there have been specific non-sparse graphs that have been mentioned to have an efficient implementation [64, 65]. A major limitation on implementing CTQWs in general (sparse and non-sparse) is the no-fast-forwarding theorem [57, 64], which implies that, in general, the complexity of implementing CTQWs scales at least linearly with the time t. However, there still exists specific classes of graphs that can be simulated in sublinear or even constant time, i.e. graphs for which their time-evolution can be fast-forwarded, as pointed out in [64]. In chapter 3, we will examine some classes of graphs for which this is possible, giving explicit quantum circuit implementations for each class of graphs.

The Szegedy quantum walk model, having only been introduced slightly over a decade ago [33, 34], is not as well-studied compared to CTQWs or the coined DTQW model. Chiang *et al* [40], using results from Grover [66], developed quantum circuits for Szegedy walks corresponding to sparse transition matrices. In chapter 4, we consider families of Szegedy walks (for which the transition matrices are not necessarily sparse) for which the evolution operator can be realised efficiently using a quantum circuit.

For quantum compilation, there are two types of methods: approximate synthesis, and exact synthesis. Given some desired unitary operation U, approximate synthesis generates a quantum circuit that yields a unitary operator $\tilde{U}$ such that $\tilde{U}$ is within some distance ϵ from U. On the other hand, exact synthesis generates a quantum circuit that gives U exactly. For exact synthesis, the method of choice varies depending on whether we restrict ourselves to a finite set of universal gates [50, 67, 68] or are allowed to use any arbitrary set of quantum gates that are well-defined [19, 42, 45, 48, 69, 70]. In the latter category, Chen and Wang [48] recently developed a quantum compiler (called *Qcompiler*) that is based on the cosine-sine decomposition scheme [71, 72]. The gate count for the cosine-sine decomposition scales with the number of qubits n as $O(4^n)$ [73, 74] in general. That is, it scales exponentially with the number of qubits, which is highly undesirable. Hence, in chapter 5, we develop an optimised quantum compiler (called *OptQC*) based on

Qcompiler by using permutation matrices in such a way that reduces the overall cost of the final quantum circuit, on a case-by-case basis.

The remainder of the thesis is structured as follows. Chapter 2 outlines the mathematical framework for the quantum circuit model and the quantum walk model, as well as providing a more precise definition of the problem. We then investigate efficient quantum circuit implementations of CTQWs using Hamiltonian diagonalisation in chapter 3, which enables us to fast-forward simulation of some CTQWs to achieve a complexity independent of the time t parameter. Using this, we provide efficient implementations for the CTQW time-evolution operator on circulant graphs and some classes of composite graphs. In chapter 4, we introduce a theoretical framework for constructing efficient quantum circuit implementations of Szegedy quantum walks, which are formed by quantising classical Markov chains. In particular, we apply this framework to the class of Markov chains that possess cyclic symmetry, and then apply this scheme to simulate Szegedy walks used in the quantum Pagerank algorithm for some classes of non-trivial graphs. Next, in chapter 5, we discuss the task of constructing a generic quantum compiler using the cosine-sine decomposition and optimising the decomposition using permutation matrices. Lastly, we consider the broader question of design principles that govern efficient quantum circuit implementations for the different kinds of quantum walks in chapter 6. The thesis concludes in chapter 7.

Chapter 2

Theory

In this chapter, we provide a brief introduction to the quantum circuit model and the quantum walk model in section 2.1 and 2.2, respectively. Subsequent chapters will utilise the definitions to formulate quantum walks in terms of quantum circuits. We also provide a more precise statement of the problem in section 2.3.

2.1 Quantum circuits

Following [70], to satisfy the three postulates of quantum mechanics, there are three fundamental elements in a quantum circuit: qubits, quantum gates, and quantum measurements.

2.1.1 Qubits

Postulate 2.1. *Associated to any isolated physical system is a complex vector space $\mathcal{H}$ with inner product (that is, a Hilbert space) known as the state space of the system. The system is completely described by its state vector $|\psi\rangle$, which is a unit vector in the system's state space, i.e. $\langle\psi|\psi\rangle = 1$ and $|\psi\rangle \in \mathcal{H}$.*

Qubits (short for quantum bits) are the quantum analogue of classical bits, in that they both represent the state of the system under study. As with classical bits, there are two orthogonal basis states $\{|0\rangle, |1\rangle\}$ (called the computational basis states). The key difference is that qubits can exist in a complex superposition of basis states. For a single qubit, its state can always be written in the generic form $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, with the normalisation constraint $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Using matrix

notation, it is common to write this in (column) vector form as $|\psi\rangle = [\alpha_0, \alpha_1]^T$ where the basis states are $|0\rangle = [1, 0]^T$ and $|1\rangle = [0, 1]^T$.

For a sequence of n qubits, there are 2^n distinct computational basis states. Notationally, the computational basis state $|s_1 s_2 \dots s_n\rangle$ denotes the basis state with the first qubit in the state $|s_1\rangle$, the second qubit in the state $|s_2\rangle$, and so on. If we have sequence of n qubits with states $|\psi_1\rangle, |\psi_2\rangle, \dots, |\psi_n\rangle$ respectively, we can collectively write the state as:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots \otimes |\psi_n\rangle, \tag{2.1}$$

where $\otimes$ denotes the tensor product. However, this only applies to a sequence of qubits with separable states, i.e. the state of one qubit is not dependent on the state of any other qubit. For example, the general state of a 2-qubit sequence is:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle, \tag{2.2}$$

where $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ is the computational basis. The general form of a separable state in a 2-qubit sequence is thus:

$$|\psi_{sep}\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle) \tag{2.3}$$

$$= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle. \tag{2.4}$$

However, the Bell state [75]:

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right), \tag{2.5}$$

cannot be written as a tensor product of single qubit states, so it is not a separable state. In the state $|\Phi^+\rangle$, a measurement of $|0\rangle$ for the first qubit implies that we will necessarily measure $|0\rangle$ for the second qubit. Similarly, if we measure $|1\rangle$ for the first qubit we will necessarily measure $|1\rangle$ for the second qubit.

In a physical system, a qubit can be realised by any quantum system with two orthogonal states. Some examples are: energy levels of a quantum simple harmonic oscillator [76], location or polarisation of a single photon [77, 78], atomic spin [79–81], and spin states of quantum dots [82].

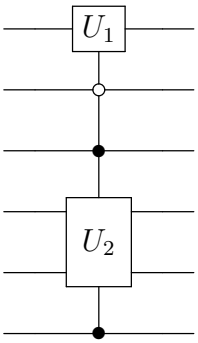


Figure 2.1: Example of a controlled unitary gate on 6 qubits.

2.1.2 Quantum gates

Postulate 2.2. *The evolution of a closed quantum system is described by a unitary transformation U, i.e. $U^\dagger U = I$, where $U^\dagger$ denotes the conjugate transpose of U. That is, the initial state $|\psi\rangle$ of the system is evolved to the new state $|\psi'\rangle$ by applying unitary operator U, i.e. $|\psi'\rangle = U|\psi\rangle$.*

Quantum gates are the quantum analogue of logic gates in classical computing, in that they are both used to evolve the state of the system in some prescribed fashion. As a consequence of the unitarity requirement in Postulate 2.2, all quantum gates (and therefore any quantum computation aside from measurement, see section 2.1.3) are reversible, since the inverse operation of a quantum gate is given by its conjugate transpose. There are a number of quantum gates that are commonly used in the literature [17, 19, 20]. Table 2.1 and 2.2 provides a list of these gates, with their corresponding diagrammatic representation and matrix representation of the operator.

The CNOT and Toffoli gate in Table 2.2 are examples of the general class of controlled unitary gates. There are two sets of qubits in a controlled unitary gate: the set of control qubits and the set of target qubits. The controlled unitary gate applies a specified unitary operation to the set of target qubits, conditional on the control qubits being in a specified state—if the condition is not met, then the controlled unitary gate leaves the target qubits unchanged. In the example shown in Figure 2.1, the unitary operations U_1 and U_2 are applied to qubits 1 and 3–4 respectively if qubit 2 is in the $|0\rangle$ state and qubits 3 and 6 are in the $|1\rangle$ state.

In a quantum circuit, quantum gates in the same column are composed using the tensor product $\otimes$, in the top to bottom order of the gates. By convention, time flows from left to right in a quantum circuit, so the first column is applied first,

Gate name	Symbol	Diagram	Matrix representation
Identity gate	I_2		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
NOT gate or Pauli-x gate	NOT or σ_x		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-y gate	σ_y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
π gate or Pauli-z gate	π or σ_z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard gate	H		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Rotation gate	$R_y(\theta)$		$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$
Phase gate	$R(\phi)$ or $R_z(\phi)$		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$
General phase gate	$R(\phi_1, \phi_2)$		$\begin{bmatrix} e^{i\phi_1} & 0 \\ 0 & e^{i\phi_2} \end{bmatrix}$

Table 2.1: Single qubit gates.

the second column applied second, and so on. Figure 2.2 shows an example of a quantum circuit with three columns, for which the unitary operator U for the whole circuit is given by:

$$U = T \cdot (S \otimes I_2) \cdot (\text{CNOT} \otimes H)\,. \tag{2.6}$$

2.1.3 Quantum measurements

Postulate 2.3. *Quantum measurements are described by a collection* $\{M_m\}$ *of measurement operators satisfying the completeness relation* $\sum_m M_m^\dagger M_m = I$. *These are operators acting on the state space* $\mathcal{H}$ *of the system being measured. The index* m

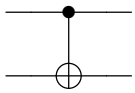
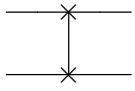
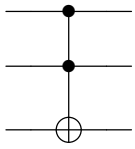
Gate name	Symbol	Diagram	Matrix representation
Controlled-NOT gate	CNOT		$\begin{bmatrix} I_2 & 0 \\ 0 & \sigma_x \end{bmatrix}$
SWAP gate	S		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \sigma_x & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Toffoli gate	T		$\begin{bmatrix} I_6 & 0 \\ 0 & \sigma_x \end{bmatrix}$

Table 2.2: Multiple qubit gates (matrices written in block-matrix form).

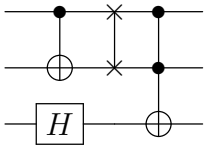


Figure 2.2: An example of a quantum circuit.

refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ *immediately before the measurement then the probability that result* m *occurs is given by* $p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle$ *and the state of the system after the measurement is* $\frac{M_m|\psi\rangle}{\sqrt{p(m)}}$.

Upon measurement of a single qubit with state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ in the computational basis (i.e. using the measurement operators $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$), the qubit collapses into the $|0\rangle$ state with probability $|\alpha_0|^2$, or into the $|1\rangle$ state with probability $|\alpha_1|^2$—hence the necessity of the normalisation requirement $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Note that, unlike quantum gates, measurement operators are not unitary operators, since they do not perform a 1-to-1 operation—as such, they are also not time-reversible. Figure 2.3 shows the diagrammatic representation of a measurement gate in a quantum circuit.

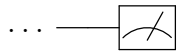


Figure 2.3: Measurement gate in a quantum circuit.

2.2 Quantum walks

As outlined in chapter 1, we primarily study the CTQW model and the (discrete-time) Szegedy walk model in this thesis. In order to define either model, we introduce some graph theory preliminaries below in section 2.2.1. Proper definitions for the CTQW model and the Szegedy quantum walk model are then provided in sections 2.2.2 and 2.2.3 respectively. For completeness, we also introduce the coined quantum walk in section 2.2.4, which we will discuss further in chapter 6. Further detail regarding efficient quantum circuit implementation can be found in [39] and [41], the latter of which is reproduced in Appendix A.

2.2.1 Graph theory

A graph has two basic elements: vertices that represent discrete states and edges representing connections between vertices. We denote a graph G as $G(V, E)$, where $V = \{v_1, \ldots, v_N\}$ and $E = \{(v_i, v_j), \ldots, (v_k, v_l)\}$ are the vertex and edge set of the graph respectively. Any graph is fully described by its N-by-N adjacency matrix A, which is defined as:

$$A_{i,j} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

A graph G is undirected if its adjacency matrix is symmetric ($A = A^T$), i.e. if $(v_i, v_j) \in E$ then necessarily $(v_j, v_i) \in E$ also. Directed graphs, on the other hand, do not have a symmetric adjacency matrix ($A \neq A^T$). Weighted graphs are a generalisation of equation (2.7), allowing for any real value in any entry of the adjacency matrix. For an undirected graph G, the degree d_i of a vertex v_i is the number of undirected edges connected to v_i, that is:

$$\deg(A)_i = d_i = \sum_{j=1}^{N} A_{j,i}. \tag{2.8}$$

For any undirected graph, the Laplacian L of the graph is defined as $L = D - A$, where D is the diagonal matrix with $D_{i,i} = \deg(A)_i$. An undirected graph is called a degree-regular graph with degree d iff every vertex in the graph has the same degree d, i.e. $\deg(A)_i = d \; \forall i \in \{1, \ldots, N\}$. For directed graphs, there are two types of degree: in-degree and out-degree, which are the number of edges pointing towards

and away from a vertex respectively. Mathematically, this becomes:

$$\text{indeg}(A)_i = \sum_{j=1}^{N} A_{j,i} \tag{2.9}$$

$$\text{outdeg}(A)_i = \sum_{j=1}^{N} A_{i,j}. \tag{2.10}$$

2.2.2 Continuous-time quantum walks

The CTQW model, introduced by Farhi and Gutmann in 1998 [30], is a continuous-time (i.e. $t \in \mathbb{R}$) model for quantum walks. The state of the system in a CTQW is described by a state vector $|\psi(t)\rangle$ in the Hilbert space $\mathcal{H}$ of dimension N spanned by the orthonormal basis states $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$ corresponding to vertices in the graph. Hence, an arbitrary state vector $|\Psi\rangle \in \mathcal{H}$ can be written as $|\Psi\rangle = \sum_{i=0}^{N-1} a_i |i\rangle$, where $a_i \in \mathbb{C}$. The time-evolution of the state $|\psi(t)\rangle$ is governed by the time-dependent Schrödinger equation:

$$i\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle, \tag{2.11}$$

where the Hamiltonian H is a Hermitian operator, i.e. $H = H^\dagger$. The formal solution to this equation is $|\psi(t)\rangle = U(t)|\psi(0)\rangle$, where:

$$U(t) = \exp(-itH), \tag{2.12}$$

is the time-evolution operator, which is what we want to implement efficiently using the quantum circuit model of computation. In order to perform a CTQW on a given graph, we choose the Hamiltonian H to be directly related to the graph. Choice of H varies in the literature between $H = \gamma(D - A)$ [37, 83] and $H = \gamma A$ [30, 84], where γ is the hopping rate per edge per unit time and D is an N-by-N (diagonal) degree matrix as defined in section 2.2.1. For degree-regular graphs, the only difference between the two choices is a global phase factor and a sign flip in t, which does not change observable quantities [85]. However, the two choices will result in different dynamics for non-degree-regular graphs [36]. In this thesis, we use $H = \gamma A$. The Hermiticity requirement of H implies that the adjacency matrix A of the graph G has to be symmetric, i.e. G must be an undirected graph for the CTQW model.

2.2.3 Szegedy quantum walks

The Szegedy quantum walk model, introduced by M. Szegedy in 2004 [33, 34], is a discrete-time model for quantum walks. Unlike the CTQW and the coined quantum walk model, it can also simulate quantum walks on directed and weighted graphs, in addition to undirected graphs. Szegedy walks are obtained by quantising a given classical Markov chain. A classical Markov chain is comprised of a sequence of random variables X_t for $t \in \mathbb{Z}_+$ with the property that $P(X_t|X_{t-1}, X_{t-2}, ..., X_1) = P(X_t|X_{t-1})$, i.e. the probability of each random variable is only dependent on the previous one. Suppose that every random variable X_t has N possible states, i.e. $X_t \in \{s_1, \ldots, s_N\}$. If the Markov chain is time-independent, that is:

$$P(X_t|X_{t-1}) = P(X_{t'}|X_{t'-1}) \ \forall t, t' \in \mathbb{Z}_+, \tag{2.13}$$

then the process can be described by a single (left) stochastic matrix P (called the transition matrix) of dimension N-by-N, where $P_{i,j} = P(X_t = s_i|X_{t-1} = s_j)$ is the transition probability of $s_j \to s_i$ with the column-normalisation constraint $\sum_{i=1}^{N} P_{i,j} = 1$.

A random walk on a graph G can be described by a Markov chain, with states of the Markov chain corresponding to vertices on the graph, and edges of the graph determining the transition matrix. Given the adjacency matrix A of the graph (as defined in equation (2.7)), we can define the corresponding transition matrix for G as:

$$P_{i,j} = A_{i,j}/\text{indeg}(j). \tag{2.14}$$

Szegedy's method for quantising a single Markov chain with transition matrix P starts by considering a Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ composed of two registers of dimension N each. A state vector $|\Psi\rangle \in \mathcal{H}$ of dimension N^2 can thus be written in the form $|\Psi\rangle = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} a_{i,j}|i,j\rangle$, where $a_{i,j} \in \mathbb{C}$. The projector states of the Markov chain are defined as:

$$|\psi_i\rangle = |i\rangle \otimes \sum_{j=0}^{N-1} \sqrt{P_{j+1,i+1}}|j\rangle \equiv |i\rangle \otimes |\phi_i\rangle, \tag{2.15}$$

for $i \in \{0, \ldots, N-1\}$. We can interpret $|\phi_i\rangle$ to be the square-root of the $(i+1)$th column of the transition matrix P. Note that $\langle\psi_{i'}|\psi_i\rangle = \delta_{i,i'}$ due to the orthonormal-

ity of basis states and the column-normalisation constraint on P. The projection operator onto the space spanned by $\{|\psi_i\rangle : i \in \{0, \ldots, N-1\}\}$ is then:

$$\Pi = \sum_{i=0}^{N-1} |\psi_i\rangle\langle\psi_i|, \tag{2.16}$$

and the associated reflection operator $\mathcal{R} = I - 2\Pi$. Define also the swap operator that interchanges the two registers as:

$$\mathcal{S} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |i,j\rangle\langle j,i|, \tag{2.17}$$

which has the property $\mathcal{S}^2 = I$. Then the walk operator for a single step of the Szegedy walk is given by:

$$U_{\text{walk}} = \mathcal{S}(I - 2\Pi) = \mathcal{S}\mathcal{R}. \tag{2.18}$$

Here, we adopt $\mathcal{R} = I - 2\Pi$ here instead of the conventional $\mathcal{R} = 2\Pi - I$. The difference is simply an overall phase of -1, but it helps us to avoid some sign issues in chapter 3. If we start in the state $|\psi_0\rangle$, then applying the step operator $t \in \mathbb{Z}_+$ times gives us the state of the system at time t, i.e.

$$|\psi(t)\rangle = U_{\text{walk}}^t |\psi_0\rangle. \tag{2.19}$$

We note here that these definitions are essentially a special case of the bipartite walks framework used by Szegedy originally [33, 34]. The above can be put into the same framework by considering the two-step walk operator $U_{\text{walk}}^2 = (2S\Pi S - I)(2\Pi - I)$, which is equal to Szegedy's bipartite walk operator using equivalent reflections in both spaces. We use the above definitions since, for an undirected graph, one step of U_{walk} is equivalent to one step of a coined quantum walk using the Grover coin operator at every vertex as the coin operator [53], which we define later in equation (2.20). However, unlike the coined quantum walk formalism, the Szegedy walk as defined above provides a unitary time-evolution operator even for directed and weighted graphs.

2.2.4 Coined quantum walks

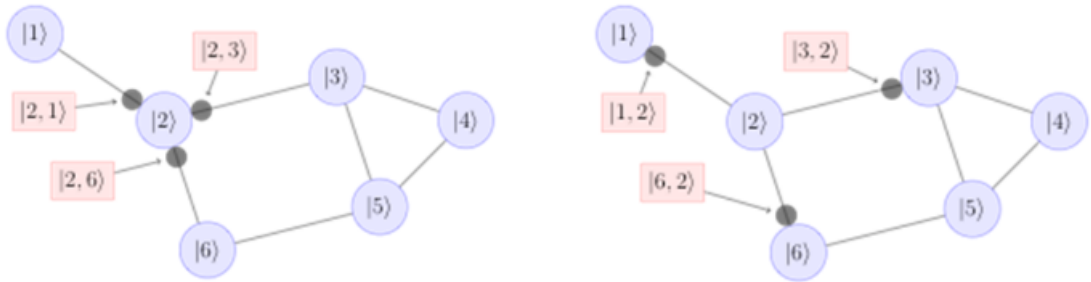
The coined quantum walk model, introduced by Aharonov *et al.* in 1993 [29], is the earliest and most well-studied discrete-time model for quantum walks. The state of

the system in a coined quantum walk is described by the combination of a position Hilbert space $\mathcal{H}_P$ and a coin Hilbert space $\mathcal{H}_C$, i.e. $|\psi\rangle \in \mathcal{H}$ where $\mathcal{H} = \mathcal{H}_P \otimes \mathcal{H}_C$.

For a given undirected graph G, the coined quantum walk on G acts on the vertex-coin space, consisting of states $|v_i, c_i\rangle \in \mathcal{H}_P \otimes \mathcal{H}_C$ where $1 \leq c_i \leq d_i$ (d_i is the degree of the vertex, as defined in equation (2.8)). That means that on a graph, the coin space essentially corresponds to the (outgoing) edges that are connected to a vertex. To propagate the coined quantum walk by a single step, we perform two operations: a coin toss operation C and a shifting operation S. The coin toss operation C essentially mixes the coin states at each vertex v_i using a prescribed (unitary) coin operator C_{v_i} of dimension d_i-by-d_i. In principle, the coin operators C_{v_i} can be chosen arbitrarily for every vertex, but in practice, one usually selects these operators in some regular fashion. The two most common choices for coin operations on N dimensions are the Grover coin G_N and the DFT coin DFT_N, which are defined by:

$$(G_N)_{i,j} = \frac{2}{N} - \delta_{i,j} \quad \text{and} \quad (\mathrm{DFT}_N)_{i,j} = \frac{1}{\sqrt{N}} e^{2\pi i j/N}, \tag{2.20}$$

where δ is the Kronecker delta function. The Grover coin is biased (not all directions are equal in magnitude) but symmetric (labels on directions can be interchanged without changing the coin operator), whereas the DFT coin is unbiased but asymmetric [86]. The shifting operator S swaps the coin states connected by an edge with its action defined by $S|v_i, c_i\rangle = |c_i, v_i\rangle$, and $S^2 = I$. We illustrate both operations in Figure 2.4.



(a) (b)

Figure 2.4: Illustration of the coin and shifting operation on a graph. (a) The coin operation mixes the coin states at each vertex; (b) The shifting operator shifts coin states along edges.

Hence, we write the step operator for the coined quantum walk as:

$$U = S \cdot C \tag{2.21}$$

i.e. the coin operator is applied first, followed by the shifting operator. If we start in the state $|\psi_0\rangle$, then we obtain the state of the system at time $t \in \mathbb{Z}_+$ by applying the step operator t times, i.e.

$$|\psi(t)\rangle = U^t|\psi_0\rangle. \tag{2.22}$$

2.3 Problem description and notions of efficiency

Having defined both the quantum circuit and quantum walk model, we can now state clearly the problem of interest. As mentioned in chapter 1, we want to reformulate a given quantum walk in terms of a quantum circuit in order to circumvent the lack of scalability of the quantum walk model as a physical model for implementation [35]. More precisely, the dynamics of a quantum walk (of any type) is determined by its unitary evolution operator—since the quantum circuit model is universal (that is, can be used to construct any unitary operator), we can (in principle) find a corresponding quantum circuit that reproduces the unitary evolution operator of the quantum walk of interest. In the case of the CTQW model and the Szegedy walk model, we want to efficiently implement the time-evolution operator $U(t)$ and the step operator U_{walk} in equations (2.12) and (2.18) respectively.

The notion of efficiency, however, varies depending on the type of quantum walk and whether one is interested in exact simulations or approximate simulations. A quantum circuit which maps to a unitary operator $\tilde{U}$ is said to simulate a given unitary evolution operator U approximately within error ϵ if the condition:

$$\left\| U - \tilde{U} \right\| < \epsilon, \tag{2.23}$$

is satisfied, where $\|..\|$ denotes the spectral norm of a matrix. An exact simulation is where $\tilde{U} = U$, i.e. $\epsilon = 0$.

In the CTQW model (i.e. $U = U(t)$), an approximate simulation of the time-evolution operator $U = \exp(-itH)$ within error ϵ is said to be efficient if the quantum circuit for $\tilde{U}$ satisfies equation (2.23) and uses at most $O(\text{poly}(\log(N), \|Ht\|, 1/\epsilon))$ one and two qubit gates [58], where poly(...) denotes a polynomial scaling in terms of the listed parameters. An efficient exact simulation of H is when $\tilde{U} = U$ and the quantum circuit for $\tilde{U}$ uses at most $O(\text{poly}(\log(N), \|Ht\|))$ one and two qubit gates.

In the Szegedy walk model (i.e. $U = U_{\text{walk}}$), an efficient approximate simulation

of the walk operator in equation (2.18) is when equation (2.23) is satisfied and the quantum circuit for $\tilde{U}$ uses at most $O(\text{poly}(\log(N), 1/\epsilon))$ one and two qubit gates [40]. An efficient exact simulation of the walk operator is when $\tilde{U} = U$ and the quantum circuit for $\tilde{U}$ uses at most $O(\text{poly}(\log(N)))$ one and two qubit gates[1].

In this thesis, we are interested in developing efficient exact simulations for the CTQW model and the Szegedy walk model. In the CTQW case, we also adopt a stronger definition of efficiency than discussed above: in addition to the constraints listed above, the number of quantum gates required to implement the time-evolution operator is independent of the time parameter t. In general, this is not always possible, as shown by the no-fast-forwarding theorem for sparse Hamiltonians [57]:

Theorem 2.1. (No-fast-forwarding theorem) *For any positive integer N there exists a row-computable sparse Hamiltonian with $\|H\| = 1$ such that simulating the evolution of H for time $t = \pi N/2$ within precision $1/4$ requires at least $N/4$ queries to H.*

As such, simulation of general sparse Hamiltonians in sublinear time is not possible. This result has been extended to non-sparse Hamiltonians as well in [64]. However, there exist classes of Hamiltonians that can be simulated in sublinear (or even constant) time, i.e. Hamiltonians for which their time evolution can be fast-forwarded, as pointed out in [64]—and it is precisely these cases that we are interested in. Hence, in chapter 3, we identify classes of graphs for which it is possible to simulate the time-evolution operator U_{walk} in constant time.

Putting everything together, this means that for both the CTQW model and the Szegedy walk model, we want a quantum circuit implementation for which $\tilde{U} = U$ is satisfied and the quantum circuit for $\tilde{U}$ uses at most $O(\text{poly}(\log(N)))$ one and two qubit gates (U here corresponds to the time-evolution operator and the step operator in equations (2.12) and (2.18) respectively).

[1]The same definition of efficiency is adopted for the coined quantum walk model.

Chapter 3

Continuous-time quantum walks

Based on X. Qiang, T. Loke, A. Montanaro, K. Aungskunsiri, X. Zhou, J.L. O'Brien, J.B. Wang, and J.C.F. Matthews's original publication in Nature Communications 7 (2016) p. 11511 [43], and T. Loke and J.B. Wang's original publication in Journal of Physics A: Mathematical and Theoretical 50 (2017) p. 055303 [44].

3.1 Introduction

The continuous-time quantum walk (CTQW) is currently a subject of intense theoretical and experimental investigation due to its established role in quantum computation and quantum simulation [30, 74, 83, 87, 88]. In fact, any dynamical simulation of a Hamiltonian system in quantum physics and quantum chemistry can be discretised and mapped onto a CTQW on specific graphs [11, 22, 89]. The primary difficulty of such a numerical simulation lies in the exponential scaling of the Hilbert space as a function of the system size, making real-world size problems intractable on classical computers. Some examples of quantum algorithms based on the CTQW are searching for a marked element on a graph [36, 90, 91] and testing graph isomorphism [55, 56].

However, in order to run quantum algorithms based on the CTQW on a quantum computer, we require an efficient quantum circuit that implements the required CTQW. As discussed in chapter 1, construction of efficient quantum circuits for the CTQW has typically been considered under two separate categories: sparse graphs and dense graphs. For sparse graphs, there has been much literature that focuses on lowering the complexity of sparse graph implementation [57, 59, 61, 62], which as of [62], has been proven to be nearly optimal. In the case of dense graphs, there

are a select few classes of graphs that have been shown to have an efficient quantum circuit implementation, notably the complete graph, complete bipartite graph and star graph [65]. However, here we adopt a stronger definition of efficiency (namely, that the Hamiltonian can be fast-forwarded, and that the simulation is exact), as discussed in section 2.3, and identify classes of graphs (which are not necessarily sparse) whose CTQW time-evolution operator can be implemented in this fashion.

This chapter is organised as follows. Firstly, in section 3.2, we discuss diagonalisation of matrices, which maps CTQWs to quantum circuit implementations that have time-independent complexity. We then apply this tool to produce efficient quantum circuit implementations of circulant graphs in section 3.3. We discuss composite graphs in section 3.4, which yields new graphs that can be efficiently implemented using known implementations of the subgraphs. We then draw our conclusions and discuss possible future work in section 3.5.

3.2 Diagonalisation

Here, we are interested in implementing the CTQW time-evolution operator given the Hamiltonian H, as defined in equation (2.12), which we restate here:

$$U(t) = \exp(-itH). \tag{3.1}$$

For the purposes of this chapter, we adopt the definition of efficiency discussed in section 2.3, that is, a quantum circuit implementation for $U(t)$ is considered as efficient if it uses at most $O(\text{poly}(\log(N)))$ one and two qubit gates to simulate $U(t)$ exactly.

The main tool we use to implement CTQWs in an efficient fashion is diagonalisation. It is well-known that for a Hermitian matrix H, the spectral theorem guarantees that H can be diagonalised using its eigenbasis, that is $H = Q^\dagger \Lambda Q$ [92]. Here Q is a unitary matrix whose column vectors are eigenvectors of H, and Λ is a diagonal matrix of eigenvalues of H, which are all real and whose order is determined by the order of the eigenvectors in Q. From this, we can express the time-evolution operator of equation (2.12) as:

$$U(t) = Q^\dagger \exp(-it\Lambda) Q. \tag{3.2}$$

The diagonalisation approach confines the time-dependence of $U(t)$ to the diagonal

matrix $\exp(-it\Lambda)$, which can be readily implemented by a sequence of at most N controlled-phase gates with phase values being linear functions of t. Experimentally, this corresponds to a sequence of tunable controlled-phase gates, where the phase values are determined by t. In principle, a quantum circuit implementation for $U(t)$ that has time-independent complexity can always be obtained by using a general quantum compiler (as discussed in [42, 48]) to obtain a quantum circuit implementation for Q and $Q^\dagger$—however this is almost never efficient, since such methods typically scale exponentially in terms of complexity. In order to be able to implement $U(t)$ efficiently as a whole, we require that at most $O(\text{poly}(\log(N)))$ one- and two- qubit gates are used in implementing Q, $\exp(-it\Lambda)$ and $Q^\dagger$ individually (which is not always possible, as per the no-fast-forwarding theorem discussed in section 2.3). In the following sections, we will cover some classes of graphs that satisfy this criterion.

3.3 Circulant graphs

Circulant graphs are defined by symmetric circulant adjacency matrices for which each row j when right-rotated by one element, equals the next row $j+1$. Hence, the adjacency matrix for a circulant graph with N vertices is given by:

$$C = \begin{bmatrix} c_0 & c_1 & c_2 & \dots & c_{N-1} \\ c_{N-1} & c_0 & c_1 & \dots & c_{N-2} \\ c_{N-2} & c_{N-1} & c_0 & \dots & c_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & c_3 & \dots & c_0 \end{bmatrix}, \tag{3.3}$$

where $c_j = c_{N-j}$ for $j \in \{1, \dots, N-1\}$. Obviously, every circulant matrix can be generated given any row of the matrix—conventionally we use the first row of the matrix, denoted as $r_C = \{c_0, \dots, c_{N-1}\}$. Some examples of subclasses of circulant graphs are complete graphs, cycle graphs, and Mobius ladder graphs. It follows that Hamiltonians for CTQWs on any circulant graph have a symmetric circulant matrix representation, since $H = \gamma A$. An important property of circulant matrices is that they can be diagonalised by the quantum Fourier transform (i.e. the unitary discrete Fourier transform) [93], i.e. $H = Q^\dagger \Lambda Q$, where:

$$Q_{jk} = \frac{1}{\sqrt{N}} \omega^{jk}, \ \omega = \exp(2\pi i/N), \tag{3.4}$$

and the eigenvalue matrix Λ can be computed as $\Lambda = \text{diag}\left(\sqrt{N}Qr_C\right)$.

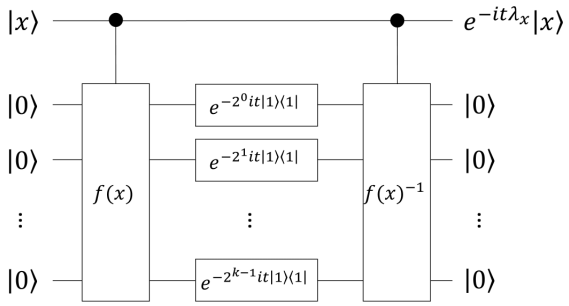
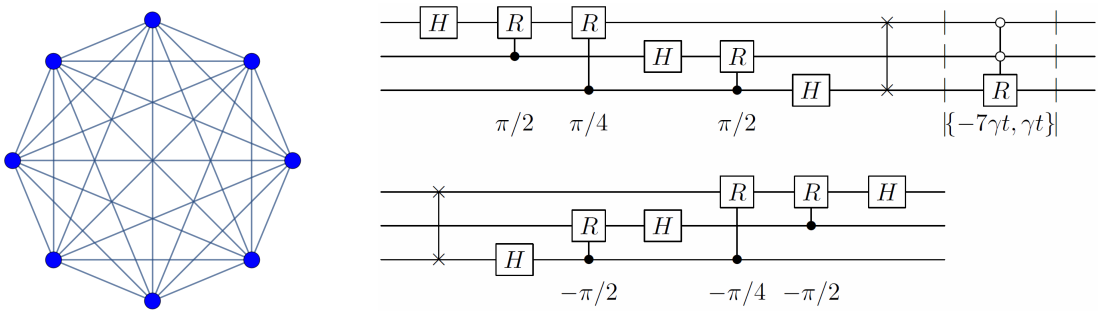


Figure 3.1: The quantum circuit for implementing the time-dependent operator $\exp(-it\Lambda)$ for a given Hamiltonian [38]. Here, $f(x)$ is a function that can be computed efficiently classically, such that $f(x)$ gives the x-th eigenvalue in Λ, i.e. $f(x) = \lambda_x = \Lambda_{x,x}$. However, this implementation of $\exp(-it\Lambda)$ will not provide an exact simulation, except in special cases.

The Fourier transformation Q can be implemented efficiently by the well-known QFT (quantum Fourier transform) quantum circuit [70]. For a circulant graph that has $N = 2^n$ vertices, the required QFT of N dimensions can be implemented with $O((\log(N))^2) = O(n^2)$ quantum gates acting on $O(n)$ qubits. To implement the inverse QFT, the same circuit is used in reverse order with phase gates of opposite sign. The diagonal term $\exp(-it\Lambda)$ can be implemented efficiently if Λ has at most $O(\text{poly}(n))$ non-zero eigenvalues, or more generally, if Λ can be characterised efficiently such that all $O(2^n)$ eigenvalues can be implemented efficiently. A general construction of efficient quantum circuits for $\exp(-it\Lambda)$ was given by Childs [38], which we reproduce in Figure 3.1.

There are many different classes of circulant graphs that can be implemented efficiently in this way. In the case of the cycle graph C_N, there are essentially $N/2$ distinct eigenvalues, given by the function $f(x) = 2\cos(2\pi x/N)$. In the case of the complete graph K_N, the eigenvalue function is $f(x) = N\delta_{x,1} - 1$. Figure 3.2 shows the K_8 graph together with its explicit quantum circuit implementation, which can easily be extended to any dimension $N = 2^n$.

However, the choice of using the quantum Fourier transform matrix as the eigenbasis of H is not strictly necessary—any equivalent eigenbasis can be chosen. An alternative quantum circuit implementation of the K_8 graph using a different eigenbasis is shown in Figure 3.3, which is much simpler. By collaborating with researchers from the Centre for Quantum Photonics in the University of Bristol, we have experimentally realised the quantum circuit implementation of the K_4 graph (with self-loops) on a two-qubit photonics quantum processor, as shown in Figure



(a) (b)

Figure 3.2: (a) The K_8 graph; (b) its corresponding quantum circuit implementation. The dashed vertical lines separates the matrices Q, $\exp(-it\Lambda)$ and $Q^\dagger$.

3.4.

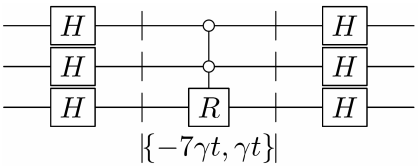


Figure 3.3: Alternative quantum circuit implementation of CTQW on the K_8 graph.

One application of CTQWs is to search a physical database for a marked entry, where the database is structured according to some graph [36, 94]. As such, it is worthwhile to consider if we can develop a quantum circuit implementation for the CTQW-based search algorithm. In this case, the marked Hamiltonian is defined as $H' = H + H_m$, where $H_m = |v_m\rangle\langle v_m|$ represents the marking at vertex v_m. As long as H is Hermitian, H' is also Hermitian (even in the case of multiple marked vertices), and so the diagonalisation procedure can be employed to implement the corresponding time evolution operator $U'(t) = \exp(-itH')$, although the required diagonalising matrix would vary depending on what v_m actually is. For example, for the complete graph K_8 with vertex 0 marked, the modified Hamiltonian is $H' = \gamma A + |0\rangle\langle 0|$, which is no longer a circulant matrix. As such, we cannot use the quantum Fourier transform matrix given by equation (3.4) or the Hadamard basis shown in Figure 3.3 to diagonalise H'. Nonetheless, diagonalisation is still possible and the corresponding eigenvalue matrix of H' is found to be $\Lambda = \text{diag}\left(\frac{1}{2}\left\{1 + 8\gamma + f(\gamma), 1 + 8\gamma - f(\gamma), 0, 0, 0, 0, 0, 0\right\}\right)$, where $f(\gamma) = \sqrt{1 + 4\gamma(16\gamma - 3)}$. Thus, if we define the amplitude rotation matrix ρ as:

$$\rho(\mu) = \begin{pmatrix} \mu & \sqrt{1-\mu^2} \\ \sqrt{1-\mu^2} & -\mu \end{pmatrix}, \tag{3.5}$$

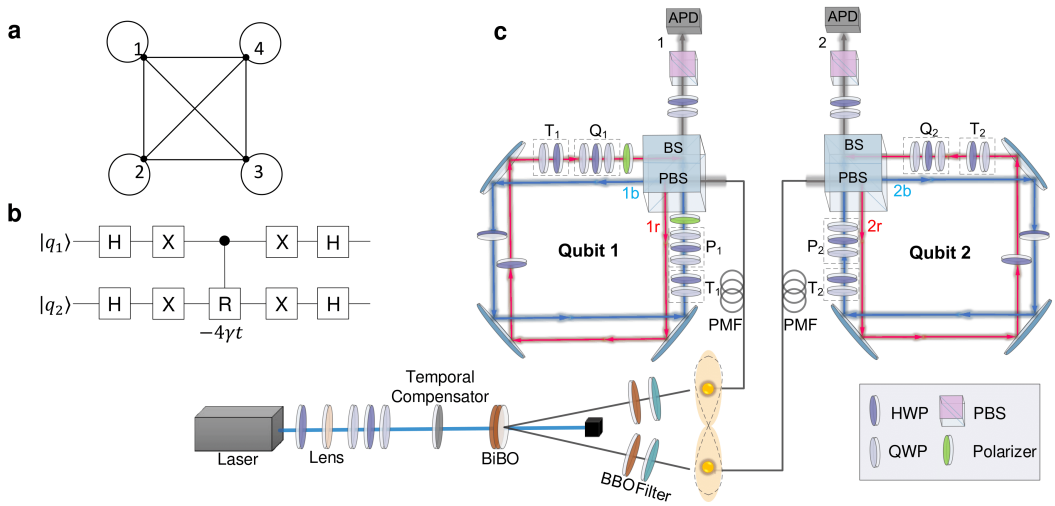


Figure 3.4: The schematic diagram and setup of experimental demonstration, sourced directly from [43]. (a) The K_4 graph with self-loops. (b) The quantum circuit for implementing CTQW on the K_4 graph with self-loops. This can also be used to implement CTQW on the K_4 graph without self-loops, up to a global phase factor $\exp(i\gamma t)$. The quantum gates H and X represent the Hadamard and Pauli-X gate respectively. $\mathrm{R} = [1, 0; 0, \exp(-i4\gamma t)]$ is a phase gate. (c) The experimental setup for a reconfigurable two-qubit photonics quantum processor, consisting of a polarisation-entangled photon source using paired type-I BiBO crystal in the sandwich configuration and displaced Sagnac interferometers.

then we find a quantum circuit implementation shown in Figure 3.5. This implementation was constructed by using the amplitude rotation matrices to selectively zero rows and columns of H'. The same construction can be extended to any K_N graph with a marked vertex. However, this construction for Q and $Q^\dagger$ scales as $O(N)$ in terms of the number of gates required, and as such, this circuit is not efficient in general.

One can also construct a particular circulant graph and use the same methodology to implement the corresponding CTQW. For example, Figure 3.6(a) shows a circulant graph with 16 vertices (which we shall label as CG_{16}). We generate the adjacency matrix A using $r_C = \{0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1\}$. The corresponding eigenvalue matrix is then $\Lambda = \mathrm{diag}\left(\gamma\left\{9, -1, 1+2\sqrt{2}, -1, -3, -1, 1-2\sqrt{2}, -1,\right.\right.$ $\left.\left.1, -1, 1-2\sqrt{2}, -1, -3, -1, 1+2\sqrt{2}, -1\right\}\right)$. The corresponding quantum circuit to perform the CTQW on the CG_{16} graph is shown in Figure 3.6(b).

Thus, the quantum circuit implementations of CTQWs on circulant graphs can be constructed which satisfies the notion of efficiency outlined in section 2.3. Compared with the best known classical algorithm based on the fast Fourier transform, which has a complexity that scales as $O(n2^n)$ [95], the proposed quantum circuit implementation generates the evolution state $|\psi(t)\rangle$ with an exponential advantage in speed.

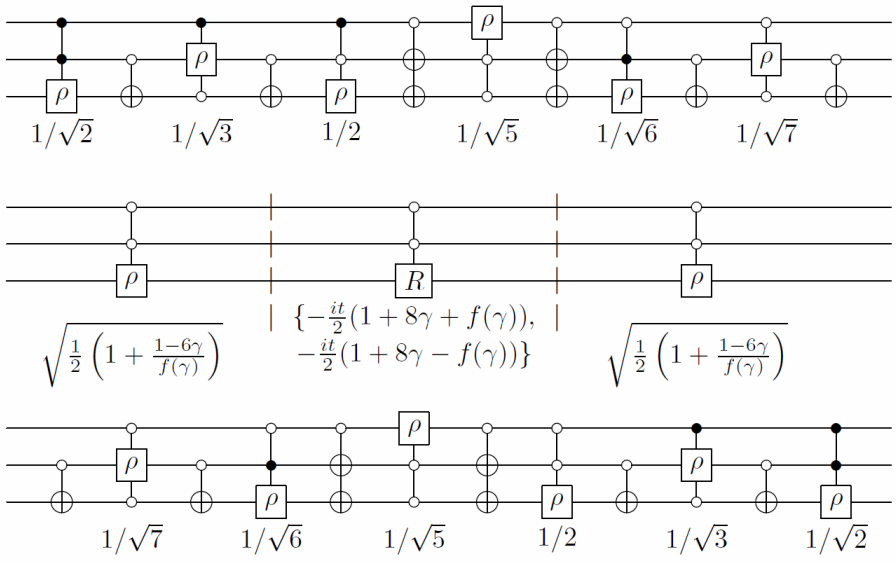
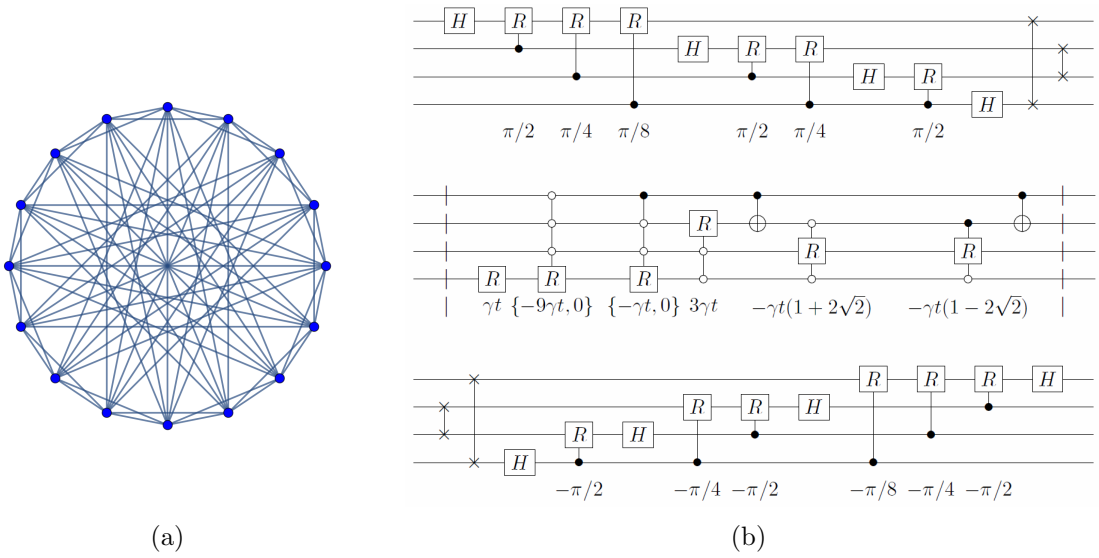


Figure 3.5: Quantum circuit implementation of CTQW on the K_8 graph (shown in Figure 3.2(a)) with vertex 0 marked.



(a) (b)

Figure 3.6: (a) The CG_{16} graph; (b) its corresponding quantum circuit implementation.

For the problem of sampling the output probability distribution given by $|\psi(t)\rangle$ for a CTQW on a circulant graph, it can be shown that solving the same problem on a classical computer is also intractable, assuming certain conjectures from computation complexity theory. The argument can be sketched as follows: Consider a circuit of the form $Q^\dagger DQ$, where D is a diagonal matrix made up of poly(n) controlled-phase gates and Q is the quantum Fourier transform. Let p_D be the probability of measuring all qubits to be 0 in the computational basis after $Q^\dagger DQ$ is applied to the initial state $|0\rangle^{\otimes n}$, i.e. $p_D = |\langle 0|^{\otimes n} Q^\dagger DQ |0\rangle^{\otimes n}|^2$. Then it can be shown that p_D can be obtained through a circuit of the form $H^{\otimes n} D H^{\otimes n}$, which belongs to a class of circuits known as instantaneous quantum polynomial time (IQP) [96, 97], implying that p_D is classically hard to compute[1]. This implies that we can solve a problem that is classically hard using the above quantum circuits for circulant graphs, thus demonstrating quantum supremacy over classical computers.

3.4 Composite graphs

In this section, we will examine two classes of composite graphs (that is, graphs that are constructed from other graphs) that can be efficiently implemented. In particular, we will look at commuting graphs and the Cartesian product of graphs in sections 3.4.1 and 3.4.2 respectively.

3.4.1 Commuting graphs

Suppose we have two matrices H_1 and H_2. In general, when H_1 and H_2 do not commute, their sum can be simulated by the Lie product formula [38]:

$$\exp(-it(H_1 + H_2)) = \lim_{m\to\infty} \left(\exp(-itH_1/m)\exp(-itH_2/m)\right)^m, \tag{3.6}$$

which, in practice, requires higher-order approximations to reduce the bounded error (which also scales with t). However, in the case where H_1 and H_2 commute, we can write the expression exactly as:

$$\exp(-it(H_1 + H_2)) = \exp(-itH_1)\exp(-itH_2). \tag{3.7}$$

[1] For the full details of the argument and experimental setup, we refer the reader to [43], which is reproduced in Appendix A.

Taking $H_1 = \gamma A$ and $H_2 = \gamma B$, where γ is constant, A and B are the adjacency matrices of two commuting graphs, i.e. $[A, B] = 0$. It follows that if the individual time-evolution operators $\exp(-it\gamma A)$ and $\exp(-it\gamma B)$ can be efficiently implemented, then the time-evolution operator for the graph $A + B$, that is, $\exp(-it\gamma(A + B))$, can also be efficiently implemented, provided $[A, B] = 0$.

The general criteria for commuting graphs are studied in [98]. One particular class of graphs is the interdependent networks, defined by:

$$A = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \text{ and } B = \begin{pmatrix} 0 & B_0 \\ B_0^T & 0 \end{pmatrix},$$

where the interlink graph B connects two subgraphs A_1 and A_2, which are both symmetric, i.e. $A_1 = A_1^T$ and $A_2 = A_2^T$. In this instance, the condition for commutativity becomes:

$$A_1 B_0 = B_0 A_2. \tag{3.8}$$

Suppose Q_1 and Q_2 diagonalise A_1 and A_2 respectively. Then, we have:

$$\Lambda_1 = Q_1^\dagger A_1 Q_1 \text{ and } \Lambda_2 = Q_2^\dagger A_2 Q_2.$$

Then the following matrix:

$$Q = \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix},$$

diagonalises A, and gives the eigenvalue matrix:

$$\Lambda = \begin{pmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{pmatrix}.$$

Suppose B_0 is diagonalised by Q_0, i.e. $\zeta_0 = Q_0 B_0 Q_0^\dagger$. Then, it can be readily shown that if $B_0 = B_0^T$ (namely a symmetric interconnection), the diagonalising matrix for B is:

$$Q' = \frac{1}{\sqrt{2}} \begin{pmatrix} Q_0 & Q_0 \\ Q_0 & -Q_0 \end{pmatrix} = H \otimes Q_0,$$

where H is the Hadamard matrix. The corresponding eigenvalue matrix is:

$$\zeta = \begin{pmatrix} \zeta_0 & 0 \\ 0 & -\zeta_0 \end{pmatrix} = \sigma_z \otimes \zeta_0,$$

where σ_z is the Pauli-z matrix. Hence, in the case where $[A, B] = 0$, we expand the CTQW time-evolution operator as:

$$\exp(-it(H_1 + H_2)) = Q^\dagger \exp(-it\Lambda) Q \; Q'^\dagger \exp(-it\zeta) Q'. \tag{3.9}$$

Next, we examine some explicit examples of interdependent networks in which $[A, B] = 0$ is satisfied and equation (3.8) holds. One special case is that of identity interconnections between two disjoint copies of a graph with N vertices, namely $A_1 = A_2$ and $B_0 = I_N$. The diagonalising matrices for A and B are, respectively:

$$Q = \begin{pmatrix} Q_1 & 0 \\ 0 & Q_1 \end{pmatrix} = I_2 \otimes Q_1 \quad \text{and} \quad Q' = H \otimes I_n,$$

giving the eigenvalue matrices:

$$\Lambda = I_2 \otimes \Lambda_1 \quad \text{and} \quad \zeta = \sigma_z \otimes I_n.$$

Hence, if we are able to implement A_1 efficiently, it follows that the interdependent network with $A_1 = A_2$ and $B_0 = I_N$ can be implemented efficiently. An equivalent result can be achieved by noting that $A+B = \sigma_x \oplus A_1$, where σ_x is the Pauli-x matrix and $\oplus$ denotes the Cartesian product (defined in section 3.4.2), and then applying the methods of section 3.4.2. Figure 3.7 shows one class of graphs (complete graphs with identity interconnections) that can be constructed as above, together with its corresponding circuit implementation.

Another case where $[A, B] = 0$ is using complete interconnections between two disjoint degree-regular graphs (with N_1 and N_2 vertices respectively) of the same degree. That is, $\deg(A_1) = \deg(A_2) = d$ and $B_0 = J_{N_1,N_2}$, where J_{N_1,N_2} is the N_1-by-N_2 matrix with all 1's. Although in general $B_0 \neq B_0^T$, the interconnection matrix:

$$B = \begin{pmatrix} 0 & J_{N_1,N_2} \\ J_{N_2,N_1} & 0 \end{pmatrix}, \tag{3.10}$$

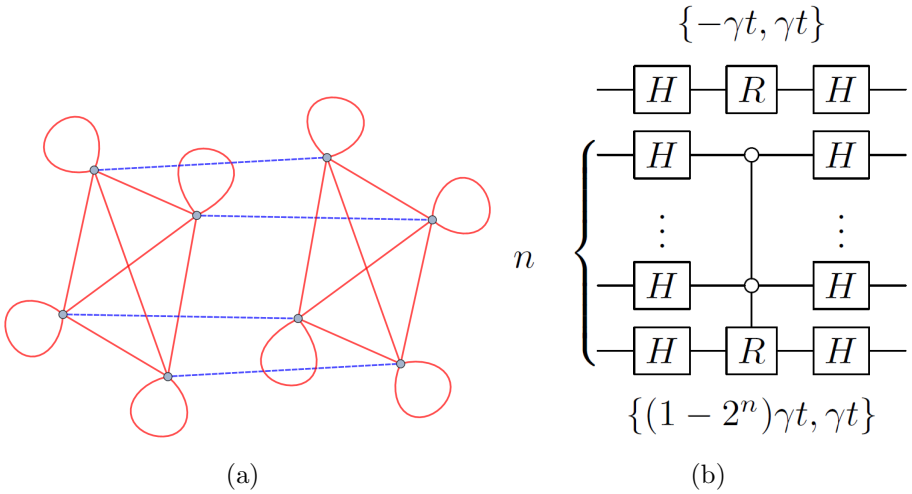


Figure 3.7: (a) Two disjoint K_N graphs (where $N = 2^n$) with identity interconnections (solid red lines and dashed blue lines indicate edges belonging to A and B respectively), where $n = 2$; (b) the corresponding quantum circuit implementation for the CTQW time-evolution operator on this graph.

can still be diagonalised easily in the case where $N_1 = 2^{n_1}$ and $N_2 = 2^{n_2}$, for $n_1, n_2 \in \mathbb{Z}_+$. For convenience, we assume $N_1 \geq N_2$, and note that B is the complete bipartite graph K_{N_1,N_2}. Keeping in mind that within the CTQW framework, we can extend any given Hamiltonian H by some number of dimensions by defining $H' = \mathrm{diag}\left(\{H, 0, \ldots, 0\}\right)$, the diagonalisation operator for B can be written mathematically as:

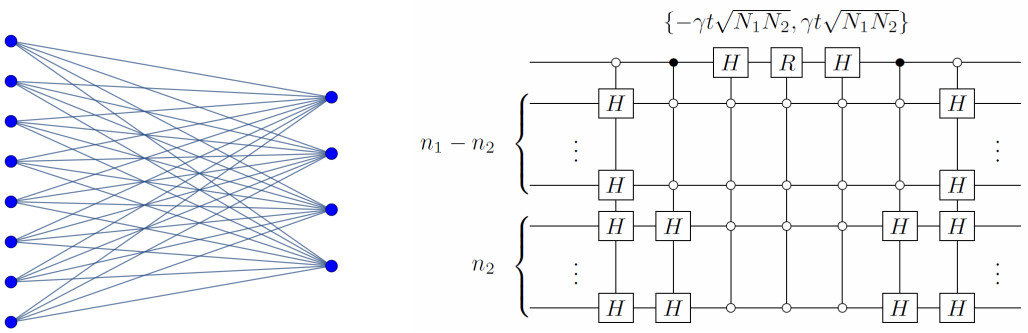
$$Q' = \left(I_{2^{n_1+1}} + (H - I_2) \otimes P_0^{\otimes n_1}\right)\left(I_{2^{n_1+1}} + P_0 \otimes \left(H^{\otimes n_1} - I_{2^{n_1}}\right) + P_1 \otimes P_0^{\otimes (n_1-n_2)} \otimes \left(H^{\otimes n_2} - I_{2^{n_2}}\right)\right), \tag{3.11}$$

where $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$ are the 2-dimensional projection operators. The corresponding eigenvalue matrix of B is then given by:

$$\zeta = \mathrm{diag}\left(\left\{(+\sqrt{N_1 N_2})^1, 0^{N_1-1}, (-\sqrt{N_1 N_2})^1, 0^{N_2-1}\right\}\right). \tag{3.12}$$

where $(\ldots)^x$ denotes a repeated eigenvalue. Figure 3.8 shows the complete bipartite graph K_{N_1,N_2} together with its corresponding quantum circuit implementation. As a corollary, the star graph S_{2^n+1} can also be implemented using the same method, since S_{2^n+1} is equivalent to $K_{2^n,1}$. Hence, if we have degree-regular graphs A_1 and A_2 satisfying:

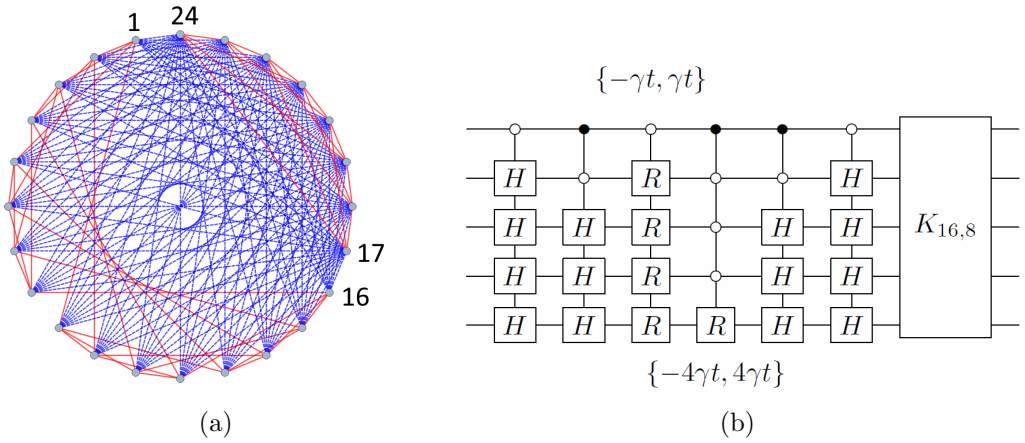
$$\deg\left(A_1\right)_v = \deg\left(A_2\right)_v = d \;\forall v \in V, \tag{3.13}$$



(a) (b)

Figure 3.8: (a) An example of the complete bipartite graph K_{N_1,N_2}, where $N_1 = 8$ and $N_2 = 4$; (b) quantum circuit implementation of the CTQW time-evolution operator for the complete bipartite graph K_{N_1,N_2}, where $N_1 = 2^{n_1}$ and $N_2 = 2^{n_2}$.

where $N_1 = 2^{n_1}$ and $N_2 = 2^{n_2}$, which can both be efficiently implemented, then it follows that the interdependent network built from A_1, A_2 and $B_0 = J_{N_1,N_2}$ can be efficiently implemented. Figure 3.9(a) gives an example of this kind of graph, where vertices 1–16 belong to the Q_4 graph (hypercube graph of dimension 4—refer to section 3.4.2), and 17–24 belong to the $K_{4,4}$ graph. The quantum circuit implementation of the composite graph shown in Figure 3.9(a) is given by Figure 3.9(b), where the $K_{16,8}$ circuit is already described above and given by Figure 3.8(b). Note that in general, the K_{N_1,N_2} graph and by extension, the resulting interdependent network with complete interconnections is not degree-regular - so this provides an example of a class of graphs that is not degree-regular but still has an efficient quantum circuit implementation for the CTQW time-evolution operator.



(a) (b)

Figure 3.9: (a) Disjoint Q_4 and $K_{4,4}$ with complete interconnections (solid red lines and dashed blue lines indicate edges belonging to A and B respectively); (b) its corresponding quantum circuit implementation.

3.4.2 Cartesian product of graphs

The Cartesian product of two graphs is a graph which has an adjacency matrix that is formed by taking the Cartesian product of two other adjacency matrices. Given two matrices H_1 and H_2 of dimension N_1-by-N_1 and N_2-by-N_2 respectively, the Cartesian product of H_1 and H_2 is given by:

$$H_1 \oplus H_2 = H_1 \otimes I_{N_2} + I_{N_1} \otimes H_2, \tag{3.14}$$

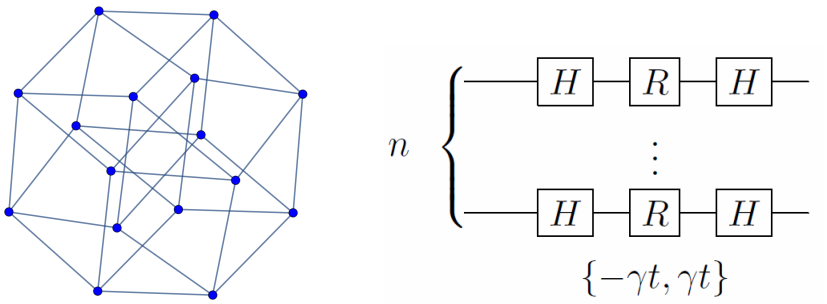
which is a matrix of dimension N_1N_2-by-N_1N_2. In particular, if we define $H = H_1 \oplus H_2$, we have:

$$\exp(-itH) = \exp(-itH_1) \otimes \exp(-itH_2), \tag{3.15}$$

or, more compactly, $U(t) = U_1(t) \otimes U_2(t)$. Again we set $H_1 = \gamma A$ and $H_2 = \gamma B$, i.e. the Hamiltonians H_1 and H_2 correspond to graphs A and B respectively. This implies that if we have an efficient quantum circuit implementation for the individual CTQW time-evolution operators on the graphs A and B, the implementation for $U(t)$ (which is the time-evolution operator that corresponds to the CTQW on the graph $A \oplus B$) is easily formed by stacking the individual quantum circuit implementations in parallel. As a corollary, in the case where $A = B$, then $H = A \oplus A$ corresponds to the Hamiltonian for the non-interacting two-particle quantum walk on A.

One particular class of graphs that is constructed using the Cartesian product of graphs is the hypercube Q_n. Given the path graph of length 2 with adjacency matrix $K_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, Q_n is constructed as $Q_n = K_2^{\oplus n}$, i.e. it is the Cartesian product of n copies of K_2 [99]. As such, Q_n is a graph with 2^n vertices and degree-regular with degree n. K_2 is diagonalisable using the Hadamard matrix H, giving its eigenvalue matrix $\Lambda_{K_2} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. Figure 3.10 shows the hypercube graph Q_n with its corresponding quantum circuit implementation.

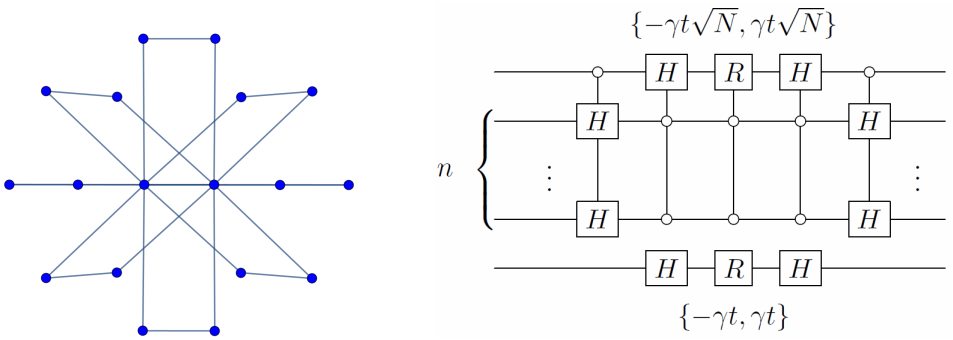
Another example of this class of graphs is the book graph B_N [100], which is constructed as $B_N = S_{N+1} \oplus K_2$, where S_{N+1} is the star graph on $N + 1$ vertices. As we have discussed in section 3.4.1, S_{N+1} can be implemented efficiently in a quantum circuit if $N = 2^n$ for some $n \in \mathbb{Z}_+$—hence book graphs of the form B_{2^n}



(a) (b)

Figure 3.10: (a) The hypercube graph Q_n, where $n = 4$; (b) its corresponding quantum circuit implementation for the CTQW time-evolution operator on Q_n.

can be efficiently implemented as well, as shown in Figure 3.11.



(a) (b)

Figure 3.11: (a) The book graph B_N, where $N = 8$; (b) corresponding quantum circuit implementation for the CTQW time-evolution operator on the book graph for $N = 2^n$.

3.5 Conclusions and future work

To summarise, we have identified several new classes of graphs for which the CTQW time-evolution operator can be implemented efficiently, in a time-independent and exact fashion. Using the tool of diagonalisation, we have described how CTQWs on circulant graphs can be efficiently implemented on a quantum computer, if the eigenvalues of the graphs can be characterised efficiently classically. We have also shown that efficient quantum circuit implementations for composite graphs (in particular, commuting graphs and Cartesian products of graphs) can be constructed provided the subgraphs can be efficiently implemented, further extending the classes of graphs that can be efficiently simulated.

We note that these results also apply to the more general setting of Hamiltonian simulation, that is, the results above apply even when the Hamiltonian does not cor-

respond to an adjacency matrix. Hence, one area for further research is to apply the above results to the more general setting by identifying classes of Hamiltonians that are of some physical significance. For example, chemical molecules with circulant symmetry can potentially be treated using the above methods.

Chapter 4

Szegedy quantum walks

Based on T. Loke and J.B. Wang's paper on "Efficient quantum circuits for Szegedy quantum walks", which has been accepted for publication in Annals of Physics.

4.1 Introduction

A limitation of the main formalisms for quantum walks (the coined quantum walk model and the CTQW model) is the requirement that the adjacency matrix of the graph be symmetric, i.e. edges connecting vertices must do so in both directions, in order to preserve unitarity of the walk. However, real-world systems such as networks [9, 10, 101] and energy transport systems [102] are usually modelled using random walks on directed or weighted graphs. In order to study these systems using a quantum walk, a formalism that supports directed graphs (or weighted graphs in general) is necessary.

In 2004, M. Szegedy introduced a different formalism for the discrete-time quantum walk (termed as the Szegedy quantum walk) by means of quantising Markov chains [33, 34], as outlined in section 2.2.3. Since it allows for the quantisation of any Markov chain, it allows one to define a unitary quantum walk on directed (or more generally, weighted) graphs [10], overcoming the limitations of the coined quantum walk model and the CTQW model. It has also been shown that the Szegedy quantum walk provides a quadratic speedup in the quantum hitting time (for any reversible Markov chain with one marked element), compared to the classical random walk [103]. The Szegedy quantum walk has also proven to be useful for a variety of different applications, for example, verifying matrix products [104], testing group commutativity [105], searching for a marked element in a graph [91],

approximating the effective resistance in electrical networks [106], and quite notably, it is instrumental in the quantum Pagerank algorithm [9, 10] for determining the relative importance of nodes in a graph.

As with any quantum walk formalism, the realisation of quantum algorithms based on Szegedy walks requires an efficient quantum circuit implementation for the walk itself. Previous work by Chiang *et al.* [40] has showed that given a transition matrix that corresponds to an arbitrary sparse classical random walk, a quantum circuit for the walk operator can be realised. These quantum circuits scale linearly with the sparsity parameter d and polynomially in $\log(1/\epsilon)$ (where ϵ is the approximation accuracy to the entries of the transition matrix). In this chapter, we consider some families of transition matrices that possess certain symmetries where an exact efficient quantum circuit for the Szegedy evolution operator can also be realised, even if they are not sparse.

This chapter is organised as follows. In section 4.2, we establish a theoretical framework for the construction of efficient quantum circuit implementations of Szegedy quantum walks, given a Markov chain that has a transition matrix satisfying some constraints. We then apply this theory to the class of Markov chains where the transition matrix is described by a cyclic permutation in section 4.3. We also show that the class of complete bipartite graphs can be efficiently simulated in section 4.4. Then, we extend the results of section 4.2 to a tensor product of Markov chains in section 4.5. We briefly discuss the implementation of weighted interdependent networks in section 4.6. Next, in section 4.7, we apply the results of section 4.3 to construct efficient quantum circuits simulating Szegedy walks involved in the quantum Pagerank algorithm for some classes of directed graphs. Lastly, we draw our conclusions and discuss possible future work in section 4.8.

4.2 Circuit implementation

Here, we are interested in implementing the Szegedy walk operator for a given Markov chain, as defined in equation (2.18), which we restate here:

$$U_{\text{walk}} = \mathcal{S}(I - 2\Pi) = \mathcal{S}\mathcal{R}. \tag{4.1}$$

For the purposes of this chapter, we adopt the definition of efficiency discussed in section 2.3, that is, a quantum circuit implementation for U_{walk} is considered as

efficient if it uses at most $O(\text{poly}(\log(N)))$ one and two qubit gates to simulate U_{walk} exactly.

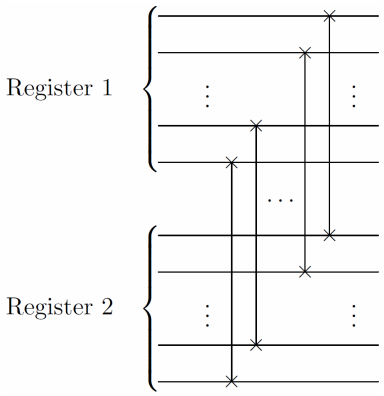


Figure 4.1: Quantum circuit implementing the swap operator $\mathcal{S}$.

We first note that implementing the swap operator $\mathcal{S}$ in a quantum circuit is straightforward and efficient—it simply consists of $\lceil \log_2(N) \rceil$ swap gates applied between the two registers, as shown in Figure 4.1. In order to implement the reflection operator $\mathcal{R}$, we diagonalise it using a unitary operation U:

$$U\mathcal{R}U^\dagger = U(I - 2\Pi)U^\dagger = I - 2U\Pi U^\dagger. \tag{4.2}$$

Let U be of the block-diagonal form $U = \sum_{i=0}^{N-1} |i\rangle\langle i| \otimes U_i$. Then expanding the above expression using equations (2.15) and (2.16) gives:

$$U\mathcal{R}U^\dagger = I - 2\sum_{i=0}^{N-1} |i\rangle\langle i| \otimes \left(U_i|\phi_i\rangle\langle\phi_i|U_i^\dagger\right). \tag{4.3}$$

Choose U_i such that $U_i|\phi_i\rangle = |b\rangle \ \forall i$ for some computational basis state $|b\rangle$. In other words, U_i acts to transform $|\phi_i\rangle$ into a fixed computational basis state $|b\rangle$, or conversely, the inverse operator $U_i^\dagger$ generates the state $|\phi_i\rangle$ from $|b\rangle$. Then:

$$U\mathcal{R}U^\dagger = I - 2I_N \otimes |b\rangle\langle b| = I_N \otimes (I - 2|b\rangle\langle b|) \equiv D, \tag{4.4}$$

which is a diagonal matrix that can be readily implemented using a controlled-π gate (with multiple conditionals) applied to the second register. Hence, the Szegedy walk operator becomes:

$$U_{\text{walk}} = \mathcal{S}U^\dagger DU. \tag{4.5}$$

It is, of course, infeasible to realise such an operator U efficiently in general—however certain symmetries in the transition matrix P do allow for U to be implemented

efficiently. Now, let us write $U_i = K_b^\dagger T_i$, such that $K_b^\dagger|\phi_r\rangle = |b\rangle$ and $T_i|\phi_i\rangle = |\phi_r\rangle$, where K_b and T_i are both unitary operations, and $|\phi_r\rangle$ is a chosen reference state. In other words, this means that K_b prepares the state $|\phi_r\rangle$ from a computational basis state $|b\rangle$, and T_i transforms $|\phi_i\rangle$ into $|\phi_r\rangle$. This corresponds to transforming the ith column of P into the rth column of P (with square roots on both). Note that this scheme works for any choice of the reference state $|\phi_r\rangle$ or the computational basis state $|b\rangle$, i.e. they can be chosen arbitrarily.

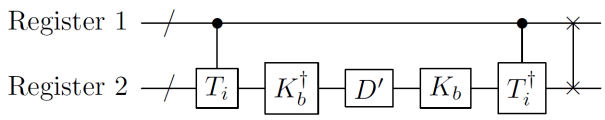


Figure 4.2: Quantum circuit implementing U_{walk} according to equation (4.5), with $D' = I_N - 2|b\rangle\langle b|$. The symbol / denotes a shorthand for registers of qubits representing N states each.

Figure 4.2 shows the general quantum circuit for implementing U_{walk}. The criteria for an efficient circuit implementation of U_{walk} is thus:

Theorem 4.1. *The Szegedy walk operator U_{walk} can be efficiently implemented if the following conditions are satisfied:*

1. *Able to implement all N transformations in the form $\mathcal{T} = \left(\sum_{i=0}^{N-1} |i\rangle\langle i| \otimes T_i\right)$ (and its inverse) where $T_i : |\phi_i\rangle \to |\phi_r\rangle$ using at most $O(\text{poly}(\log(N)))$ gates; and*

2. *Able to identify and implement a preparation routine $K_b : |b\rangle \to |\phi_r\rangle$ (and its inverse) for a chosen computational basis state $|b\rangle$ efficiently.*

Proof. From equation (4.5), U_{walk} is efficiently implementable if its components $\mathcal{S}$, $U^\dagger$, D and U are efficiently implementable. As discussed previously, $\mathcal{S}$ and D are efficiently implementable. Since $U = \sum_{i=0}^{N-1} |i\rangle\langle i| \otimes U_i$ and $U_i = K_b^\dagger T_i$, we can write $U = (I_N \otimes K_b^\dagger)\left(\sum_{i=0}^{N-1} |i\rangle\langle i| \otimes T_i\right)$. Hence U and $U^\dagger$ (and thus U_{walk}) are efficiently implementable if the conditions above are satisfied. □

For condition 2 of theorem 4.1 in particular, we can make use the state preparation method using integrals [12, 66, 107] in order to implement K_b and its inverse. This gives the following result:

Corollary 4.1.1. *Suppose $\mathcal{T}$ (and its inverse) can be efficiently implemented, and that the probability distribution $\{P_{j,r}\}$ can be generated by integrating a certain probability density function $p(z)$, i.e. $P_{j,r} = \int_{z_j}^{z_{j+1}} p(z)dz$ where $z_{j+1} - z_j = \Delta z$. If $p(z)$ is (classically) efficiently integrable, then the Szegedy walk operator U_{walk} can be efficiently implemented.*

Proof. This follows directly from [66], which provides a method to prepare $|\phi_r\rangle = \sum_{i=0}^{N-1} \sqrt{P_{j,r}}|j\rangle$ under these conditions. The inverse procedure $K_b^\dagger$ can be obtained by uncomputation of K_b, i.e. applying the circuit for K_b in reverse order. Hence, the conditions in Theorem 4.1 are met. □

However, we note that K_b doesn't have to be constructed using this method—any K_b that satisfies $K_b^\dagger|\phi_r\rangle = |b\rangle$ is sufficient. Some examples of this will be provided in the following sections.

This formalism can be generalised slightly by allowing for more than one reference state. Consider a partition $Z = \{Z_1, \ldots, Z_M\}$ of the set $X = \{0, \ldots, N-1\}$ such that $\emptyset \notin Z$, $\bigcup_{x=1}^{M} Z_x = X$ and $Z_x \cap Z_y = \emptyset \; \forall x \neq y$. generalising the previous equations gives:

$$U_{\text{walk}} = \mathcal{S} \prod_{x=1}^{M} U_x^\dagger D_x U_x \tag{4.6}$$

$$U_x = \sum_{y \in Z_x} |y\rangle\langle y| \otimes U_{x,y} + \left(\sum_{y' \notin Z_x} |y'\rangle\langle y'| \right) \otimes I_N \tag{4.7}$$

$$D_x = I - \left(2 \sum_{y \in Z_x} |y\rangle\langle y| \right) \otimes |b_x\rangle\langle b_x| \tag{4.8}$$

$$U_{x,y} = K_{b_x}^\dagger T_{x,y} \tag{4.9}$$

with $K_{b_x}^\dagger|\phi_{r_x}\rangle = |b_x\rangle$ and $T_{x,y}|\phi_y\rangle = |\phi_{r_x}\rangle$, where $|\phi_{r_x}\rangle$ is the reference state for the subset Z_x. The corresponding quantum circuit is similar to that given in Figure 4.2, except with M independent segments with conditionals determined by the partition Z. The criteria for efficient circuit implementation of U_{walk} with the partition $Z = \{Z_1, ..., Z_M\}$ is thus:

Theorem 4.2. *The Szegedy walk operator U_{walk} can be efficiently implemented with the partition $Z = \{Z_1, ..., Z_M\}$ if the following conditions are satisfied:*

1. *The number of subsets must be bounded above as $M \leq O(\text{poly}(\log(N)))$.*

2. *For every subset Z_x,*

   (a) *Able to implement controlled unitary operations according to Z_x efficiently, i.e. $\left(\sum_{y\in Z_x}|y\rangle\langle y|\right)\otimes U+\left(\sum_{y'\notin Z_x}|y'\rangle\langle y'|\right)\otimes I_N$ can be efficiently implemented assuming U can be done efficiently;*

   (b) *Able to implement all $|Z_x|$ transformations in the form $\mathcal{T}_x=\sum_{y\in Z_x}|y\rangle\langle y|\otimes T_{x,y}+\left(\sum_{y'\notin Z_x}|y'\rangle\langle y'|\right)\otimes I_N$ where $T_{x,y}:|\phi_y\rangle\rightarrow|\phi_{r_x}\rangle$ using at most $O(\text{poly}(\log(N)))$ gates; and*

   (c) *Able to identify and implement a preparation routine $K_{b_x}:|b_x\rangle\rightarrow|\phi_{r_x}\rangle$ (and its inverse) for a chosen computational basis state $|b_x\rangle$ efficiently.*

Proof. From equation (4.6), U_{walk} is efficiently implementable iff each of its components $\mathcal{S}$, $U_x^\dagger$, D_x and U_x are efficiently implementable and the number of subsets is bounded above as $M\leq O(\text{poly}(\log(N)))$. As discussed previously, $\mathcal{S}$ is efficiently implementable. If condition 2(a) is satisfied, then it follows that D_x can be efficiently implemented, since $U=I_N-2|b_x\rangle\langle b_x|$ can be implemented using a single controlled-π gate. From equations (4.7) and (4.9), we can write:

$$U_x=\left(\left(\sum_{y\in Z_x}|y\rangle\langle y|\right)\otimes K_{b_x}^\dagger+\left(\sum_{y'\notin Z_x}|y'\rangle\langle y'|\right)\otimes I_N\right)\left(\sum_{y\in Z_x}|y\rangle\langle y|\otimes T_{x,y}+\left(\sum_{y'\notin Z_x}|y'\rangle\langle y'|\right)\otimes I_N\right). \tag{4.10}$$

If conditions 2(a) and 2(c) are satisfied, then the first term can be implemented by identifying $U=K_{b_x}^\dagger$ in condition 2(a). Hence U_x and $U_x^\dagger$ (and thus U_{walk}) are efficiently implementable if the conditions above are satisfied. □

As before, we can use the state preparation method using integrals in order to implement K_{b_x} and its inverse, although any other construction of K_{b_x} that satisfies $K_{b_x}^\dagger|\phi_{r_x}\rangle=|b_x\rangle$ is sufficient.

4.3 Cyclic permutations

A cyclic permutation is where elements of a set are shifted by a fixed offset. Formally, let R be the one-element right-rotation operator such that $R\,[c_1,c_2,...,c_{N-1},c_N]^T=$

$[c_N, c_1, ..., c_{N-2}, c_{N-1}]^T$, and $L = R^\dagger$ is the one-element left-rotation operator. Both R and L can be implemented in a quantum circuit using $O(\log(N))$ generalised CNOT gates, as shown in Figure 4.3. In this section, we consider classes of transition matrix P where each column is a fixed cyclic permutation of the previous column, i.e. we can write $|\phi_{i+1}\rangle = R^x|\phi_i\rangle$ or $|\phi_{i+1}\rangle = L^x|\phi_i\rangle$ for some $x \in \mathbb{Z}_+$ for all values of i.

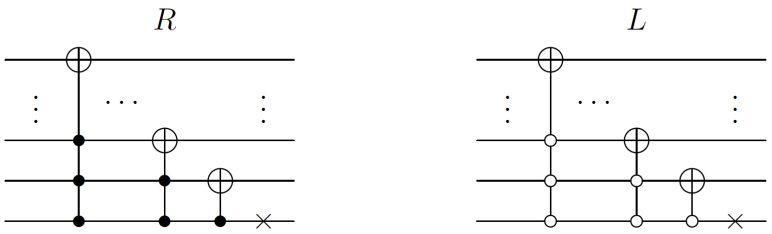


Figure 4.3: Quantum circuit implementing the R and L operators.

One important class of cyclic permutations is circulant matrices. In a circulant matrix, each column is right-rotated by one element with respect to the previous column, i.e. $|\phi_{i+1}\rangle = R|\phi_i\rangle$. Hence, we can construct the projector states as:

$$|\psi_i\rangle = |i\rangle \otimes R^i|\phi_0\rangle \tag{4.11}$$

Hence, if we select a single reference state as $|\phi_r\rangle = |\phi_0\rangle$, we can define the transformations as $T_i = (R^\dagger)^i = L^i$. This set of transformations can always be efficiently implemented, since the operations $\{L, L^2, L^4, L^8, ...\}$ can be realised by decrementing the corresponding bit values by 1—this takes $O(\log(N)^2)$ generalised CNOT gates. The inverse transformations $T_i^\dagger = R^i$ can be implemented in a similar fashion by incrementing bit values by 1 where needed. The preparation routine K_b for preparing $|\phi_0\rangle$ from some computational basis state $|b\rangle$ can either be constructed by the state preparation method using integrals (as discussed in section 4.2) or by finding some other unitary transformation that does $|b\rangle \to |\phi_r\rangle$. Assuming one such K_b can be efficiently implemented, then by Theorem 4.1 the Szegedy walk operator U_{walk} for the circulant transition matrix can be efficiently implemented.

In the case of an undirected graph, that is, $A = A^T$, we can construct the Szegedy walk using the above method. For example, the cycle graph C_N for some $N = 2^n$ (example shown in Figure 4.4) where $n \in \mathbb{Z}_+$ can be constructed by identifying $|\phi_0\rangle = [0, 1/\sqrt{2}, 0, ..., 0, 1/\sqrt{2}]^T$, which can be prepared using a Hadamard gate and a sequence of controlled-NOT gates. Figure 4.5 shows the quantum circuit for C_N

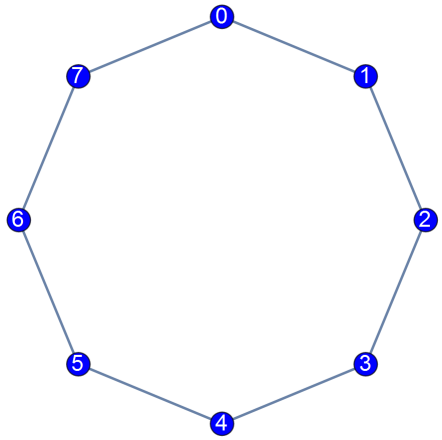


Figure 4.4: Cycle graph C_N with parameter $N = 8$.

with $|b\rangle = |N/2 - 1\rangle$.

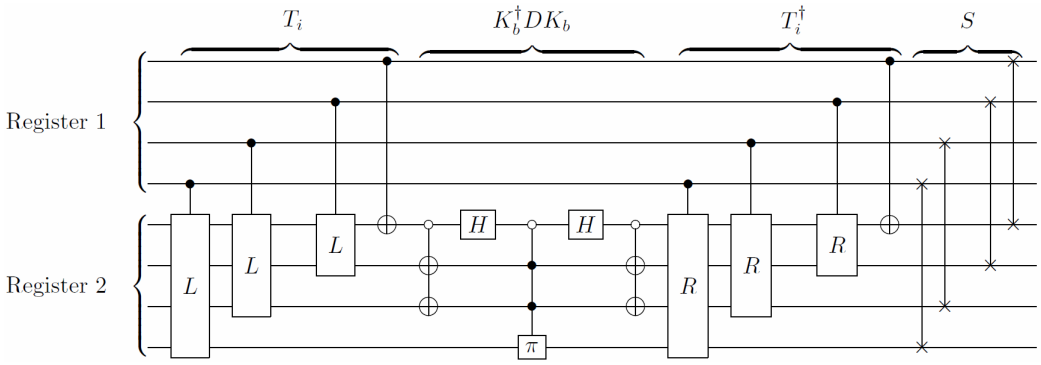


Figure 4.5: Quantum circuit implementing U_{walk} for the C_N graph.

More dense graphs, such as the complete graph K_N for some $N = 2^n$ (example shown in Figure 4.6) where $n \in \mathbb{Z}_+$, which is circulant with $|\phi_0\rangle = [0, \sqrt{1/(N-1)}, \ldots, \sqrt{1/(N-1)}]^T$, can be implemented in similar fashion using rotation gates for the preparation routine K_b, as shown in Figure 4.7 for $|b\rangle = |0\rangle$. Using this preparation routine, the quantum circuit implementing U_{walk} for K_N is shown in Figure 4.8.

Note, however, that we are not constrained to work with transition matrices P which correspond to a Markov chain on an undirected graph—any kind of circulant matrix where $|\phi_r\rangle$ can be efficiently prepared can be implemented in this fashion.

We can generalise this construction to any kind of P where the columns are formed by any cyclic permutation, by changing equation (4.11) into:

$$|\psi_i\rangle = |i\rangle \otimes R^{ix} |\phi_0\rangle, \tag{4.12}$$

for some $x \in \mathbb{Z}_+$ (or similarly for left-rotations), and then changing the transformations T_i accordingly to $T_i = (R^\dagger)^{ix} = L^{ix}$. Since $\{L, L^2, L^4, L^8, ...\}$ can be realised

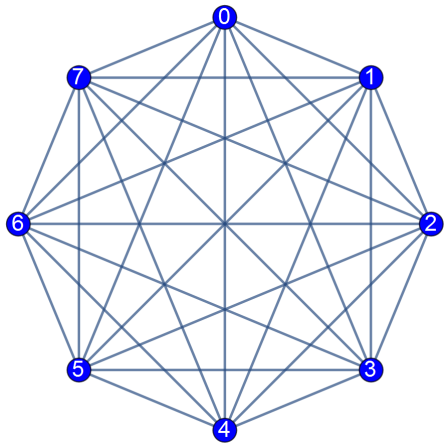


Figure 4.6: Complete graph K_N with parameter $N = 8$.

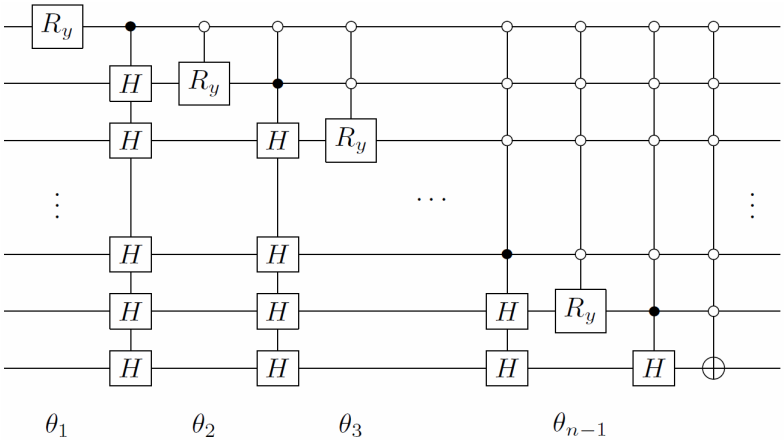


Figure 4.7: Quantum circuit implementing $K_b : |b\rangle \to |\phi_0\rangle$ for the complete graph K_{2^n}. The rotation angles are given by $\cos(\theta_i) = \sqrt{2^{n-i} - 1}{2^{n-i+1} - 1}$.

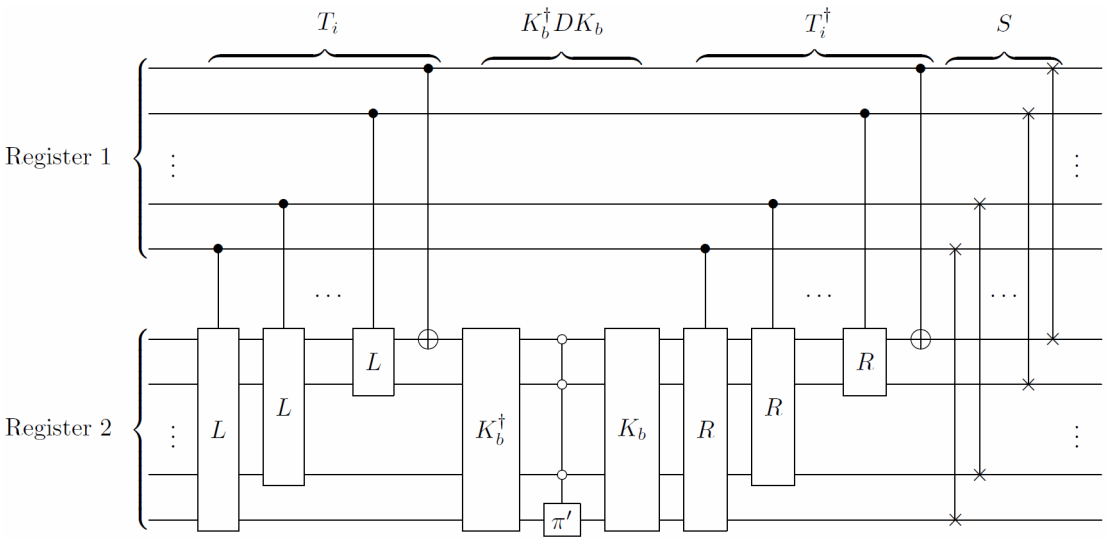


Figure 4.8: Quantum circuit implementing U_{walk} for the K_N graph, where the quantum circuit for the preparation routine K_b is given in Figure 4.7. The π' gate is equivalent to $\sigma_x \pi \sigma_x$, i.e. a -1 phase applied to the first state of the qubit.

efficiently, it follows that any L^x can be realised efficiently. Hence, every transformation of the form $T_i = (L^x)^i = L^{ix}$ can also be realised efficiently.

4.4 Complete bipartite graphs

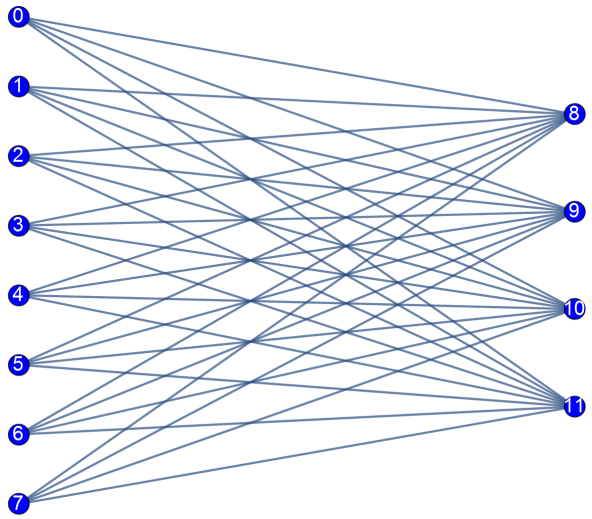


Figure 4.9: Complete bipartite graph K_{N_1,N_2} with parameters $N_1 = 8$ and $N_2 = 4$.

One particular class of dense graphs that can be efficiently implemented using the construction in section 4.2 is the class of complete bipartite graphs. A complete bipartite graph K_{N_1,N_2} (example shown in Figure 4.9) is a bipartite graph (i.e. the set of graph vertices is decomposable into two disjoint sets such that no two graph vertices within the same set are adjacent) such that every pair of graph vertices in the two sets are adjacent, with N_1 and N_2 vertices in the sets respectively. The transition matrix (i.e. the column-normalised adjacency matrix) of dimension $(N_1 + N_2)$-by-$(N_1 + N_2)$ for the graph K_{N_1,N_2} is:

$$P(K_{N_1,N_2}) = \begin{pmatrix} 0 & \frac{1}{N_1} J_{N_1,N_2} \\ \frac{1}{N_2} J_{N_2,N_1} & 0 \end{pmatrix}, \tag{4.13}$$

where $J_{a,b}$ denotes the all-1's matrix of dimension a-by-b. The graph suggests the natural partition $Z = Z_1 \cup Z_2$ where $Z_1 = \{0, \ldots, N_1 - 1\}$ and $Z_2 = \{N_1, \ldots, N_1 + N_2 - 1\}$. The transformations $T_{x,y}$ for either set are all identical, i.e. $T_{x,y} = I_{N_1+N_2}$, since within each subset every column of the transition matrix is equal. From equation (4.13), we can identify $|\phi_0\rangle = [0, \ldots, 0, 1/\sqrt{N_2}, \ldots, 1/\sqrt{N_2}]^T$ and $|\phi_{N_1}\rangle = [1/\sqrt{N_1}, \ldots, 1/\sqrt{N_1}, 0, \ldots, 0]^T$ as the reference states $|\phi_{r_1}\rangle$ and $|\phi_{r_2}\rangle$ respectively. Both reference states can be generated by integrating step functions, which are

efficiently integrable. Lastly, condition 2(a) in Theorem 4.2 is also satisfied by the definition of Z_1 and Z_2, since at most $O(\log(N_1+N_2))$ repetitions of U (with different sets of conditionals) are required to implement the controlled unitary operation in either case. Hence, by Theorem 4.2, U_{walk} for the complete bipartite graph K_{N_1,N_2} can be efficiently implemented. As a corollary of this result, the step operator for the star graph S_N can also be implemented efficiently, since S_N is equivalent to $K_{N,1}$.

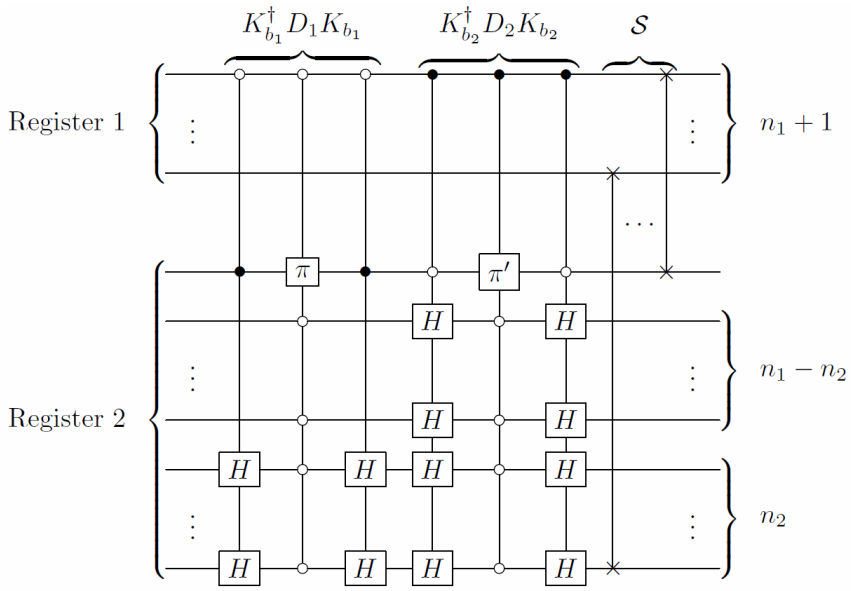


Figure 4.10: Quantum circuit implementing U_{walk} for the $K_{2^{n_1},2^{n_2}}$ graph.

In the case where $N_1 = N_2$, the complete bipartite graph is isomorphic to the circulant graph with $|\phi_0\rangle = [0, \sqrt{2/N_1}, 0, \sqrt{2/N_1}, ..., 0, \sqrt{2/N_1}]^T$. For the particular case where $N_1 = 2^{n_1}$ and $N_2 = 2^{n_2}$ (where $n_1, n_2 \in \mathbb{Z}_+$ and $n_1 \geq n_2$), the quantum circuit for the Szegedy walk operator U_{walk} can be implemented explicitly without using the state preparation method using integrals, as shown in Figure 4.10.

4.5 Tensor product of Markov chains

Similar to section 3.4, we can consider the composition of Markov chains to produce a new Markov chain. In the Szegedy walk formalism, the most natural composition of Markov chains is the tensor product, i.e. given two Markov chains defined by the transition matrices P_1 and P_2, a new Markov chain can be defined by $P = P_1 \otimes P_2$.

Theorem 4.3. *The Szegedy walk operator U_{walk} corresponding to the Markov chain with transition matrix $P = P_1 \otimes P_2$ can be efficiently implemented if the Szegedy walk operators $U_{\text{walk},1}$ and $U_{\text{walk},2}$ corresponding to the Markov chains with transition ma-*

trix P_1 and P_2 respectively can be efficiently implemented using the diagonalisation approach.

Proof. From equation (2.18), if the walk operator can be efficiently implemented using the diagonalisation approach, it follows that the associated reflection operator $\mathcal{R}$ can also be efficiently implemented. From equations (2.15), (2.16) and (4.4), we have that:

$$\mathcal{R}_1 = I_{N_1^2} - 2\left(\sum_{i=0}^{N_1-1} |i\rangle\langle i| \otimes |\phi_{i,1}\rangle\langle\phi_{i,1}|\right) \tag{4.14}$$

$$U_1\mathcal{R}_1U_1^\dagger = I_{N_1^2} - 2\left(\sum_{i=0}^{N_1-1} |i\rangle\langle i| \otimes |b_1\rangle\langle b_1|\right) \tag{4.15}$$

where $|\phi_{i,1}\rangle$ is the square root of the ith column of P_1. Similarly:

$$\mathcal{R}_2 = I_{N_2^2} - 2\left(\sum_{j=0}^{N_2-1} |j\rangle\langle j| \otimes |\phi_{j,2}\rangle\langle\phi_{j,2}|\right) \tag{4.16}$$

$$U_2\mathcal{R}_2U_2^\dagger = I_{N_2^2} - 2\left(\sum_{j=0}^{N_2-1} |j\rangle\langle j| \otimes |b_2\rangle\langle b_2|\right) \tag{4.17}$$

where $|\phi_{j,2}\rangle$ is the square root of the jth column of P_2. For the composite system:

$$\mathcal{R} = I_{N_1^2N_2^2} - 2\left(\sum_{k=0}^{N_1N_2-1} |k\rangle\langle k| \otimes |\phi_k\rangle\langle\phi_k|\right), \tag{4.18}$$

where $|\phi_k\rangle$ is the square root of the kth column of P. Writing $k = iN_2 + j$, this becomes:

$$\mathcal{R} = I_{N_1^2N_2^2} - 2\left(\sum_{i=0}^{N_1-1}\sum_{j=0}^{N_2-1} |i,j\rangle\langle i,j| \otimes |\phi_{i,1},\phi_{j,2}\rangle\langle\phi_{i,1},\phi_{j,2}|\right) \tag{4.19}$$

$$= I_{N_1^2N_2^2} - 2\left(\sum_{i=0}^{N_1-1}\sum_{j=0}^{N_2-1} |i\rangle\langle i| \otimes |j\rangle\langle j| \otimes |\phi_{i,1}\rangle\langle\phi_{i,1}| \otimes |\phi_{j,2}\rangle\langle\phi_{j,2}|\right), \tag{4.20}$$

since $P = P_1 \otimes P_2$. Now, if we apply the diagonalising operation U_1 to the $|i\rangle$ and $|\phi_{i,1}\rangle$ registers of $\mathcal{R}$:

$$U_1\mathcal{R}U_1^\dagger = I_{N_1^2N_2^2} - 2\left(\sum_{i=0}^{N_1-1}\sum_{j=0}^{N_2-1} |i\rangle\langle i| \otimes |j\rangle\langle j| \otimes |b_1\rangle\langle b_1| \otimes |\phi_{j,2}\rangle\langle\phi_{j,2}|\right). \tag{4.21}$$

Similarly, applying the diagonalising operation U_2 to the $|j\rangle$ and $|\phi_{j,2}\rangle$ registers of

$\mathcal{R}$ gives:

$$U_2 U_1 \mathcal{R} U_1^\dagger U_2^\dagger = I_{N_1^2 N_2^2} - 2\left(\sum_{i=0}^{N_1-1}\sum_{j=0}^{N_2-1} |i\rangle\langle i| \otimes |j\rangle\langle j| \otimes |b_1\rangle\langle b_1| \otimes |b_2\rangle\langle b_2|\right), \quad (4.22)$$

which can be rewritten as:

$$U_2 U_1 \mathcal{R} U_1^\dagger U_2^\dagger = I_{N_1 N_2} \otimes (I_{N_1 N_2} - 2|b\rangle\langle b|) \equiv D, \quad (4.23)$$

where $|b\rangle = |b_1, b_2\rangle$ and D can be readily implemented using a controlled-π gate (with multiple conditionals). Hence $\mathcal{R} = U_1^\dagger U_2^\dagger D U_2 U_1$ (and hence U_{walk}) can be efficiently implemented when $U_{\text{walk},1}$ and $U_{\text{walk},2}$ can be efficiently implemented using the diagonalisation approach, as required. □

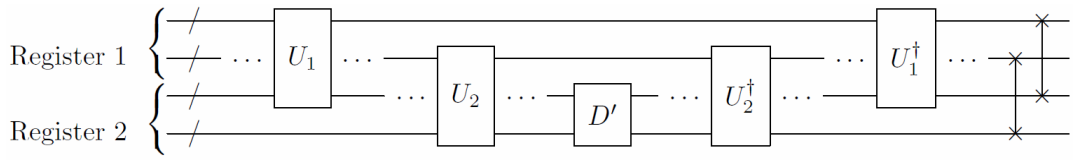


Figure 4.11: Quantum circuit implementing U_{walk} corresponding to the Markov chain with transition matrix $P = P_1 \otimes P_2$, with $D' = I_{N_1 N_2} - 2|b_1, b_2\rangle\langle b_1, b_2|$.

Figure 4.11 shows the general quantum circuit for implementing U_{walk} corresponding to the Markov chain with transition matrix $P = P_1 \otimes P_2$. The above result can also be easily extended to multiple systems as follows:

Corollary 4.3.1. *The Szegedy walk operator U_{walk} corresponding to the Markov chain with transition matrix $P = P_1 \otimes \ldots \otimes P_p$ can be efficiently implemented if the Szegedy walk operators $U_{\text{walk},1}, \ldots, U_{\text{walk},p}$ corresponding to the Markov chains with transition matrix $P_1, \ldots, P_p$ respectively can be efficiently implemented using the diagonalisation approach.*

Proof. This follows easily by applying Theorem 4.3 recursively. □

In the case of Markov chains on graphs described by adjacency matrices, the tensor product of transition matrices corresponds to taking the graph tensor product [108] of the underlying graphs. One important class of graphs formed from a tensor product of graphs is the bipartite double cover of a given graph G. If the adjacency matrix of G is given by A, then the bipartite double cover of G is given by the

adjacency matrix $A \otimes K_2$, where $K_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ [109]. From Theorem 4.3, it follows that if we can implement the Szegedy walk on the K_2 graph, then we can implement the Szegedy walk on the bipartite double cover of a given graph. Figure 4.12 shows the quantum circuit implementation of the Szegedy walk with transition matrix $P = K_2$.

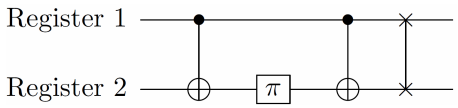


Figure 4.12: Quantum circuit implementing U_{walk} for the K_2 graph.

For example, the crown graph S_N^0 (example shown in Figure 4.13) can be formed by the bipartite double cover of the complete graph K_N [109], i.e. $S_N^0 = K_N \otimes K_2$. From section 4.3, we already have an efficient implementation for the Szegedy walk operator for the K_N graph (where $N = 2^n$ for some $n \in \mathbb{Z}_+$)—specifically, refer to Figures 4.7 and 4.8. Hence, applying Theorem 4.3, this means that we also have an efficient implementation for the crown graph S_N^0. By identifying U_1 and U_2 with the diagonalising operations in Figures 4.8 and 4.12 respectively, and taking $|b_1\rangle = |0\rangle$ and $|b_2\rangle = |1\rangle$ (giving $|b_1, b_2\rangle = |0, 1\rangle$), we can implement the Szegedy walk operator U_{walk} for the crown graph S_N^0, as shown in Figure 4.14.

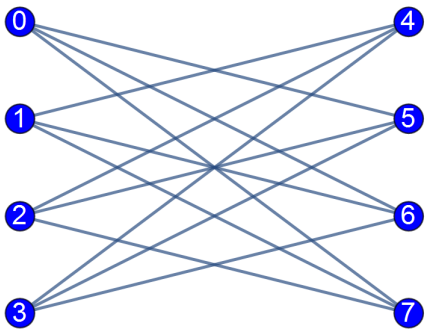


Figure 4.13: Crown graph S_N^0 with parameter $N = 4$.

4.6 Weighted interdependent networks

A composition of Markov chains in the form of a weighted interconnected network (a generalisation of the interdependent network discussed in 3.4.1) can also be considered. Suppose we have the transition matrices $P_A = \begin{pmatrix} P_{A_1} & 0 \\ 0 & P_{A_2} \end{pmatrix}$ and $P_B =$

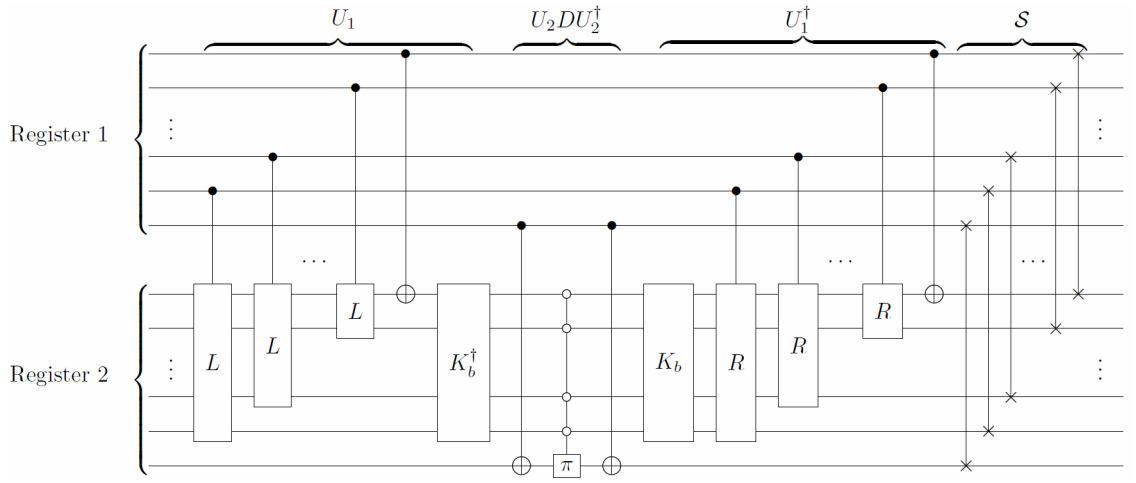


Figure 4.14: Quantum circuit implementing U_{walk} for the S_N^0 graph.

$\begin{pmatrix} 0 & P_{B_2} \\ P_{B_1} & 0 \end{pmatrix}$, where P_{A_1}, P_{A_2}, P_{B_1} and P_{B_2} each satisfy the column-normalisation constraint and are of dimensions N_1-by-N_1, N_2-by-N_2, N_2-by-N_1, and N_1-by-N_2 respectively. Then a weighted interdependent network formed from P_A and P_B has the form:

$$P = \begin{pmatrix} \alpha_1 P_{A_1} & \beta_1 P_{B_2} \\ \alpha_2 P_{B_1} & \beta_2 P_{A_2} \end{pmatrix}, \tag{4.24}$$

where $\alpha_1, \alpha_2, \beta_1, \beta_2 \in [0, 1]$ are independent real parameters that are chosen to preserve column-normalisation. Under some circumstances, it is possible to show that the Szegedy walk operator corresponding to P is efficiently implementable if the Szegedy walk operators corresponding to P_A and P_B are efficiently implementable. However, for this section, we will simply demonstrate the efficient implementation of a particular case.

Consider the weighted interdependent network (example shown in Figure 4.15) formed by connecting two disjoint cycle graphs C_{N_1} and C_{N_2} using complete interconnections (i.e. a complete bipartite graph K_{N_1,N_2}). Mathematically, this corresponds to:

$$P_A = \begin{pmatrix} \frac{1}{2}C_{N_1} & 0 \\ 0 & \frac{1}{2}C_{N_2} \end{pmatrix} \text{ and } P_B = \begin{pmatrix} 0 & \frac{1}{N_2}J_{N_1,N_2} \\ \frac{1}{N_1}J_{N_2,N_1} & 0 \end{pmatrix}, \tag{4.25}$$

which, from sections 4.3 and 4.4, we have an efficient implementation for the Szegedy walk operator of each. We can combine these Markov chains to form a weighted interconnected network, giving the transition matrix:

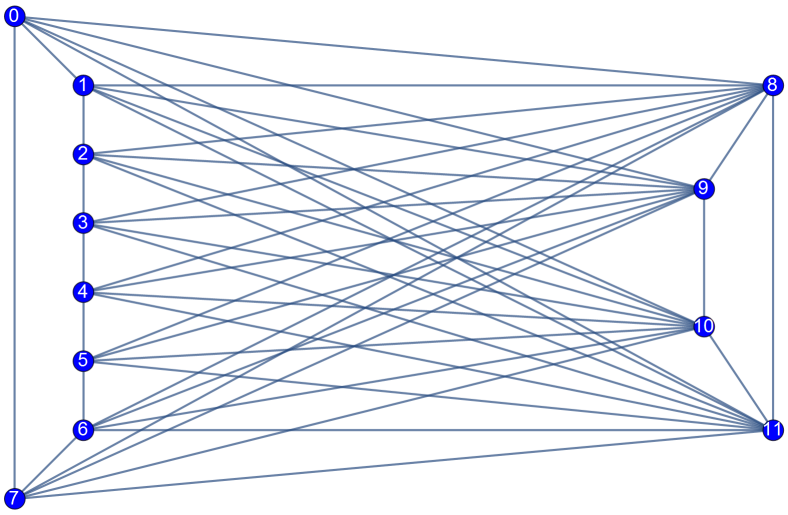


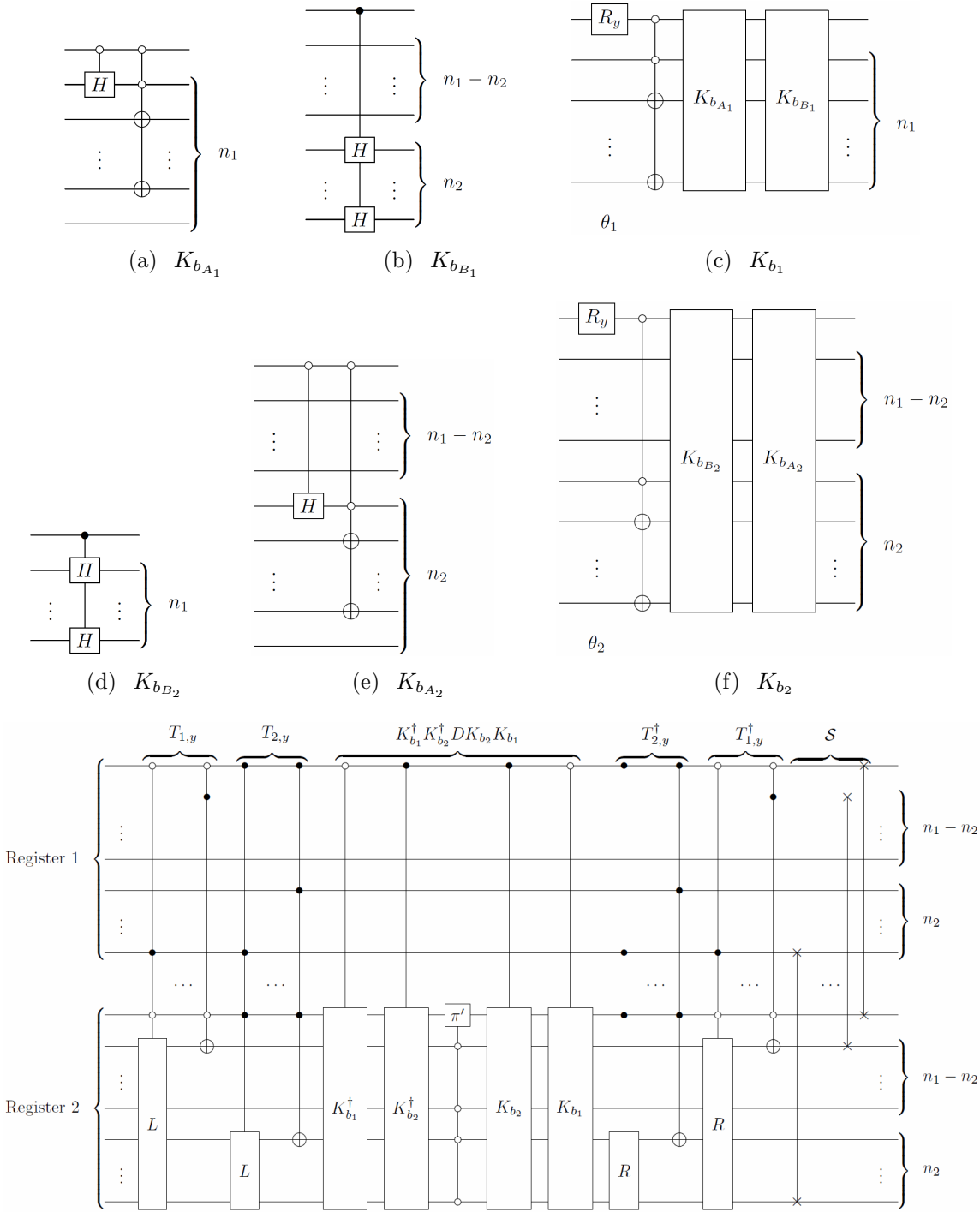
Figure 4.15: Weighted interdependent network with parameters $N_1 = 8$ and $N_2 = 4$.

$$P = \begin{pmatrix} \frac{1}{2+N_2} C_{N_1} & \frac{1}{2+N_1} J_{N_1,N_2} \\ \frac{1}{2+N_2} J_{N_2,N_1} & \frac{1}{2+N_1} C_{N_2} \end{pmatrix}, \tag{4.26}$$

where we have chosen the parameters α_1 and α_2 such that each non-zero element in a column is weighted equally. Set the partition as $Z = Z_1 \cup Z_2$ where $Z_1 = \{0, \ldots, N_1 - 1\}$ and $Z_2 = \{N_1, \ldots, N_1 + N_2 - 1\}$. For the subsets Z_1 and Z_2, each column has $(2 + N_2)$ and $(2 + N_1)$ non-zero elements respectively. Select the reference states as $|\phi_{r_1}\rangle = |\phi_0\rangle$ and $|\phi_{r_2}\rangle = |\phi_{N_1-1}\rangle$, where $|\phi_0\rangle = \frac{1}{\sqrt{2+N_2}}[0, 1, 0, \ldots, 0, 1, 1, \ldots, 1]^T$ and $|\phi_{N_1-1}\rangle = \frac{1}{\sqrt{2+N_1}}[1, \ldots, 1, 0, 1, 0, \ldots, 0, 1]^T$. For Z_1, the transformations $T_{1,y} : |\phi_y\rangle \to |\phi_0\rangle$ can be defined as a (restricted) cyclic permutation of the reference state $|\phi_0\rangle$, i.e. $T_{1,y} = L^y$ over the first N_1 rows. For Z_2, the transformations $T_{2,y} : |\phi_y\rangle \to |\phi_{N_1-1}\rangle$ can be defined in similar fashion as $T_{2,y} = L^{y-(N_1-1)}$ over the last N_2 rows. In the case where $N_1 = 2^{n_1}$ and $N_2 = 2^{n_2}$ (where $n_1, n_2 \in \mathbb{Z}_+$ and $n_1 \geq n_2$), the quantum circuit for the Szegedy walk operator U_{walk} can be implemented explicitly, as shown in Figure 4.16.

4.7 Application: Quantum Pagerank algorithm

The quantum Pagerank algorithm [9, 10] is a quantisation of the classical Google Pagerank algorithm [110] that is used to calculate the relative importance of nodes in a directed graph. Compared to the classical Pagerank algorithm, the quantum Pagerank algorithm is able to better distinguish secondary hubs and lower-ranking vertices better, and demonstrates increased stability with respect to variation in the



(a) $K_{b_{A_1}}$ (b) $K_{b_{B_1}}$ (c) K_{b_1}

(d) $K_{b_{B_2}}$ (e) $K_{b_{A_2}}$ (f) K_{b_2}

(g) Complete circuit for U_{walk}

Figure 4.16: The quantum circuit implementing U_{walk} for the weighted interdependent network formed from disjoint cycle graphs connected using complete interconnections is shown in (g). The circuits implementing the preparation routines $K_{b_1} : |0\rangle \to |\phi_0\rangle$ and $K_{b_2} : |0\rangle \to |\phi_{N_1-1}\rangle$ are shown in (c) and (f) respectively, with rotation angles $\cos(\theta_1) = \sqrt{\frac{2}{2+N_2}}$ and $\cos(\theta_2) = \sqrt{\frac{N_1}{2+N_1}}$ respectively.

damping parameter [10]. It utilises Szegedy quantum walks to quantise the classical Markov chain, and the average probability of the wavefunction at a node is taken to be the measure of relative importance of that node.

Suppose that we have a directed graph (representing a network) of N vertices described by a connectivity matrix C, where $C_{i,j} = 1$ if there is a link $j \to i$. Then, the patched connectivity matrix E is defined as:

$$E_{i,j} = \begin{cases} 1/N & \text{outdeg}(A)_j = 0 \\ C_{i,j}/\text{outdeg}(A)_j & \text{otherwise,} \end{cases} \tag{4.27}$$

The Google matrix G is then:

$$G = \alpha E + \frac{1-\alpha}{N} J, \tag{4.28}$$

where $\alpha \in [0, 1]$ is the damping parameter (typically chosen to be 0.85) and J is the all-1's matrix. Taking the transition matrix as $P = G$, we can define the Szegedy walk operator U_{walk} given by equation (2.18). Then the instantaneous quantum Pagerank of the jth vertex is given by:

$$Q(j, t) = |\langle j|_2 U_{\text{walk}}^{2t} |\Psi_0\rangle|^2, \tag{4.29}$$

where $\langle j|_2 = (|j\rangle_2)^\dagger$ and $|j\rangle_2$ is the jth standard basis vector of the second Hilbert space $\mathcal{H}_2$. The initial state $|\Psi_0\rangle$ is taken to be an equal superposition over the $|\psi_i\rangle$ as defined in equation (2.15), i.e.

$$|\Psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\psi_i\rangle. \tag{4.30}$$

The average quantum Pagerank for a vertex j, over some number of steps T, is defined as:

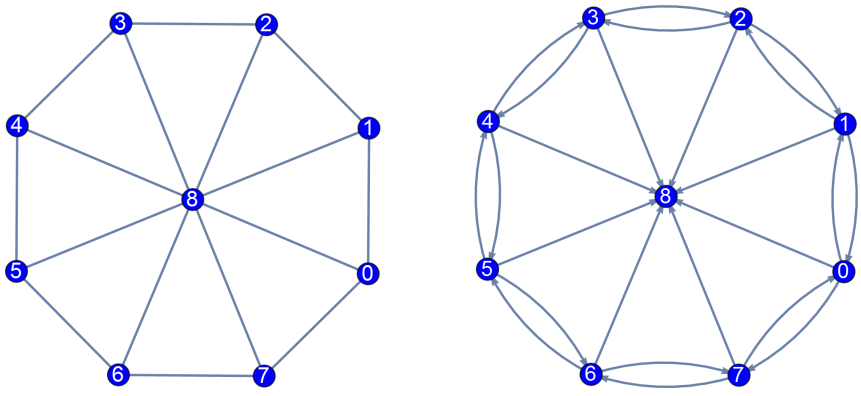
$$\langle Q(j)\rangle = \frac{1}{T} \sum_{t=0}^{T-1} Q(j, t), \tag{4.31}$$

which can be shown to converge for sufficiently large T—and this is the quantity that is called the quantum Pagerank of a graph [9, 10]. Here, we are interested in simulating the walk operator U_{walk} using the Google matrix G as the transition matrix, as this is necessary in order to compute the Google pagerank of the system.

Note that from the definition of G in equation (4.28), G is not a sparse matrix whenever $\alpha \neq 1$, so it is not simulable by the methods of Chiang *et al.* [40].

Now, if E is a matrix with columns related by cyclic permutations as defined before, then G also has the same property, since addition by $\frac{1-\alpha}{N}\mathbf{1}$ does not change the cyclic permutation property, i.e. T_i is the same as in section 4.3 (but K_b does change—see next paragraph). Of course, if all columns are related by cyclic permutations, then the Pagerank of each vertex would be equal—however we can exploit partitioning into subsets (from equations (4.6)–(4.9)) and cyclic symmetry to simulate the Szegedy walk using the Google matrix G as the transition matrix on a graph that has non-equivalent sets of vertices.

In general, in order to prepare any column state $|\phi_i\rangle$ of the Google matrix G, we note that if the corresponding column state $|\phi'_i\rangle$ of the patched connectivity matrix E can be prepared by the state preparation method using integrals (i.e. there exists an efficiently integrable function $p'_i(x)$ that generates the probability distribution $\{E_{j,i}\}$), then $p_i(x) = \alpha p'_i(x) + \frac{1-\alpha}{NL}$ (where L is the length of the domain of x) is also an efficiently integrable function that generates $\{G_{j,i}\}$, i.e. the required column state $|\phi_i\rangle$. Hence if Corollary 4.1.1 can be applied to the patched connectivity matrix E, then it follows that it can also be applied to the Google matrix G by changing the probability density function as above.



(a) Undirected (b) Directed

Figure 4.17: Undirected wheel graph W_N and its directed variant W'_N with parameter $N = 8$.

As before, in some special cases, we can construct the preparation routine K_b explicitly without using the state preparation method using integrals. One class of graphs for which this can be done is the wheel graph W_N (undirected and directed example shown in Figure 4.17) is a graph that contains a cycle graph C_N of length

N, which has each vertex connected to a single hub vertex. The connectivity matrix for the undirected case shown in Figure 4.17(a) is given by the block matrix:

$$C(W_N) = \begin{pmatrix} C(C_N) & 1 \\ 1 & 0 \end{pmatrix}, \tag{4.32}$$

and the directed case shown in Figure 4.17(b) is given by the modified block matrix:

$$C(W'_N) = \begin{pmatrix} C(C_N) & 0 \\ 1 & 0 \end{pmatrix}. \tag{4.33}$$

We consider $N = 2^n$ for some $n \in \mathbb{Z}_+$. For both W_N and W'_N, the vertices can be partitioned into the sets $Z_1 = \{0, \ldots, N-1\}$ and $Z_2 = \{N\}$. For the set Z_1, we pick the reference state as $|\phi_0\rangle$, which from equation (4.28), can be written as $|\phi_0\rangle = [\sqrt{\beta}, \sqrt{\gamma}, \sqrt{\beta}, \ldots, \sqrt{\beta}, \sqrt{\gamma}, \sqrt{\gamma}]^T$ where $\beta = \frac{(1-\alpha)}{N+1}$ and $\gamma = \frac{\alpha}{3} + \beta$. Setting $|b_1\rangle = |0\rangle$, we can use the circuit in Figure 4.18 to perform the operation $K_{b_1} : |b_1\rangle \to |\phi_0\rangle$.

The transformations $T_{1,y} : |\phi_y\rangle \to |\phi_0\rangle$ can be defined as a (restricted) cyclic permutation of the reference state $|\phi_0\rangle$, i.e. $T_{1,y} = L^y$ over the first N rows. For the set Z_2, there is only one state, so $T_{2,y}$ is not needed, and $K_{b_2} : |b_2\rangle \to |\phi_N\rangle$ can be constructed easily. Figure 4.19 shows the complete circuit that simulates the Szegedy walk for the wheel graph W_N, the total cost being within $O(\text{poly}(\log(N)))$ elementary gates.

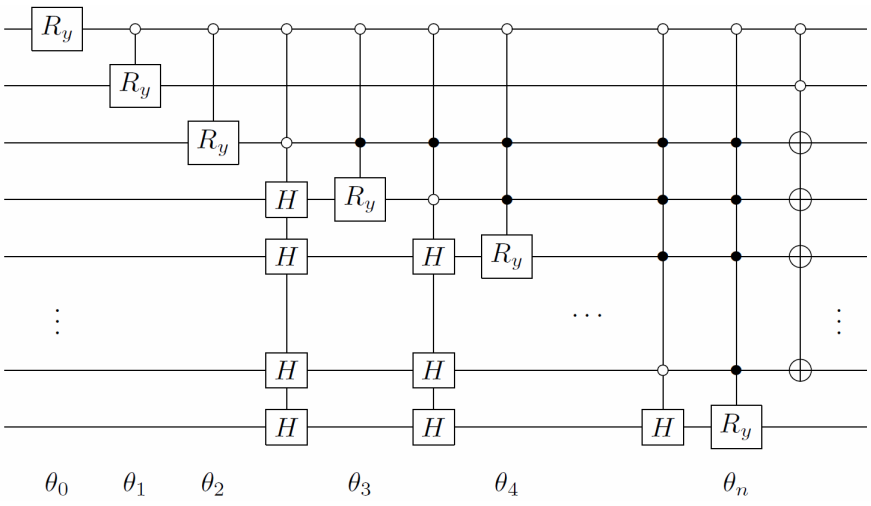


Figure 4.18: Quantum circuit implementing $K_{b_1} : |b_1\rangle \to |\phi_0\rangle$ for the vertex group Z_1 in the wheel graph W_{2^n}. The rotation angles are given by $\cos(\theta_0) = \sqrt{1-\gamma}$, $\cos(\theta_1) = \sqrt{\frac{1}{2}}$ and $\cos(\theta_{i>1}) = \sqrt{\frac{2^{n-i}\beta}{(2^{n-i+1}-1)\beta+\gamma}}$.

Running the Szegedy walk using U^2_{walk} as the walk operator, we obtain the instantaneous and average quantum Pagerank results, shown in Figure 4.20. As expected,

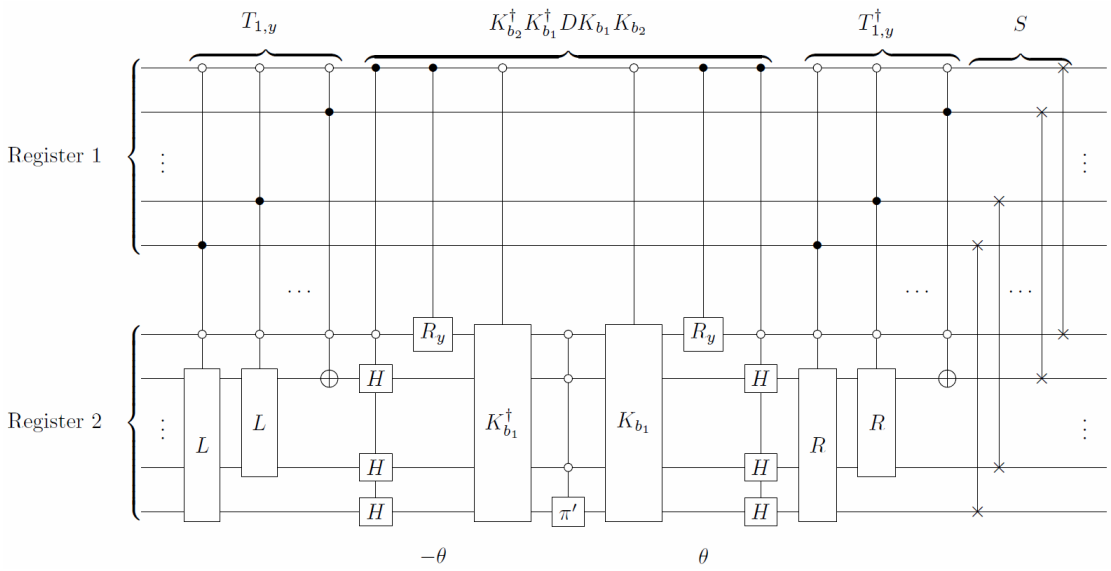


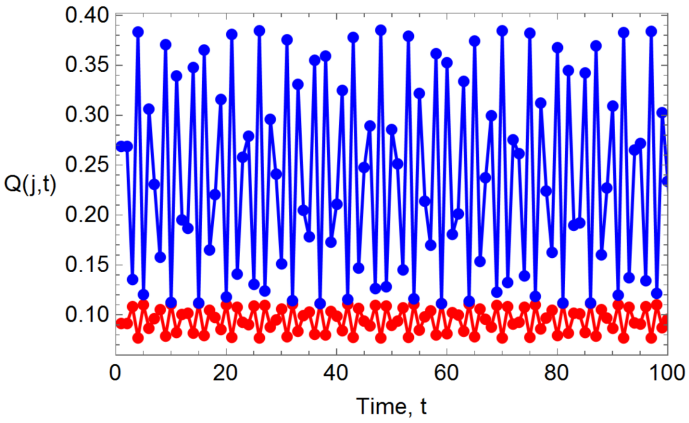
Figure 4.19: Quantum circuit implementing U_{walk} for the W_N and W'_N graph, where the quantum circuit for the preparation routine K_{b_1} is given in Figure 4.18. The rotation angle for K_{b_2} is given by $\cos(\theta) = \sqrt{1-\beta}$ for the W_N graph and $\cos(\theta) = \sqrt{\frac{N}{N+1}}$ for the W'_N graph.

the hub vertex (corresponding to the set Z_2) has a much higher quantum Pagerank value than the outer vertices (corresponding to the set Z_1), which all have the same quantum Pagerank value since they are equivalent vertices. In the case of the directed graph W'_8, the hub vertex has a slightly higher quantum Pagerank value compared to its value in the case of the undirected graph W_8, because of the lack of outgoing edges from the hub in W'_8.

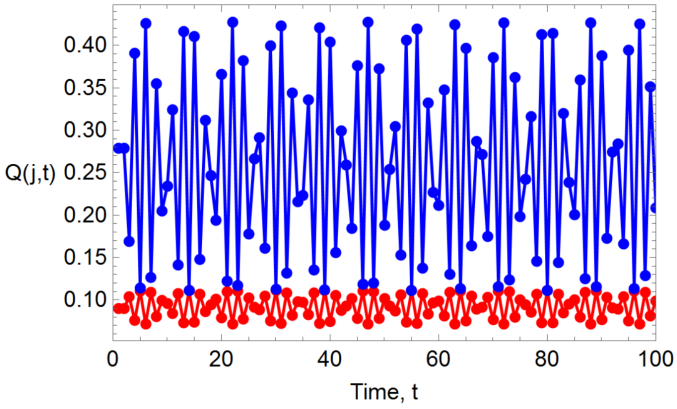
A second example of computing the quantum Pagerank on directed graphs is shown in Figure 4.21, the vertices of which can be partitioned into subsets of equivalent vertices as $Z = Z_1 \cup Z_2 \cup Z_3$ where $Z_1 = \{0,1,2,3\}$, $Z_2 = \{4,5\}$ and $Z_3 = \{6,7\}$. The connectivity matrix is given by:

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{4.34}$$

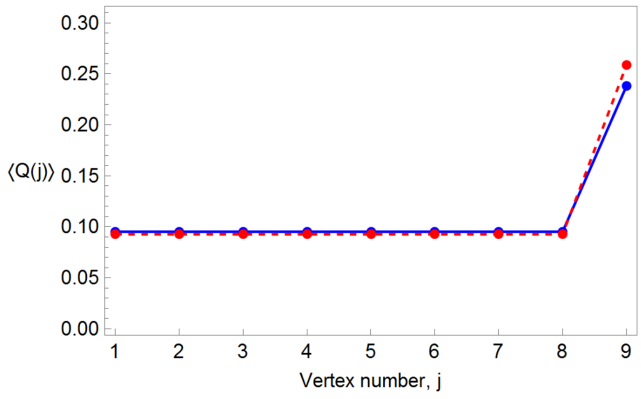
Set the basis state for each set to be $|b_1\rangle = |b_2\rangle = |b_3\rangle = |0\rangle$. For the set Z_1, we



(a) Instantaneous quantum Pagerank for W_8



(b) Instantaneous quantum Pagerank for W_8'



(c) Average quantum Pagerank

Figure 4.20: Instantaneous and average quantum Pagerank results for the wheel graph W_8. In (a) and (b), the small red curve and large blue curve denotes the instantaneous quantum Pagerank for vertices 1-8 and 9 respectively (as labelled in Figure 4.17). In (c), the solid blue curve and the dashed red curve denotes the averaged quantum Pagerank for W_8 and W_8' respectively.

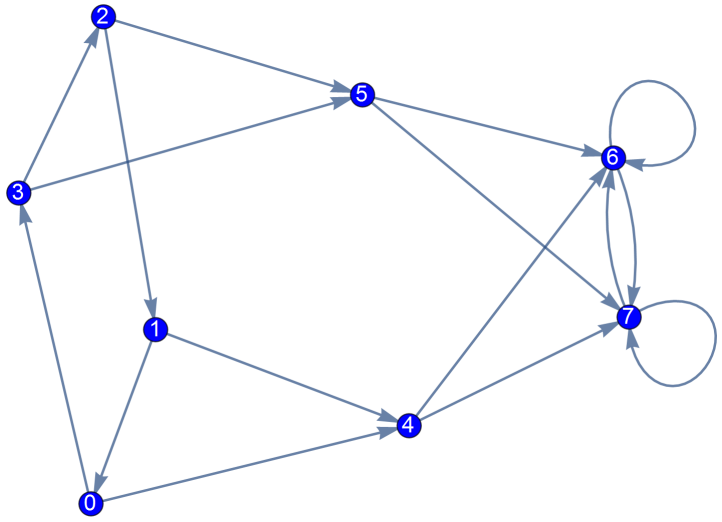


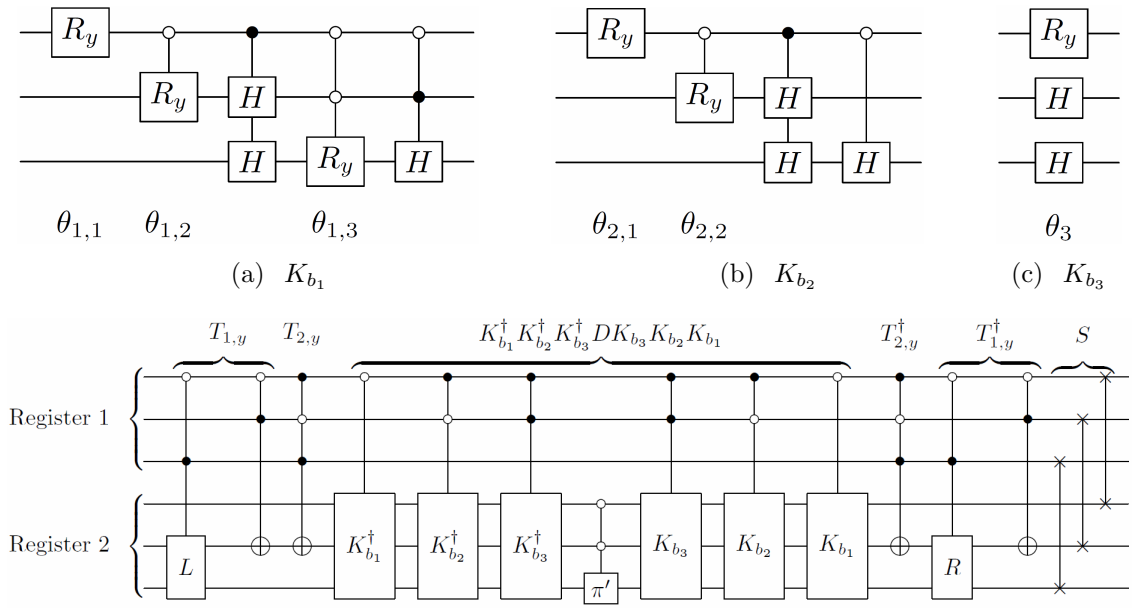
Figure 4.21: A directed graph on 8 vertices with three subsets of equivalent vertices.

pick the reference state as $|\phi_0\rangle$, which can be written as $|\phi_0\rangle = [\sqrt{\beta}, \sqrt{\gamma_1}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}]^T$ where $\beta = \frac{1-\alpha}{8}$ and $\gamma_1 = \alpha + \beta$. The required transformations $T_{1,y} : |\phi_y\rangle \to |\phi_0\rangle$ can be identified as $T_{1,y} = L^y$. For the set Z_2, we pick the reference state as $|\phi_4\rangle$, which can be written as $|\phi_4\rangle = [\sqrt{\gamma_2}, \sqrt{\gamma_2}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}]^T$ where $\gamma_2 = \frac{\alpha}{2} + \beta$. The required transformations $T_{2,y}$ that does the analogous transformation is simply $T_{2,4} = I$ and $T_{2,5} = L^2$. For the set Z_3, we pick the reference state as $|\phi_6\rangle$, which can be written as $|\phi_6\rangle = [\sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\beta}, \sqrt{\gamma_3}, \sqrt{\gamma_3}, \sqrt{\gamma_3}, \sqrt{\gamma_3}]^T$ where $\gamma_3 = \frac{\alpha}{4} + \beta$. Since $|\phi_6\rangle = |\phi_7\rangle$, no transformations are required. Figure 4.22 shows the quantum circuit implementing U_{walk} for this directed graph.

Running the Szegedy walk using U_{walk}^2 as the walk operator, we obtain the instantaneous and average quantum Pagerank results, shown in Figure 4.23. We find the centrality of each subset to be ordered (from highest to lowest) as Z_3, Z_2 and Z_1.

4.8 Conclusions and future work

In summary, we have presented a scheme that can be used to construct efficient quantum circuits for Szegedy quantum walks if the transition matrix of the Markov chain possesses translational symmetry in the columns and if the reference state $|\phi_r\rangle$ (or states $|\phi_{r_x}\rangle$) can be prepared efficiently. This scheme, which applies to both sparse and non-sparse matrices, allows for the efficient realisation of quantum algorithms based on Szegedy quantum walks. We have identified the class of cyclic permutations and complete bipartite graphs to be amenable to this scheme, as well a



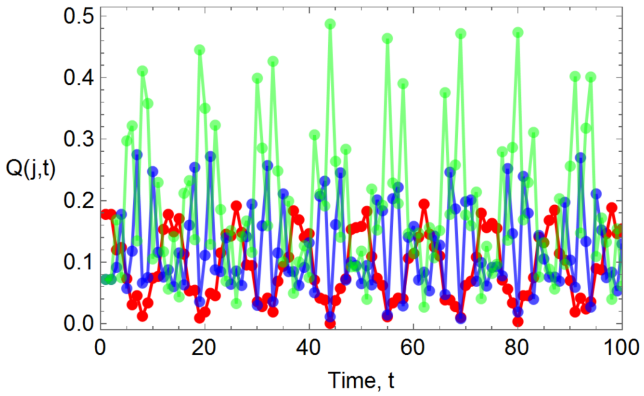
(a) K_{b_1} (b) K_{b_2} (c) K_{b_3}

(d) Complete circuit for U_{walk}

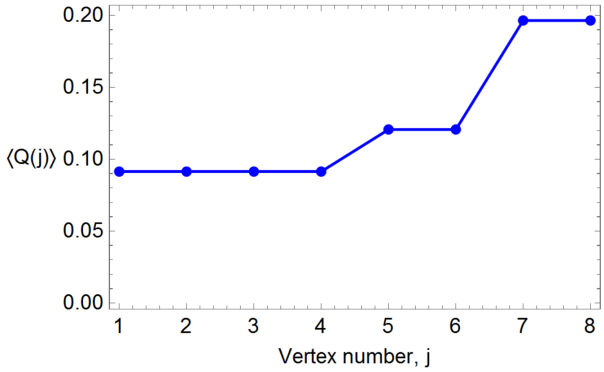
Figure 4.22: The quantum circuit implementation of U_{walk} for the directed graph is shown in (d), with circuits implementing the preparation routines $K_{b_1} : |0\rangle \to |\phi_0\rangle$, $K_{b_2} : |0\rangle \to |\phi_4\rangle$ and $K_{b_3} : |0\rangle \to |\phi_6\rangle$ in (a), (b) and (c) respectively. The rotation angles used in (a) to (c) are $\cos(\theta_{1,1}) = \sqrt{\frac{3\beta+\gamma_1}{7\beta+\gamma_1}}$, $\cos(\theta_{1,2}) = \sqrt{\frac{\beta+\gamma_1}{3\beta+\gamma_1}}$, $\cos(\theta_{1,3}) = \sqrt{\frac{\beta}{\beta+\gamma_1}}$, $\cos(\theta_{2,1}) = \sqrt{\frac{\beta+\gamma_2}{3\beta+\gamma_2}}$, $\cos(\theta_{2,2}) = \sqrt{\frac{\gamma_2}{\beta+\gamma_2}}$ and $\cos(\theta_3) = \sqrt{\frac{\beta}{\beta+\gamma_3}}$.

class of weighted interdependent networks. We have also applied our formalism to a tensor product of Markov chains, which further extends the classes of Markov chains for which the Szegedy walk can be efficiently simulated. Lastly, we have applied our results to construct efficient quantum circuits simulating Szegedy walks used in the quantum Pagerank algorithm, providing a means to experimentally demonstrate the quantum Pagerank algorithm.

A potential area for further research in this direction would be identifying other useful classes of transition matrices to which the formalism of section 4.2 can be applied. To begin with, we can generalise the idea in section 4.3. Suppose we have some preparation routine $K_b|b\rangle = |\phi_0\rangle$. Consider the class of transition matrices where $|\phi_{i+1}\rangle = U|\phi_i\rangle$, where U is some unitary operation. If the operations $\{U, U^2, U^4, U^8, \ldots\}$ can be efficiently implemented, then the Szegedy walk corresponding to the transition matrix can always be efficiently realised. In the case of section 4.3, we had $U = R^x$ or $U = L^x$ for some $x \in \mathbb{Z}_+$. We can obtain different classes of transition matrices by changing U—choices such as rotation matrices $R_y(\theta)$ and phase shift matrices $R_z(\phi)$ satisfy the required condition, since $R_y(\theta)^x = R_y(x\theta)$ and $R_z(\phi)^x = R_z(x\phi)$ for any $x \in \mathbb{Z}_+$. Other classes of graphs that are useful in



(a) Instantaneous quantum Pagerank



(b) Average quantum Pagerank

Figure 4.23: Instantaneous and average quantum Pagerank results for the directed graph shown in Figure 4.21. In (a), the red, blue and green curves denote the instantaneous quantum Pagerank for vertices 1–4, 5–6 and 7–8 respectively.

quantum algorithms, such as complete regular trees, can also be analysed using the above formalism to see if an efficient implementation can be obtained.

Chapter 5

Quantum compilation

Based on T. Loke, J.B. Wang and Y.H. Chen's original publication in Computer Physics Communications 185.12 (Dec. 2014) pp. 3307–3316 (2014) [42] (with revisions in Computer Physics Communications 207 (Oct. 2016) pp. 531–532 [111]), with an expanded introduction on the cosine-sine decomposition method.

5.1 Introduction

The task of efficiently implementing a given quantum walk exactly (or quantum algorithm in general) represented by a unitary matrix, is in general an impossible one. To demonstrate this, we note that a N-by-N unitary operator has $N^2 - 1$ complex parameters, so a quantum circuit implementing the unitary operator exactly would have at least $O(N^2)$ elementary gates. When N scales exponentially, the number of gates required would also scale exponentially—for this reason, any generic quantum compiler that performs an exact decomposition will in general provide a quantum circuit with exponentially many quantum gates. Nevertheless, it is still possible to optimise a quantum compiler so as to reduce the number of gates required on a case-by-case basis—in fact, this is of vital importance in any practical application of a quantum compiler to be able to do so as much as possible.

Earlier studies into quantum compilation applied the standard triangularisation or QR-factorisation scheme with Givens rotations and Gray codes to map a quantum algorithm to a series of elementary gate operations [19, 45, 69, 70]. Several research groups examined a more efficient and versatile scheme based on the CSD (cosine-sine decomposition) [72–74, 112, 113]. De Vos *et al.* [114, 115] looked into another decomposition scheme, namely the Birkhoff decomposition, which was found

to provide simpler quantum circuits for certain types of unitary matrices than the cosine-sine decomposition. However, the Birkhoff decomposition does not work for general unitary matrices. More recently, Chen and Wang [48] developed a general quantum compiler package written in Fortran, entitled the *Qcompiler*, which is based on the CSD scheme and works for arbitrary unitary matrices. The number of gates required to implement a general 2^n-by-2^n unitary matrix using the CSD method scales as $O(4^n)$ [73, 116].

In this work, we adopt the CSD method to decompose a given unitary into a quantum circuit. Specifically, we build on Chen and Wang's *Qcompiler* package [48] to produce *OptQC* [42], a package that utilises optimisation techniques to reduce the number of required gates in the quantum circuit provided by the CSD method. We do this by splitting the unitary matrix U into an equivalent sequence of unitaries such that the entire sequence of unitaries has a lower total cost than the original unitary matrix. In general, this means writing U as a sequence of s unitaries, i.e. $U = U_s U_{s-1} \dots U_1$. At first glance, this seems counterintuitive, since if we were to apply the CSD to each unitary, this would increase the scaling of the number of gates required to $O(4^n s)$, which is undesirable. However, we note that (1) certain U_i can be decomposed more efficiently without using the CSD method; and (2) some matrices require only a few gates when separately decomposed using the CSD method.

This chapter is organised as follows. Section 5.2 provides a short introduction to the CSD method. Section 5.3 provides an overview of our approach for reducing the number of gates required to implement any given unitary matrix U using the CSD method. Section 5.4 details our developed program, called *OptQC*, based on the methods described in section 5.3. Some sample results using the program are given in section 5.5, and then we discuss our conclusions and possible future work in section 5.6.

5.2 Cosine-sine decomposition

The CSD method is used to a decompose unitary matrix U into a product of unitary matrices [48, 71, 72, 117]. Suppose U has dimension N-by-N where $N = 2^n$. Then the decomposition in block matrix form is:

$$U = \begin{bmatrix} u & \\ & v \end{bmatrix} \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} x & \\ & y \end{bmatrix}, \tag{5.1}$$

where:

$$C = \text{diag}\{\cos(\theta_1), \ldots, \cos(\theta_{2^{n-1}})\} = \begin{bmatrix} \cos(\theta_1) & & & \\ & \cos(\theta_2) & & \\ & & \ddots & \\ & & & \cos(\theta_{2^{n-1}}) \end{bmatrix}, \tag{5.2}$$

and:

$$S = \text{diag}\{\sin(\theta_1), \ldots, \sin(\theta_{2^{n-1}})\} = \begin{bmatrix} \sin(\theta_1) & & & \\ & \sin(\theta_2) & & \\ & & \ddots & \\ & & & \sin(\theta_{2^{n-1}}) \end{bmatrix}, \tag{5.3}$$

are 2^{n-1}-by-2^{n-1} diagonal matrices. The matrices u, v, x and y can be further decomposed by applying the CSD method recursively to the lowest level [73]. After applying the CSD method recursively, the resulting product of unitaries can be mapped to controlled rotation gates $R_y(\theta)$[1] and $R_z(\phi)$. Figure 5.1 shows the quantum circuit obtained from recursively applying the CSD to a generic 8-by-8 unitary matrix—the half-open and half-closed circles represent the uniformly controlled rotational gates defined in Mottonen *et al.* [73].

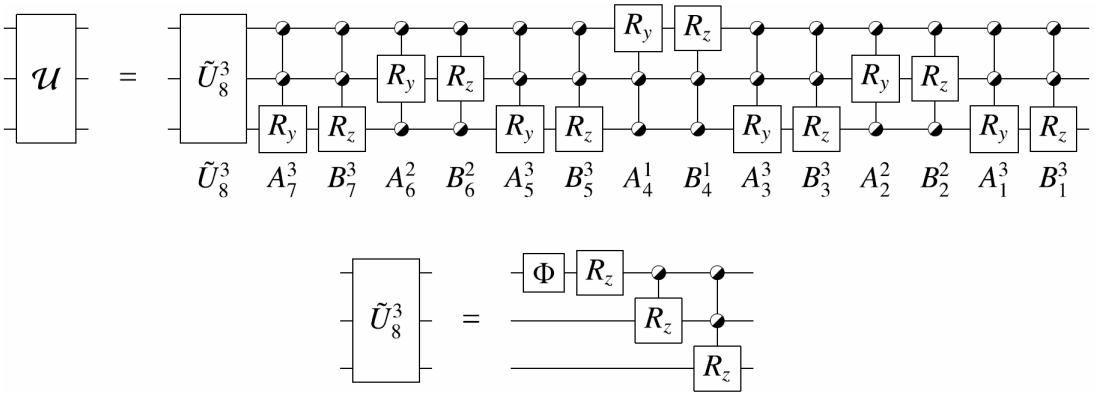


Figure 5.1: Quantum circuit for an 8-by-8 complex unitary matrix—sourced directly from Chen and Wang [48].

In general, all $O(4^n)$ gates are required to implement a unitary operator. However, in practice, a number of the rotation angles θ and ϕ are zero, so the corre-

[1]For the remainder of this chapter, treat all $R_y(\theta)$ gates as having the opposite sign convention to the same gate in section 2.1.2, i.e. the θ here is equivalent to $-\theta$ in said section.

sponding controlled rotations can be removed from the circuit, reducing the overall cost. For our purposes, we can mostly treat the CSD method as a black-box, which takes in some unitary matrix and then provides a quantum circuit composed of controlled rotation gates. Hence, define CSD(M) as the number of gates required to implement the unitary matrix M according to the CSD method.

5.3 Optimising the CSD

As briefly discussed before, we want to optimise (that is, reduce the number of quantum gates required to implement a given unitary) the CSD method by splitting the given unitary matrix U into an equivalent sequence of unitaries which has a lower overall cost. To achieve this, we utilise similarity transforms by permutation matrices heavily. A permutation matrix is a square binary matrix that contains, in each row and column, precisely a single 1 with 0s everywhere else. For any permutation matrix P, its corresponding inverse is $P^{-1} = P^T$. For convenience, we also define an equivalent representation of permutations using lists—a permutation list p (lowercase) is equivalent to the permutation matrix P (uppercase) by the relation:

$$(P)_{i,j} = \delta_{p[\![i]\!],j}, \tag{5.4}$$

where δ is the Kronecker delta function, and $p[\![i]\!]$ denotes the ith list element of p. For example, the permutation list $p = \{2, 3, 1, 4\}$ corresponds to the 4-by-4 permutation matrix:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Our initial approach to optimising the CSD was to write U as $U = P^T U' P$ where U' is related to U by that similarity transform. In general, we find that applying the CSD to U yields a different cost than the total cost of applying the CSD to P^T, U' and P, i.e.

$$\mathrm{CSD}(U) \neq \mathrm{CSD}(P^T) + \mathrm{CSD}(U') + \mathrm{CSD}(P). \tag{5.5}$$

In fact, it is possible to find P's such that the right-hand side of equation (5.5) gives a lower cost than the original cost CSD(U). This can be cast into a global discrete optimisation problem by identifying the cost function explicitly as:

$$c_{num}(U, P) = \text{CSD}(PUP^T) + \text{CSD}(P) + \text{CSD}(P^T), \tag{5.6}$$

where P is the discrete variable that we optimise the cost function c_{num} over for a given unitary matrix U. In practice, the permutation matrix P is formed by a product of two other permutation matrices of specific types. The similarity transform that we use is:

$$U = Q^T P^T U' P Q, \tag{5.7}$$

where P here is a (general) permutation matrix formed by a sequence of NOT and generalised controlled-NOT gates, and Q is a (qubit) permutation matrix formed by a sequence of SWAP gates. In principle, there are $N! = (2^n)!$ possible P's and $n!$ possible Q's. The advantage of this approach is that the CSD method only has to be applied once to U' and not to the permutation matrices—instead, we form the permutation matrices using a sequence of elementary gates which in most cases provides a permutation matrix with a lower cost than by applying the CSD method. Q is treated separately from P because Q allows for the parallel version of the *OptQC* program to start different threads at different points in the search space.

Denote the number of gates required to form P and Q as $g_{num}(P)$ and $s_{num}(Q)$ respectively. Note that $g_{num}(P) = g_{num}(P^T)$ and $s_{num}(Q) = s_{num}(Q^T)$, since the inverse permutation is formed by simply applying the NOT, generalised controlled-NOT, or SWAP gates in reverse order, so it has the same gate count. The new cost function is thus:

$$c_{num}(U, P, Q) = \text{CSD}(PQUQ^T P^T) + 2g_{num}(P) + 2s_{num}(Q), \tag{5.8}$$

which we optimise over the discrete variables P and Q.

5.4 Program outline

We have developed a Fortran program, called *OptQC*, which reads in a unitary matrix U, minimises the total cost function $c_{num}(U, P, Q)$ given in equation (5.8),

and outputs a quantum circuit that implements U. A significant portion of this program is based on the CSD code provided by the LAPACK library [118]) and the recursive procedure implemented in *Qcompiler*, developed by Chen and Wang [48]. As with *Qcompiler*, the new *OptQC* program has two different branches, one treating strictly real unitary (i.e. orthogonal) matrices, and another treating arbitrary complex unitary matrices, with the former generally providing a circuit that is half in size of the latter [48].

Note that the CSD procedure requires the round up of the matrix dimension to the closest power of two. Hence, in cases where N is not a power of two, the actual dimension used is:

$$N' = 2^{\lceil \log_2 N \rceil}. \tag{5.9}$$

The expanded unitary operator $\bar{U}$ is an N'-by-N' matrix, where:

$$\left(\bar{U}\right)_{i,j} = \begin{cases} (U)_{i,j} & : i \leq m, j \leq m \\ \delta_{i,j} & : \text{otherwise} \end{cases} \tag{5.10}$$

which we will subsequently treat as the unitary U to be optimised via permutations.

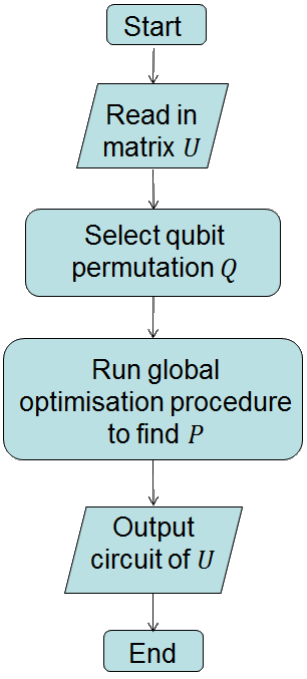


Figure 5.2: Flowchart overview of the serial version of *OptQC*. Details about the qubit permutation Q selection procedure and global optimisation procedure to find P is provided in Figures 5.4 and 5.5 respectively.

In the following subsections we describe the key procedures in *OptQC*, depicted in Figure 5.2, which serve to progressively reduce the total cost function $c_{num}(U, P, Q)$.

We first detail the serial version of the program, followed by an extension to a parallel architecture using MPI (Message Passing Interface).

5.4.1 Selection of qubit permutation

Qubit permutations are a class of permutations that are expressible in terms of a reordering of qubits, which can be efficiently implemented using SWAP gates that serve to interchange qubits. Recalling that U is of dimensions N-by-N (where $N = 2^n$), this implies that there are only $n!$ qubit permutations possible for a given U. A qubit permutation can be expressed as a list q (lowercase) of length n, or as a permutation matrix Q (uppercase) of dimensions N-by-N. A qubit permutation of length n requires at most $n - 1$ SWAP gates.

The selection of the qubit permutation matrix Q is done by varying Q and computing the corresponding change in the cost function c_{num}, while holding P constant as the identity matrix I. An example implementation of the $n = 3$ qubit permutation $q = \{3, 1, 2\}$ is shown in Figure 5.3. By considering how the basis states are mapped to each other by q, a regular permutation list $\bar{q}$ of length N that permutes the states can be readily constructed from q, and then we use the relation between permutation lists and permutation matrices (see equation (5.4)) to obtain Q from $\bar{q}$.

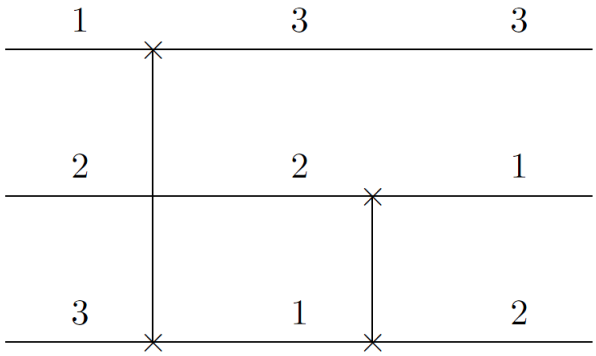


Figure 5.3: Example implementation of qubit permutation $q = \{3, 1, 2\}$ using SWAP gates.

We start the program with an identity qubit permutation q, i.e. $q[[i]] = i$ (corresponding to $Q = I$), and compute the corresponding cost of implementation $c_{num}(U, I, Q)$. Then, for some prescribed number of iterations j_{max}, we generate a random qubit permutation q' each time and compute the new cost as $c_{num}(U, I, Q')$. If the new cost is lower than the initial cost (recorded by $c_{num}(U, I, Q)$), the current qubit permutation q is replaced by q'. Figure 5.4 shows a flowchart overview of

the qubit selection procedure. After this procedure, we have an optimised qubit permutation matrix Q, which will remain unchanged while we find the unrestricted permutation matrix P in the next section through the main optimisation procedure.

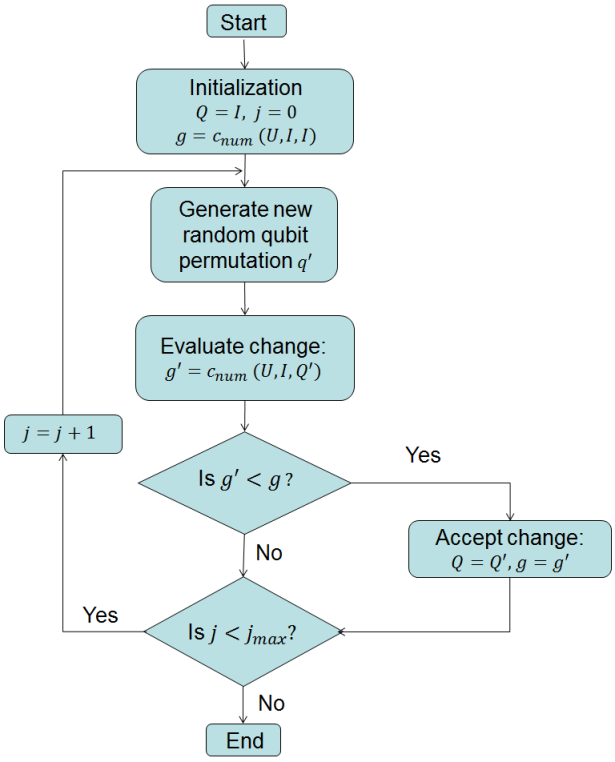


Figure 5.4: Flowchart overview of the qubit selection procedure.

5.4.2 Optimisation procedure

Here, we aim to find an optimal permutation p' such that $c_{num}(U, P', Q) < c_{num}(U, P, Q)$ in the discrete search space of all $N!$ permutations. Given the massive size of the search space, use of a heuristic optimisation method is practically necessary. Our initial approach for minimizing the quantity c_{num} was to use a threshold acceptance based simulated annealing algorithm [42]—however, we found that in practice, a plain descending random walk (equivalent to a simulated annealing algorithm with zero acceptance threshold) works better in most cases [111].

In either case, we define a procedure (called the neighbourhood operator) that changes the current permutation p to a slightly different permutation p'. As described in section 5.3, we form P by a sequence of NOT and generalised controlled-NOT gates, so we choose our neighbourhood operator such that given P, it produces P' by adding a NOT or generalised controlled-NOT gate to the circuit for P. In a descending random walk, if the new cost $c_{num}(U, P', Q)$ is lower than the current

cost $c_{num}(U, P, Q)$, then p' replaces p as the current permutation.

We start the optimisation procedure with p as the identity permutation, i.e. $p = I = \{1, 2, \ldots, N\}$. By iterating the neighborhood operator and evaluating the subsequent change in the number of gates, we accept the change in the permutation if it reduces the number of gates. After some prescribed number of iterations i_{max}, we terminate the optimisation procedure, returning the permutation p_{min} that provides the minimum number of gates. Figure 5.5 shows a flowchart overview of the optimisation procedure.

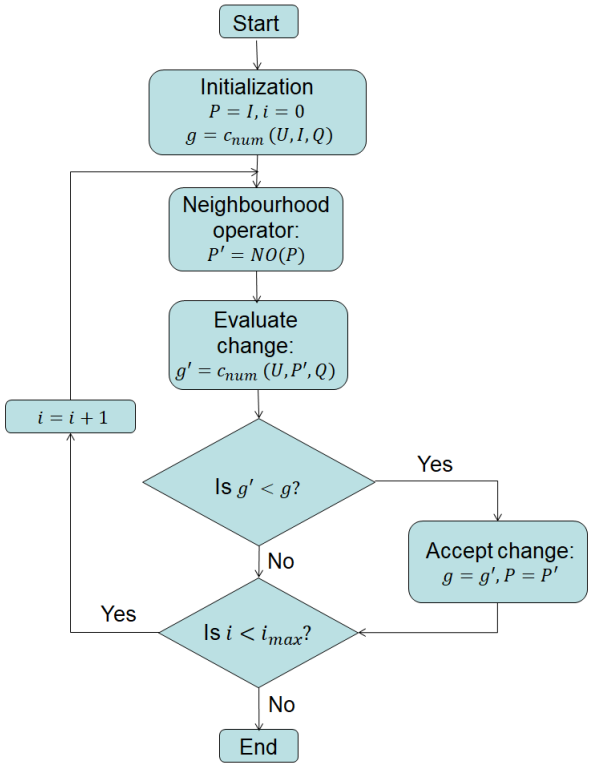
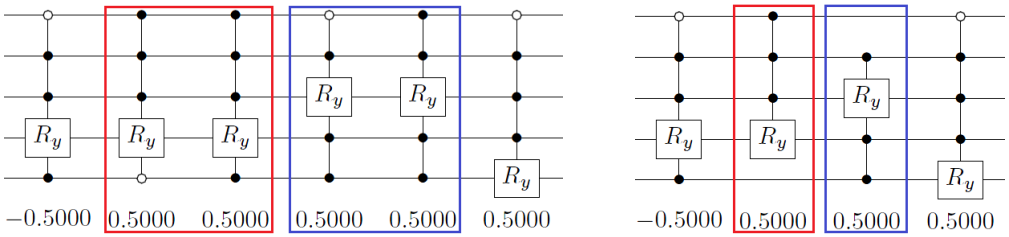


Figure 5.5: Flowchart overview of the optimisation procedure.

5.4.3 Gate reduction procedure

Here, we focus on reducing the number of gates in some prescribed quantum circuit by combining 'similar' gates. In a quantum circuit, we can combine CUGs (controlled unitary gates) that apply the same unitary operation U_{op} to the same qubit, with all but one of the conditionals of the CUGs being the same. This reduction process is carried out after every application of the CSD method to a matrix—in particular, it is applied to the CSD result of $PQUQ^TP^T$ when computing $c_{num}(U, P, Q)$ (see equation (5.8)). While it does impose a significant computational overhead, it gives a better reflection of the true cost function, since the reduced circuit is the

circuit that one would use for implementation. Figure 5.6 shows an example result of applying the reduction procedure to a quantum circuit.



(a) Circuit before reduction procedure

(b) Circuit after reduction procedure

Figure 5.6: Example of applying the reduction procedure to a quantum circuit.

5.4.4 MPI parallelisation

The program described above can be readily extended to a parallel architecture using MPI. Since the neighbourhood operator in the optimisation procedure acts to interchange any two random positions, it follows that if the random number generator is seeded differently, then a different set of positions would be interchanged, i.e. a different search through the space of permutations would be conducted. Similarly, the qubit permutation that is generated would also change when seeded differently, which enables the program to start threads at multiple locations in the search space of $n!$ permutations, so that the search procedure explores as much of the permutation space as possible. We do, however, restrict the root thread (thread index 0) of the program to use the identity qubit permutation for comparison purposes. Hence, using MPI, we can spawn a team of threads that simultaneously searches through the space of permutations independently and differently (by seeding the random number generator of each thread differently), and then collate the results to pick out the thread with the most optimal permutation, that is, it has the lowest $c_{num}(U, P, Q)$ value.

As of version 1.3 of *OptQC* [111], a synchronisation mechanism between threads (after some prescribed number of iterations s_{max}) that copies over the current state of the top (i.e. possessing lowest cost) 10% processes to the remaining 90% was added. This works so as to discard the less fit solutions and focus the searching algorithm in the state space with the fittest solutions.

5.5 Results

We now apply the software program *OptQC*[2] to various unitary operations to obtain corresponding optimised quantum circuits. All the results shown here are obtained using parameters $i_{max} = 40000$, $j_{max} = 1000$ and $s_{max} = 15000$. Using these parameters, we ran *OptQC* on the supercomputer Fornax with Intel Xeon X5650 CPUs, managed by iVEC@UWA, using 8 nodes with 12 cores on each (i.e. 96 threads).

5.5.1 Real unitary matrix

A random real unitary (i.e. orthogonal) matrix is given below:

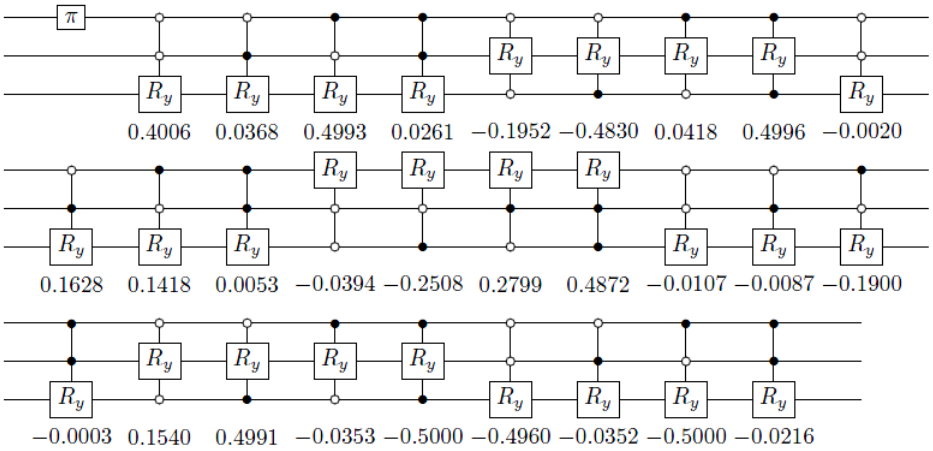
$$U = \begin{pmatrix} 0.0438 & 0 & 0 & 0 & 0.9990 & 0 & 0 & 0 \\ 0.1297 & 0.8689 & -0.2956 & 0 & -0.0057 & 0.1538 & -0.3423 & 0 \\ -0.2923 & 0 & 0.6661 & 0 & 0.0128 & 0 & -0.6861 & 0 \\ -0.0061 & -0.0412 & 0.0140 & 0.7058 & 0.0003 & 0.3008 & 0.0162 & -0.6397 \\ 0.9147 & 0 & 0.4021 & 0 & -0.0401 & 0 & 0 & 0 \\ 0.0185 & 0.1242 & -0.0422 & 0.3961 & -0.0008 & -0.9073 & -0.0489 & 0 \\ 0.2424 & -0.4762 & -0.5524 & 0 & -0.0106 & 0 & -0.6397 & 0 \\ 0.0051 & 0.0343 & -0.0117 & -0.5874 & -0.0002 & -0.2503 & -0.0135 & -0.7686 \end{pmatrix}$$

Note that this matrix is not completely filled, otherwise no optimisation via permutations would generally be possible. By using *OptQC*, the optimisation process gives the following results for the thread which achieves the optimal solution:

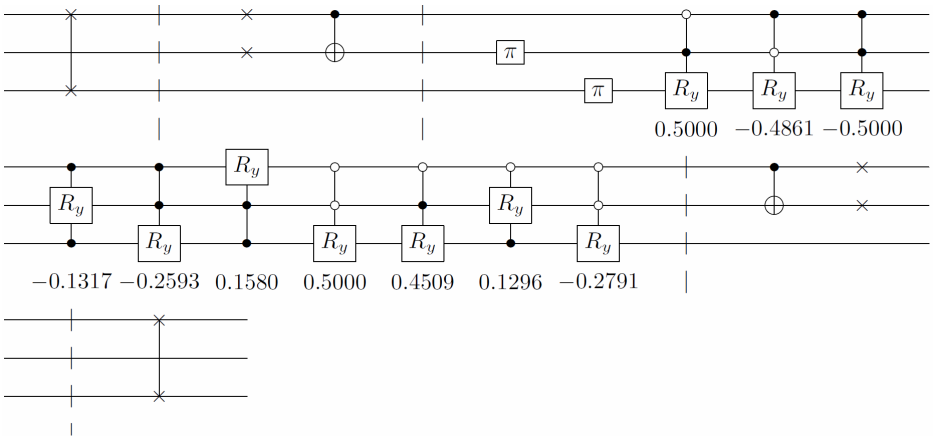
- No optimisation: $c_{num}(U, I, I) = 29$ gates
- After selection of an optimised qubit permutation q: $c_{num}(U, I, Q) = 1 + 26 + 1 = 28$ gates
- After optimisation procedure for the permutation p: $c_{num}(U, P, Q) = 1 + 2 + 12 + 2 + 1 = 18$ gates

Hence, we achieve a reduction of $\sim$ 38% from the original number of gates. Figure 5.7 shows a comparison between the original and optimised circuit for U. Runtime for this calculation is $\sim$ 8.5 seconds.

[2]The source code for version 1.3 of *OptQC* is freely available online at `http://cpc.cs.qub.ac.uk/summaries/AEUA_v1_3.html`



(a) Original circuit: 29 gates



(b) Optimised circuit: 18 gates

Figure 5.7: Result of quantum circuit optimisation as performed by *OptQC* on a random real unitary matrix. In (b), the dashed vertical lines separate the circuit for each matrix—from left to right, this corresponds to Q, P, U', P^T and Q^T respectively.

5.5.2 Quantum walk operators

One important class of unitary operators are quantum walk operators—in particular, coined quantum walk step operators [41, 83, 119]. The definition of the step operator U is given in equation (2.21).

8-star graph

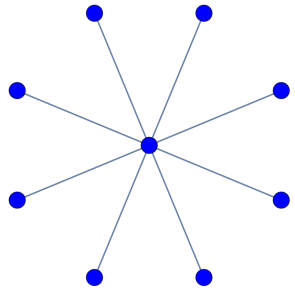


Figure 5.8: The 8-star graph.

The 8-star graph (shown in Figure 5.8) is a graph with 1 centre vertex connected to 8 leaf vertices by undirected edges. Using the Grover coin operator defined in equation (2.20), the resulting step operator on this graph corresponds to a 16-by-16 real unitary matrix, as given below:

$$U = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

By using *OptQC*, the optimisation process gives the following results for the thread which achieves the optimal solution:

- No optimisation: $c_{num}(U, I, I) = 34$ gates
- After selection of an optimised qubit permutation: $c_{num}(U, I, Q) = 27$ gates
- After optimisation procedure for the permutation p: $c_{num}(U, P, Q) = 0 + 1 + 19 + 1 + 0 = 21$ gates

Hence, we achieve a reduction of $\sim$ 38% from the original number of gates. Figure 5.7 shows the optimised circuit obtained for U. Runtime for this calculation is $\sim$ 20 seconds.

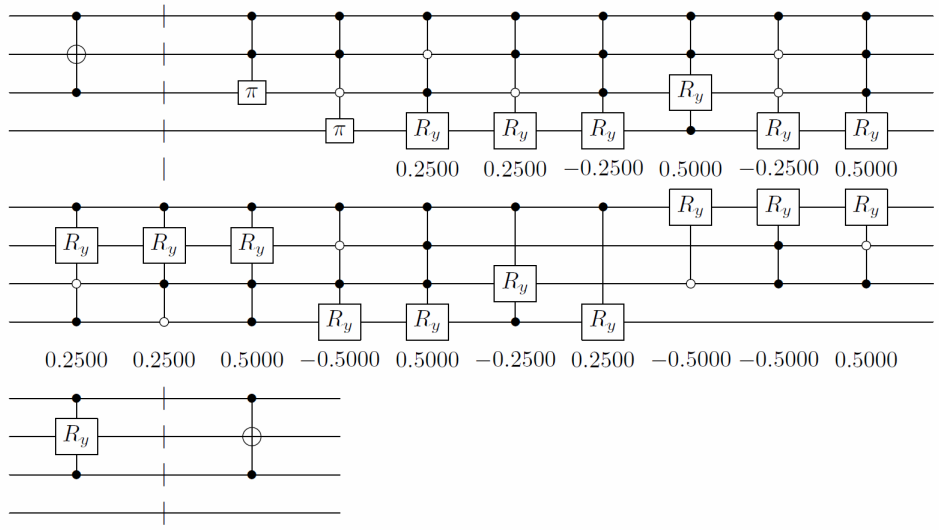


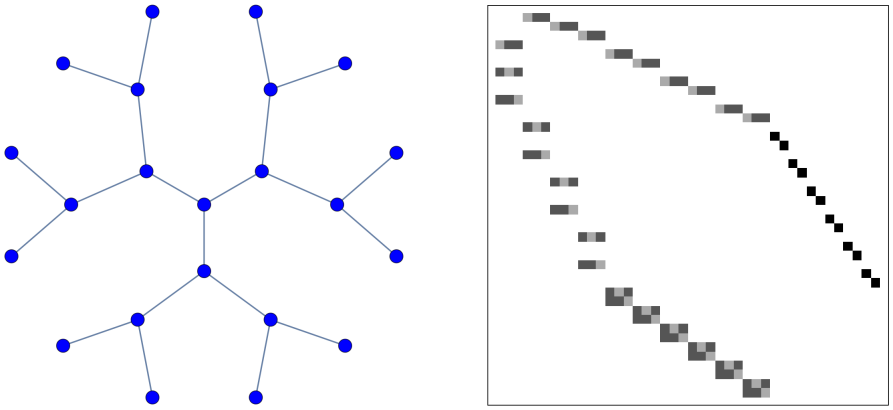
Figure 5.9: Optimised circuit (with 21 gates) for the step operator of the 8-star graph.

3rd generation 3-Cayley tree

The 3rd generation 3-Cayley tree (abbreviated as the 3CT3 graph) is a tree of 3 levels in which all interior nodes have degree 3, as shown in Figure 5.10(a). The corresponding step operator using the Grover coin operator is shown in Figure 5.10(b)—the step operator U is a 42-by-42 real unitary matrix (which is fairly sparse), which, for the purposes of the decomposition, is expanded to a 64-by-64 unitary matrix as per equation (5.10).

By using *OptQC*, the reduction process gives the following results for the thread which achieves the optimal solution:

- No optimisation: $c_{num}(U, I, I) = 996$ gates



(a) 3CT3 graph

(b) Quantum walk operator for the 3CT3 graph

Figure 5.10: The 3CT3 graph and its corresponding step operator using the Grover coin operator. The colours/shades in (b) denote the matrix entries for $-1/3$ (light grey), $2/3$ (dark grey) and 1 (black)—all other matrix entries are 0 (white).

- After selection of an optimised qubit permutation: $c_{num}(U, I, Q) = 2 + 452 + 2 = 456$ gates
- After optimisation procedure for the permutation p: $c_{num}(U, P, Q) = 2 + 17 + 178 + 17 + 2 = 216$ gates

Hence, we achieve a reduction of $\sim$ 78% from the original number of gates. Runtime for this calculation is $\sim$ 7.6 minutes. Figure 5.11 shows the time-series for $c_{num}(U, P, Q)$ during both the qubit permutation selection phase and the main optimisation procedure (separated by a dotted line) to achieve the above result.

5.5.3 Quantum Fourier transform

The quantum Fourier transform is the quantum counterpart of the discrete Fourier transform in classical computing. It is an essential ingredient in several well-known quantum algorithms, such as Shor's factorisation algorithm [120] and the quantum phase estimation algorithm [121]. The matrix representation of the quantum Fourier transform on N dimensions is given by:

$$(\mathrm{QFT})_{jk} = \frac{1}{\sqrt{n}}\omega^{jk}, \quad \text{where } \omega = \exp(2\pi i/n). \tag{5.11}$$

An efficient quantum circuit implementation of the quantum Fourier transform is given in [70], which scales logarithmically as $O(\log(N)^2)$. Such a circuit implementation for $N = 2^6 = 64$ is shown in Figure 5.12.

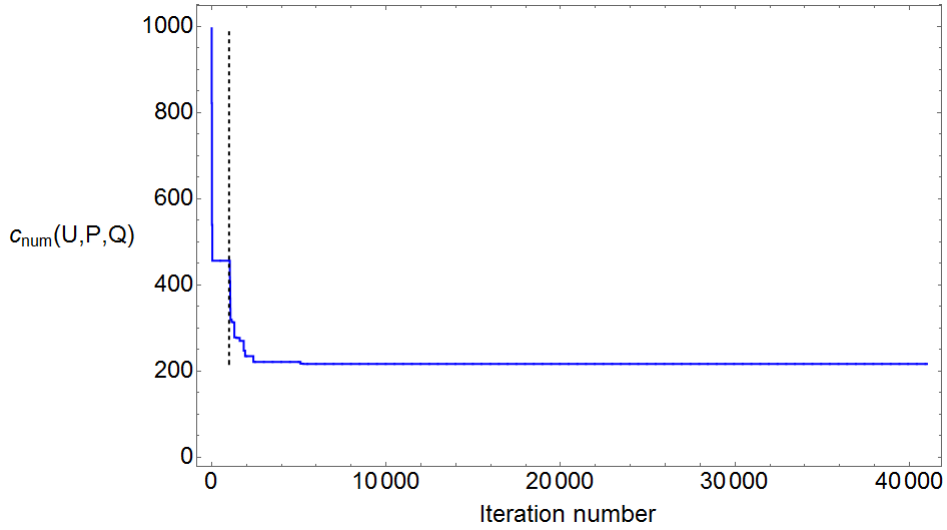


Figure 5.11: Time-series of $c_{num}(U, P, Q)$ during the whole program for the thread which obtains the optimal solution. The original number of gates required by the CSD method to implement U is 996; selection of a qubit permutation reduces this cost to 456 gates, which is used as the starting point for the main optimisation procedure. The main optimisation procedure then monotonically decreases the required number of gates to 216 gates in total. The iterations before the dotted line indicate the qubit permutation selection phase, and the subsequent iterations shows the main optimisation procedure.

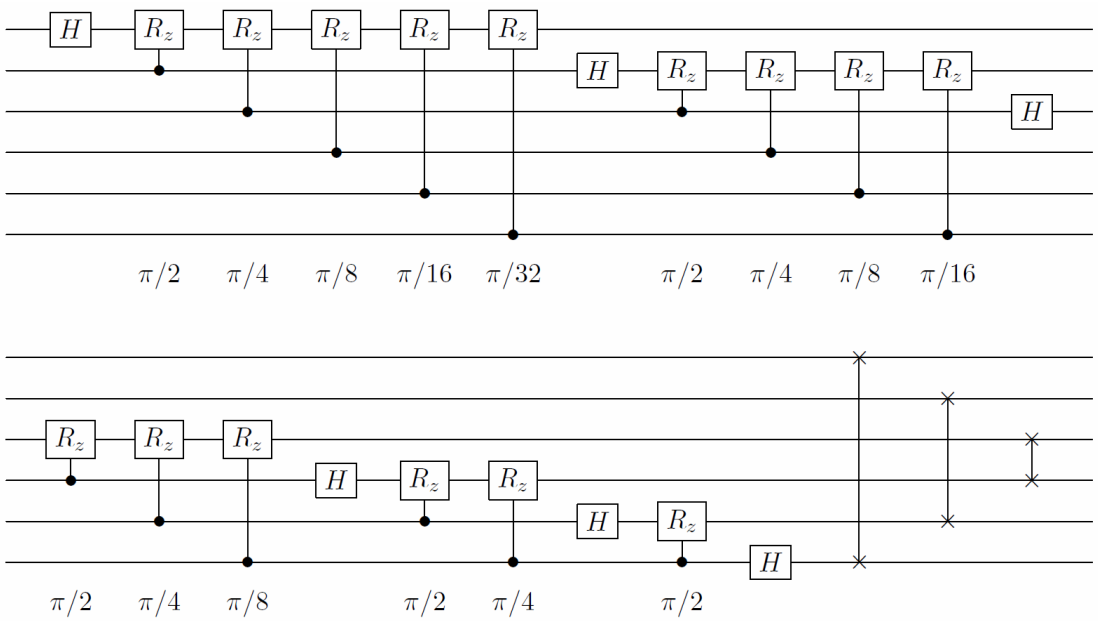


Figure 5.12: Circuit implementation of quantum Fourier transform for $N = 64$.

Now, let us apply *OptQC* to the corresponding 64-by-64 complex unitary operator, given by equation (5.11). The optimisation process gives the following results for the thread achieving an optimal solution:

- No optimisation: $c_{num}(U, I, I) = 4095$ gates
- After selection of an optimised qubit permutation: $c_{num}(U, I, Q) = 4 + 3567 + 4 = 3575$ gates
- After optimisation procedure for the permutation p: $c_{num}(U, P, Q) = 4 + 54 + 3028 + 54 + 4 = 3144$ gates

Hence, we achieve a reduction of $\sim 23.2\%$ from the original number of gates. Runtime of this calculation is ~ 12.7 minutes. Figure 5.13 shows the time-series for $c_{num}(U, P, Q)$ during both the qubit permutation selection phase and the main optimisation procedure (separated by a dotted line) to achieve the above result. Clearly, this result is by far inferior to the quantum circuit of only 24 gates shown in Figure 5.12. Similarly, the *OptQC* package would not be able to provide quantum circuits as efficient as those presented in [39, 41] for the implementation of quantum walks on highly symmetric graphs. This is to be expected, since the CS decomposition is a general technique that decomposes a given unitary into a fixed circuit structure using many conditional gates, with an upper bound of $\mathrm{O}(4^n)$. This algorithm is performed without foreknowledge or explicitly exploiting the structure of the unitary, which would clearly be crucial in achieving the lowest possible number of gates for a given unitary, as exemplified by the above examples. Instead, the *OptQC* package is designed to work for any arbitrary unitary operator for which we do not already have an efficient quantum circuit implementation, for example, quantum walk operators on arbitrarily complex graphs. In such cases, we have demonstrated that the *OptQC* package provides optimised quantum circuits that are far more efficient than the original *Qcompiler*.

5.6 Conclusions and future work

We have developed an optimised quantum compiler, named as *OptQC*, that runs on a parallel architecture to minimise the number of gates in the resulting quantum circuit of a unitary matrix U. This is achieved by finding permutation matrices Q and P

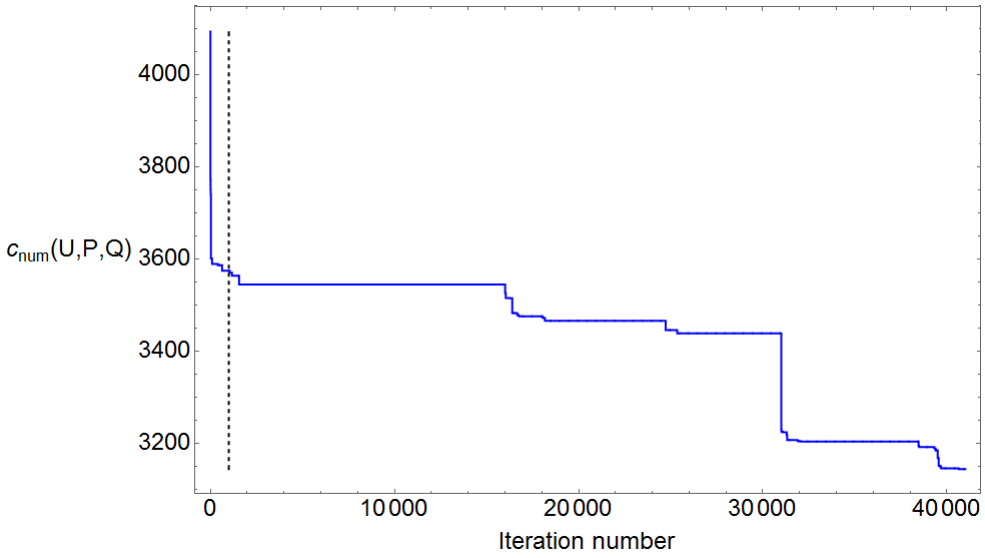


Figure 5.13: Time-series of $c_{num}(U, P, Q)$ during the whole program for the thread which obtains the optimal solution. The original number of gates required by the CSD method to implement U is 4095; selection of a qubit permutation reduces this cost to 3587 gates, which is used as the starting point for the main optimisation procedure. The main optimisation procedure then monotonically decreases the required number of gates to 3144 gates in total. The iterations before the dotted line indicate the qubit permutation selection phase, and the subsequent iterations show the main optimisation procedure.

such that $U = Q^T P^T U' P Q$ requires less total number of gates to be implemented, where the implementation for each matrix is considered separately. Decompositions of unitary matrices are done using the CSD subroutines provided in the LAPACK library [118] and adapted from *Qcompiler* [48]. *OptQC* utilises an optimal selection of qubit permutations Q, a descending random walk procedure to find P, and a combination of similar gates in order to reduce the total number of gates required as much as possible. We find that for many different types of unitary operators, *OptQC* is able to reduce the number of gates required by a significant amount, but its efficacy does vary depending on the unitary matrix given. In particular, this optimisation procedure works well for sparse unitary matrices.

For future work, we hope to look at characterising the optimal solutions reached to see if the matrix U' has some common preferential structure that leads to a reduced cost of implementation using the CSD method. Such information could be used to implement a guided search for the optimal solution, rather than using random adjustments of the permutation matrix. Another potential area for research is using a different set of elementary gates to generate P—for example, using rotation operators such as $R_y(\theta)$ or $R_z(\phi)$ to generate P would allow for more general similarity transforms outside the set of permutation matrices.

Chapter 6

Design principles of efficient quantum circuits

6.1 Introduction

In quantum circuit design for quantum walks (for both analytical and numerical methods), it is clear that there are infinitely many valid quantum circuits that would implement the required quantum walk on a specified graph G. For example, for CTQWs, there are many different eigenbases that can be use to diagonalise the Hamiltonian (see section 3.3). For Szegedy quantum walks, a different choice of the reference state $|\phi_r\rangle$ can be made, giving a different initialisation procedure K_b, a different set of transformations T_i, and hence a different quantum circuit. For numerical methods, a different decomposition scheme, such as the QR-factorisation scheme [70], can be used in place of the cosine-sine decomposition to produce a different decomposition of the same unitary.

However, not all quantum circuits that implement the same quantum walk have the same resource requirements—some of them would satisfy the criteria for efficiency (which we defined in section 2.3), and others would not. It is important, then, to consider if there are any general principles that can be utilised to arrive at an efficient quantum circuit for a given quantum walk—or to determine if an efficient quantum circuit is even possible.

For analytical quantum circuit design of quantum walks, it has been generally found that an efficient quantum circuit is only possible if there is a high degree of symmetry in the underlying graph [39]. As defined by Douglas and Wang [39], sym-

metry here refers to the ability to characterise the structure of the graph by a small number of parameters that increases at most logarithmatically with the number of vertices. As such, this includes sparse graphs with efficiently computable neighbours [57]. This chapter reviews the relevant design principles for each type of quantum walks, including the coined quantum walk, Szegedy quantum walk, and continuous-time quantum walk in sections 6.2, 6.3, and 6.4 respectively, and then draws some conclusions in section 6.5. Since we only interested here in quantum circuit implementations that are efficient, we exclude discussion on quantum compilation, as it is not an efficient method in general.

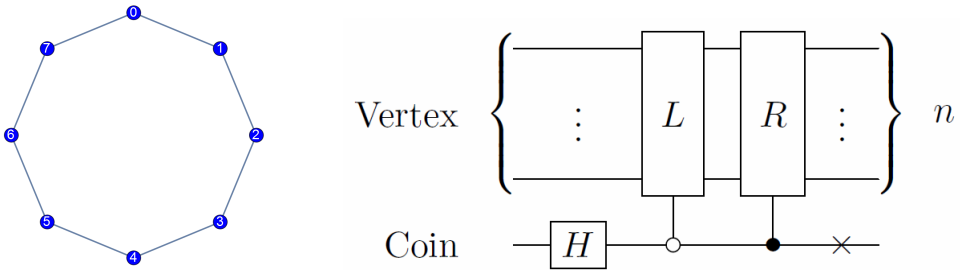
6.2 Coined quantum walk

In a coined quantum walk, the implementation of the step operator given by equation (2.21) is simplified by the fact that the time t is discrete and the evolution is repetitive. As a result, if we can implement a single time-step U in a quantum circuit, the implementation for $U(t)$ can be generated by repeating the same circuit t times. Another property of the coined quantum walk model that is exploited in quantum circuit design is that the single time-step operator U acts locally on the vertex-coin states encoding the graph. In other words, applying U to the vertex-coin states associated with a particular vertex will only propagate the corresponding amplitudes to adjacent vertices, so vertices that are a distance of two or more apart do not affect each other in the single time-step. This means that the *local* structure of the graph is the primary consideration in implementing a coined quantum walk. Mathematically, in the step operator for a coined quantum walk:

$$U = S \cdot C, \tag{6.1}$$

the coin operation C mixes the coin states for each vertex independently and the shifting operator S swaps coin states connected by an edge. The coin operation C is only dependent on the degree of each vertex, i.e. if two graphs with adjacency matrices A and B satisfy the relation $\deg(A)_i = \deg(B)_i \; \forall i$, then the same coin operation C can be used for both graphs (unless some inhomogeneity is intentionally introduced). C does not directly encode information about adjacency relations, and so is often trivial to implement. The shifting operator S expresses the adjacency relations between vertices, but only locally (i.e. to the vertices directly connected

by an edge). Hence, implementation of S is dependent on characterising the local structure of the graph.



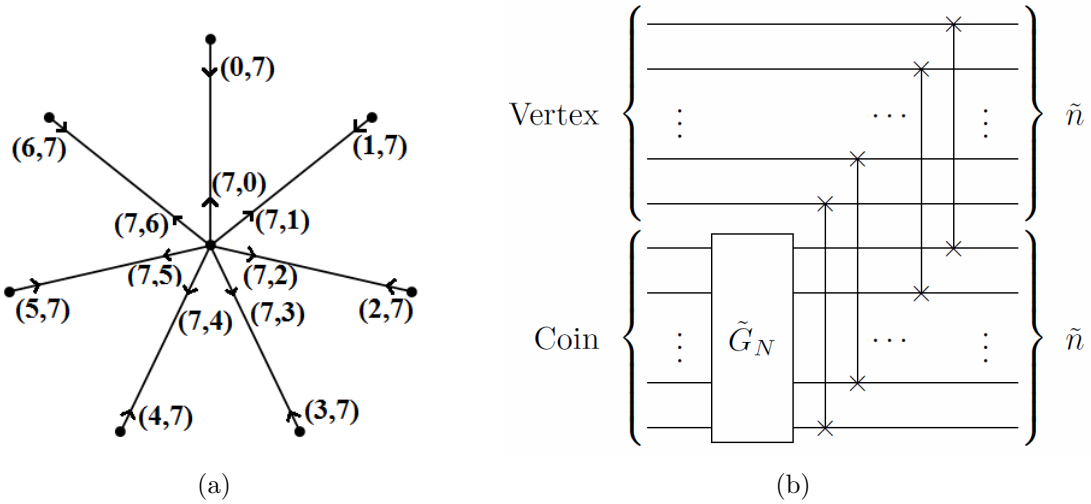
(a) (b)

Figure 6.1: (a) Cycle graph C_N with parameter $N = 8$; (b) its corresponding quantum circuit implementation for the case $N = 2^n$, using the Hadamard coin as the 2-dimensional coin operator for each vertex. The implementation for L and R can be found in Figure 4.3.

As an example, consider the cycle graph C_N [39]. Assuming a lexicographical labeling of the vertices (shown in Figure 6.1(a)), it is clear that every vertex v has two coin states (i.e. has a degree of 2) and is connected by an edge to vertices $\{v \pm 1 \bmod N\}$. Since every vertex has a degree of 2, the coin operator C can be easily implemented, as the same 2-by-2 unitary operator can be used as the coin operation of each vertex. From the above characterisation, we find that the shifting operator S can also be easily implemented, since the same local operation (incrementing or decrementing the vertex number by 1 in a modular fashion) characterises the adjacency relations for each vertex. Figure 6.1(b) shows the quantum circuit for the step operator $U = S \cdot C$ of the graph C_N for the case $N = 2^n$ where $n \in \mathbb{Z}_+$, where the L and R operators are the left-rotation and right-rotation operators defined in section 4.3.

The simplicity of the quantum circuit implementation (as demonstrated in Figure 6.1(b)) is highly dependent on being able to characterise the adjacency relations between vertices in a simple fashion. This, in turn, is dependent on the *labelling* of the graph (including the coin states). For example, labelling the C_N graph shown in Figure 6.1(a) in some random order would necessarily mean that the characterisation of v being adjacent to $\{v \pm 1 \bmod N\}$ no longer holds, thus requiring a completely different quantum circuit implementation for the shifting operator S. As such, for good quantum circuit design for coined quantum walks, one usually picks a labelling that provides a simple characterisation of the adjacency relations[1].

[1]Here, a simple characterisation of the adjacency relations is one that is easily mapped to a



(a) (b)

Figure 6.2: (a) Star graph S_N with parameter $N = 7$ and vertex-coin states labelled as (v_i, c_i); (b) its corresponding quantum circuit implementation, using the N-dimensional Grover coin as the coin operator for the central vertex and identity coin operations for the outer vertices.

Another example where the labelling of the graph is crucial to a simple characterisation of the adjacency relations can be seen in the star graph S_N [41], in which one central vertex is adjacent to N outer vertices. For this graph, we label the vertex-coin states such that the shifting operator acts as $S|v, c\rangle = |c, v\rangle$, i.e. S swaps the vertex and coin labels in order to produce the adjacent edge state. As a result, S can be easily implemented using a sequence of swap gates. An example of this labelling is shown in Figure 6.2(a). In total, there are $2N$ distinct vertex-coin states:

$$|v, c\rangle \in \{|0, N\rangle, \ldots, |N-1, N\rangle, |N, N-1\rangle, \ldots, |N, 0\rangle\}. \tag{6.2}$$

For the coin operator C, we use an identity operation for the outer vertices, and the N-dimensional Grover coin, expanded to the nearest power of two (including at least one extra coin state):

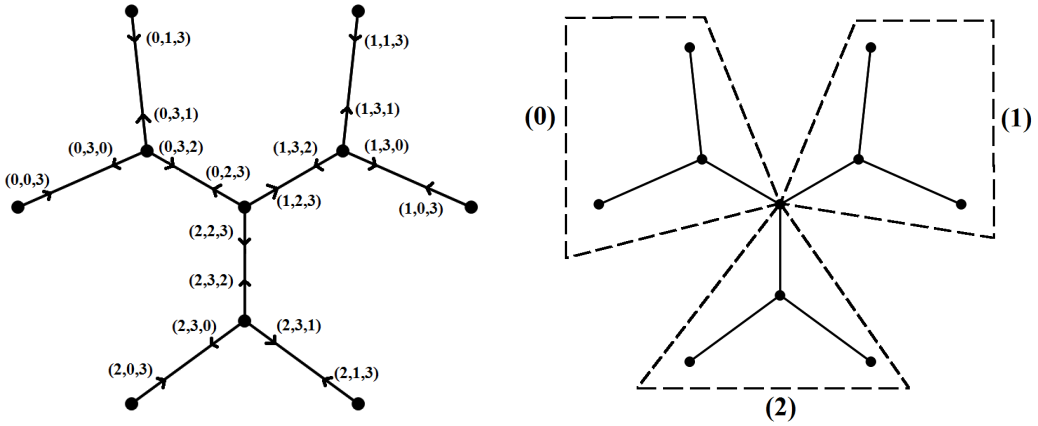
$$(\tilde{G}_N)_{i,j} = \begin{cases} \frac{2}{N} - \delta_{i,j} & : i \leq N, j \leq N \\ \delta_{i,j} & : \text{otherwise}, \end{cases} \tag{6.3}$$

for the central vertex. Note that this operator does not affect states of the form $|v, N\rangle$, since only the first N coin states are mixed. This gives the quantum circuit implementation for the S_N graph shown in Figure 6.2(b), where $\tilde{n} = \lceil \log_2(N+1) \rceil$

quantum circuit, although an algebraically simple characterisation is often one that can be easily mapped to a quantum circuit.

qubits are required for both the vertex and coin registers.

As before, the particular labelling of the graph provided a simple characterisation of the adjacency relations. We note however that this labelling is not minimalistic—that is, this labelling scheme resulted in a quantum circuit implementation that uses more states than minimally required. As noted above, there are only $2N$ distinct vertex-coin states, and so in theory at least $(1 + \lceil \log_2 N \rceil)$ qubits are required. However, using the above labelling scheme results in an implementation that requires $2\tilde{n} = 2\lceil \log_2(N+1) \rceil$ qubits in total, which is strictly larger than the lower bound of $(1 + \lceil \log_2 N \rceil)$. In general, this is the tradeoff from using a convenient labelling scheme—a larger space enables adjacency relations between states to be expressed in a simpler form (leading to fewer gates required), whereas a smaller space does not always have the degrees of freedom necessary for a simple relation to be possible.



(a) Labeled by (t_1, v_1, c) (b) Partitioned by (t_1)

Figure 6.3: The 2nd generation 3-Cayley tree graph, partitioned into three S_3 graphs.

For more complex graphs, characterising the adjacency relations becomes more difficult, so it typically requires more degrees of freedom (i.e. a larger space) to obtain a characterisation that is simple. For example, consider the 2nd generation 3-Cayley tree graph shown in Figures 6.3(a) and 6.3(b). In order to characterise the graph, we recognise that the graph can be formed by connecting three S_3 graphs together (as shown in Figure 6.3(b)), so a similar 2-parameter characterisation using v_1 and c can be used, with an additional parameter t_1 to distinguish between the three S_3 graphs. In this way, 3 parameters can be used to characterise the 2nd generation 3-Cayley tree graph, giving the quantum circuit implementation shown in Figure 6.4. One can easily extend this 3-parameter characterisation to any 2nd generation d-Cayley

tree graph, which is formed by connecting d S_d graphs together. In fact, it has been shown in [41] that any n^{th} generation d-Cayley tree can be characterised in similar fashion, using an $(n+1)$-parameter characterisation, using a total of $\lceil (n+1)\log_2 d \rceil$ qubits and $O((n\,\log_2 d)^2)$ one and two qubit gates to implement, which is efficient.

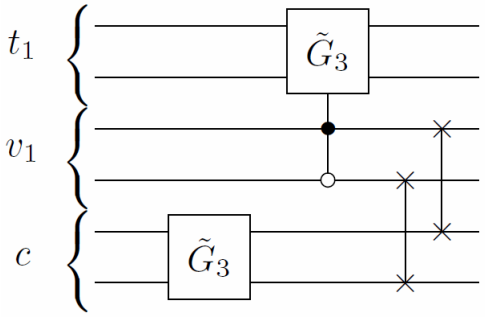


Figure 6.4: Quantum circuit implementation for the 2^{nd} generation 3-Cayley tree graph.

In summary, for a coined quantum walk, the coin operator C is often easy to implement (due to its action being confined to the coin states of each vertex), as long as the choice of the coin operator is consistent. For the shifting operator S, the implementation can (in most cases) be simplified by choosing a labelling scheme (i.e. a parameterisation) for the graph that gives a simple characterisation of the adjacency relations, but this often comes at the tradeoff of requiring a larger state space to work with. This characterisation is usually only possible for graphs with some form of geometrical or combinatoral symmetry.

6.3 Szegedy quantum walk

For the Szegedy quantum walk, some of the design principles from the coined quantum walk carry over, since they are both discrete-time quantum walk models and are, to some extent, equivalent [54]. In particular, the implementation for $U(t)$ can be generated by repeating t times the circuit for the step operator U_{walk}. Since a single application of U_{walk} only propagates the corresponding amplitude to adjacent vertices, paths of length 2 or more from a vertex need not be considered in one time-step. Of course, some differences exist from the coined quantum walk model—for example, Szegedy walks on directed and weighted graphs can be defined, whereas quantum walks on such graphs cannot, in general (see [122] for some exceptions), be defined consistently in the coined quantum walk model. In the Szegedy quantum walk model, recall that the step operator is given by:

$$U_{\text{walk}} = \mathcal{S}(2\Pi - I) = \mathcal{S}\mathcal{R}. \tag{6.4}$$

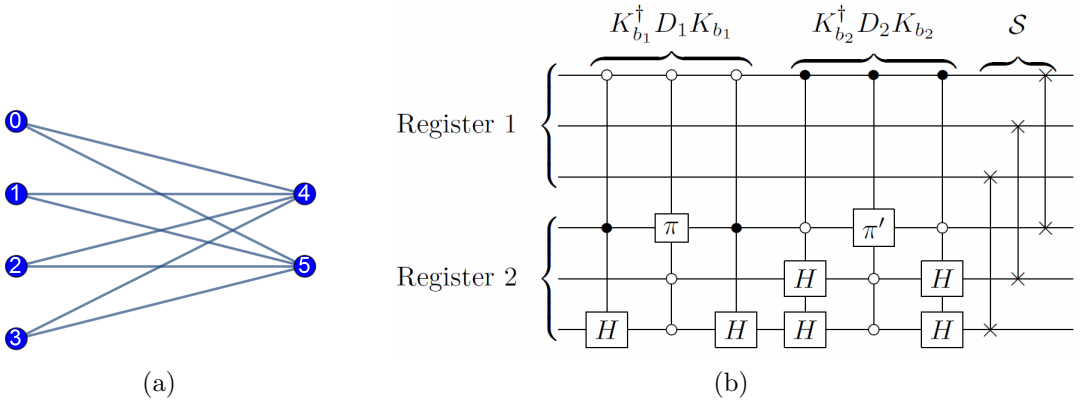


Figure 6.5: (a) Complete bipartite graph $K_{4,2}$; (b) its corresponding quantum circuit implementation.

From this, a partial equivalence between the step operator for the Szegedy quantum walk and the coined quantum walk can be established. A more detailed study of equivalence between the two quantum walk models can be found in [54]. The step operator for the Szegedy quantum walk on any Markov chain with N states can be treated as a coined quantum walk on the complete graph K_N with self-loops, with each vertex v_i having the coin operator $C_{v_i} = 2|\phi_i\rangle\langle\phi_i| - I_N$. Hence, a change in the transition matrix for the Szegedy quantum walk simply corresponds to changing the coin operators of the corresponding coined quantum walk on the K_N graph with self-loops. Mathematically, $\mathcal{S}$ and $\mathcal{R}$ correspond to the shifting and coin operator of a coined quantum walk, respectively.

However, in the Szegedy quantum walk model, the primary difficulty lies in the implementation of the reflection operator $\mathcal{R}$, since $\mathcal{R}$ characterises the adjacency relations of the graph in the Szegedy quantum walk. The swap operator $\mathcal{S}$ can be easily implemented using a sequence of swap gates, as noted in section 4.2. However, instead of modifying the labelling of the graph (as we did for the coined quantum walk) in order to obtain a simple characterisation of the graph, we can instead use the results in chapter 4 (specifically Theorem 4.2 in section 4.2) as a framework for characterising the vertices of the graph. We partition the vertex set $\{0, \ldots, N-1\}$ into subsets such that, within each subset, the corresponding columns of the transition matrix are simply related in some fashion. That is, the transformations $T_{x,y}$ and preparation routines K_{b_x} can be done efficiently. In practice, finding efficient preparation routines K_{b_x} is usually much simpler than implementing the transfor-

mations $T_{x,y}$. For example, the complete bipartite graph $K_{4,2}$ shown in Figure 6.5(a) has the following transition matrix:

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 1/2 & 1/2 & 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 1/2 & 1/2 & 0 & 0 \end{pmatrix}. \tag{6.5}$$

From the transition matrix, it can be readily seen that the first four columns are simply related by an identity transformation, and so are the last two columns. Hence the partitioning $Z = Z_1 \cup Z_2$ where $Z_1 = \{0, 1, 2, 3\}$ and $Z_2 = \{4, 5\}$ is a natural choice. Since it satisfies the other sufficient conditions of Theorem 4.2 (the preparation routines K_{b_x} and controlled unitary operations can be done efficiently), it follows that the step operator U_{walk} can be implemented efficiently. In other words, an extension of this graph to higher dimensions can be implemented using $O(\text{poly}(\log(N)))$ one and two qubit gates, as shown in Figure 6.5(b).

Modifications to the transition matrix that occur within subsets may allow for the same partitioning to be used, with a change in the transformation and/or the preparation routines. For example, consider the following modified transition matrix for the $K_{4,2}$ graph:

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 1/4 \\ 1-\alpha & \alpha & 1-\alpha & \alpha & 0 & 0 \\ \alpha & 1-\alpha & \alpha & 1-\alpha & 0 & 0 \end{pmatrix}, \tag{6.6}$$

where $\alpha \in [0, 1]$ is a real parameter. For this transition matrix, the same partitioning $Z = Z_1 \cup Z_2$ can be used, with a modification to the transformation operators for Z_1 as:

$$T_{1,y} = \begin{cases} \begin{pmatrix} I_4 & 0 \\ 0 & \sigma_x \end{pmatrix} & : y \in \{1, 3\} \\ I_6 & : \text{otherwise}, \end{cases} \tag{6.7}$$

which is essentially a CNOT gate that acts when $y \in \{1, 3\}$. The preparation routine K_{b_1} can be modified easily to prepare the reference state $|\phi_0\rangle = \{0, 0, 0, 0, 1-\alpha, \alpha\}^T$ by using a rotation gate. Putting these modifications together gives us the quantum circuit for U_{walk} shown in Figure 6.6.

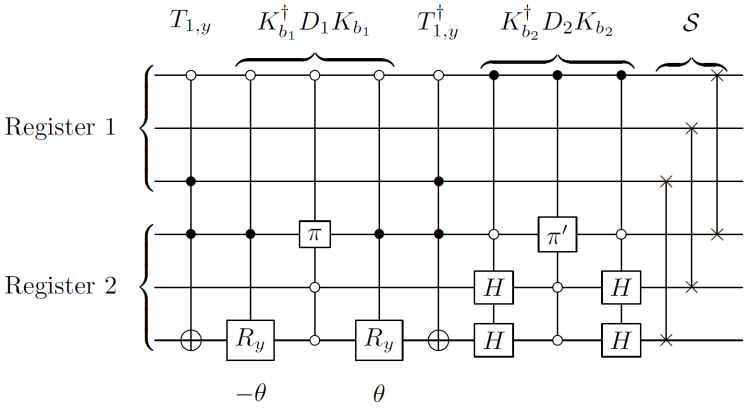


Figure 6.6: Quantum circuit implementing U_{walk} for the modified transition matrix in equation (6.6). The rotation angle for K_{b_1} is given by $\cos(\theta) = \sqrt{1-\alpha}$.

In summary, for a Szegedy quantum walk, the swap operator $\mathcal{S}$ is trivial to implement, so the difficulty is in implementing the reflection operator $\mathcal{R}$. In order to simplify the implementation of $\mathcal{R}$, a partition of the graph that results in efficiently implementable transformations $T_{x,y}$ within each subset is required. The transformations $T_{x,y}$ are typically efficient when, within each subset, the corresponding columns are simply related in some fashion.

6.4 Continuous-time quantum walk

In contrast to the DTQW models (the coined quantum walk model and the Szegedy quantum walk model), we find that quantum circuit design for CTQWs are substantially more difficult, for several reasons. First, the time-evolution operator $U(t)$ for the CTQW is defined as a continuous function of $t \in \mathbb{R}$, requiring a time-dependent quantum circuit implementation. Second, the CTQW does not act locally on vertices—any two even distantly connected vertices on a graph will propagate amplitudes to each other in a CTQW. Consequently, the global structure of a graph must be taken into account in designing quantum circuits to implement CTQWs. Given a Hamiltonian H which satisfies $H = H^\dagger$, the corresponding time-evolution operator for the CTQW is given by:

$$U(t) = \exp(-itH), \tag{6.8}$$

for which we take $H = \gamma A$, where A is the adjacency matrix of the graph. Under the definition of efficiency outlined in section 2.3, there are necessarily a limited number of types of graphs that are possible to simulate efficiently, since it corresponds to classes of Hamiltonians whose time-evolution can be fast-forwarded efficiently. One method of achieving quantum circuits with time-independent complexity is by diagonalising the Hamiltonian using its eigenbasis Q, i.e.

$$U(t) = Q^{\dagger}\exp(-it\Lambda)Q, \tag{6.9}$$

as outlined in section 3.2. While any Hamiltonian can be diagonalised in this fashion to obtain a quantum circuit with time-independent complexity, the resulting circuit may not be efficient with respect to the parameter N. So far, only a few types of graphs that are highly symmetric have been found to be efficiently diagonalisable. In the literature, the complete graph, complete bipartite graph, star graph, and the Winnie Li graph on odd dimensions have been found to be diagonalisable [65]. Our results in chapter 3 enable simulation of some classes of circulant graphs via diagonalisation using the quantum Fourier transform matrix. We can also generate new classes of graphs from this list of efficiently implementable graphs by composing them in some fashion, as outlined in section 3.4.

One important degree of freedom in the diagonalisation method is the choice of eigenbasis when the eigenvalues are degenerate. For example, in section 3.3, we noted that the complete graph K_N for the case $N = 2^n$ where $n \in \mathbb{Z}_+$ could be diagonalised using two different eigenbases: the quantum Fourier transform or the Hadamard basis $H^{\otimes n}$, as shown in Figure 3.2(b) and 3.3 respectively. This is because the eigenvalue spectrum for the K_N graph is $\{(N-1)^1, (-1)^{N-1}\}$, i.e. the eigenvalue -1 is degenerate with algebraic multiplicity $N-1$. Since the Hamiltonian is Hermitian, this implies that the Hamiltonian is never defective [123], i.e. the geometric and algebraic multiplicities of all eigenvalues are always equal—implying that any orthogonal basis of $N-1$ vectors belonging to the eigenspace (i.e. the kernel $\ker(H+I)$) can be chosen as the corresponding eigenvectors. In more general terms, for an eigenvalue λ of the Hamiltonian with algebraic multiplicity m_λ, any orthogonal basis of m_λ vectors for the eigenspace $\ker(H-\lambda I)$ can be chosen as the eigenvectors for the eigenvalue λ. As such, degeneracy of the eigenspectrum can be exploited to select an eigenbasis that can be more easily implemented in a quantum circuit, as demonstrated by the K_N example. From this, conjecture that

if the eigenspectrum of the Hamiltonian has more degeneracy (i.e. high algebraic multiplicities for eigenvalues), then it is more likely that the Hamiltonian can be efficiently diagonalised.

Another example of the application of this conjecture is to the complete bipartite graph K_{N_1,N_2}. This graph has the eigenvalue spectrum $\{(\sqrt{N_1N_2})^1, (-\sqrt{N_1N_2})^1, 0^{N_1+N_2-2}\}$, which is highly degenerate since the eigenvalue 0 has algebraic multiplicity N_1+N_2-2. As such, according to the conjecture, it is likely that the Hamiltonian corresponding to the graph K_{N_1,N_2} can be efficiently implemented (its implementation was provided in section 3.4.1). The star graph S_N is equivalent to the $K_{N,1}$ graph, and so is also highly degenerate.

Of course, diagonalisation as a method has its drawbacks. In particular, it requires full knowledge of the eigenspectrum of H. Hence, it is impractical to use diagonalisation without having this knowledge, since precomputation of the eigenspectrum scales as $O(N^3)$ typically [124], which scales exponentially as N scales exponentially. Rather, it is best used for graphs for which we already have this information analytically. Another drawback of the diagonalisation method is that small alterations to a graph are often non-trivial to implement, because the eigenspectrum (which is encoded by the matrix of column eigenvectors Q and the eigenvalue matrix Λ) can change significantly. An example of this is the vertex marking operation, which we discussed briefly in section 3.3. Recall that the marked Hamiltonian is defined as $H' = H + H_m$ where $H_m = |v_m\rangle\langle v_m|$ represents the marking at vertex v_m. Even though this operation only modifies one diagonal element of the Hamiltonian, the eigenspectrum changes completely (since it breaks the circulant symmetry of the graph), requiring a completely different (and more complex) quantum circuit implementation, as shown in Figure 6.7.

In summary, for a CTQW, efficient quantum circuit implementations for $U(t)$ can only be done for some graphs with some particular types of symmetry. We conjecture that this type of symmetry corresponds to graphs which have a highly degenerate eigenspectrum. This can be used to perform efficient diagonalisation of the Hamiltonian and hence simulation of $U(t)$ according to equation (6.9). However, these symmetries are often quite restrictive, hence even minor changes to the Hamiltonian that break the symmetry of the graph would require non-trivial alterations to the quantum circuit (or a completely different one) to simulate the modified Hamiltonian.

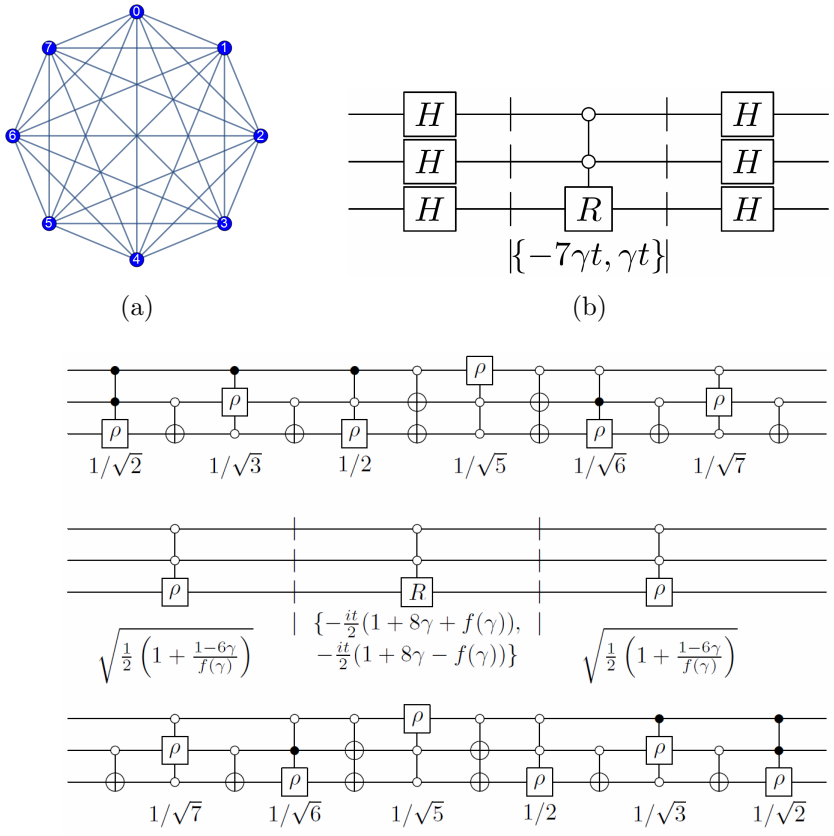


Figure 6.7: Quantum circuit implementation of $U(t)$ for the K_8 graph (shown in (a)) with (b) no marking; (c) vertex 0 marked, where $f(\gamma) = \sqrt{1+4\gamma(16\gamma-3)}$.

6.5 Conclusions

To summarise, we have reviewed the relevant design principles for the coined quantum walk, Szegedy quantum walk, and continuous-time quantum walk. We list them as follows:

- Coined quantum walk: Pick a labelling scheme that provides a simple characterisation of the adjacency relations. This is generally only possible by utilising a larger space instead of a minimalistic space, i.e. using more qubits than the minimum required.
- Szegedy quantum walk: Partition the vertex set into subsets in which the corresponding columns of the transition matrix are simply related in some fashion.
- Continuous-time quantum walk: Use diagonalisation of the Hamiltonian and exploit degeneracy of the eigenspectrum to choose an eigenbasis that is efficiently implementable.

When considering the different types of quantum walks and their implementation, a natural question to ask is whether the implementation of one type of quantum walks is more difficult than another in some sense. To provide some insight to this question, we would propose the following ranking, in descending order of difficulty, for the three types of quantum walks:

$$\text{CTQW} >> \text{Szegedy quantum walk} > \text{Coined quantum walk}. \tag{6.10}$$

Our rationale behind this ranking is dependent on two main factors: generality and robustness. Generality refers to the range of graphs (undirected, directed, or weighted in general) for which the quantum walk can be defined—this can be thought of as the allowed number of free parameters in the particular quantum walk framework. Robustness refers to how easy it is to make a modification (e.g. addition/deletion of vertices or edges, marking a vertex, etc.) to the underlying graph by making a minor change to the corresponding quantum circuit. Hence, in our ranking scheme, a quantum walk framework that is more general (requiring a quantum circuit implementation that accounts for more free variables) and less robust (requiring large alterations in the quantum circuit implementation when a minor

change is introduced to the graph) is therefore ranked to be more difficult than its counterparts.

In terms of generality, the Szegedy quantum walk is the most general framework out of the three formalisms, since it can be applied to directed and weighted graphs in general, unlike the coined quantum walk[2] and the CTQW[3]. In terms of robustness, we find that compared to the Szegedy quantum walk and the coined quantum walk, the CTQW is, by far, less robust than either model, since as pointed out in section 6.4, even minor alterations such as vertex marking can drastically change the eigenspectrum (and hence the quantum circuit implementation) of the graph. In comparison, such alterations are much easier to implement in the coined quantum walk model (see [39, 41] for examples on vertex marking) and the Szegedy walk model (vertex marking would correspond to a replacement of a single column in the transition matrix). Hence, we rank the CTQW model as the most difficult to implement (due to its lack of robustness), followed by the Szegedy walk model, as it is more general (but about the same in robustness) than the coined quantum walk model.

[2]For a fixed undirected graph, the coined quantum walk does have more freedom in its choice of coin operators, since any unitary matrix (including those with complex elements, which is not allowed by the reflection operator of the Szegedy walk) of appropriate dimensions is allowed—the reflection operator (i.e. the coin operator equivalent) of the Szegedy quantum walk is defined directly with respect to the graph, and so is inflexible in that sense.

[3]However, the more general form of CTQW (Hamiltonian simulation) can be applied to Hermitian adjacency matrices which allows for complex values (see papers on the topic of chiral quantum walks [125]), which the Szegedy quantum walk does not allow for, so it is slightly more general in a different sense.

Chapter 7

Conclusion

In this thesis, the crucial step from theory to implementation for quantum walks is developed using the quantum circuit model of computation, enabling the implementation of quantum walk-based algorithms on a quantum computer. We study the application of analytical and numerical methods to this task while looking at different types of quantum walks separately, since the construction of the unitary evolution operator for each type is markedly different.

For the continuous-time quantum walk (CTQW), we utilise diagonalisation of the Hamiltonian in order to generate quantum circuit implementations that have time-independent complexity. An efficient quantum circuit for CTQWs, then, requires that the eigenbasis (matrix of column eigenvectors Q and diagonal matrix of eigenvalues $e^{-it\Lambda}$) be efficiently implementable. We find that the class of circulant graphs (for example, cycle graphs and complete graphs) can be efficiently diagonalised using the quantum Fourier transform. We have also shown that the quantum circuit implementation for some types of composite graphs (that is, graphs generated by combining two other graphs in some prescribed fashion) can be constructed, specifically for commuting graphs and the Cartesian product of graphs. It is the aim of future work to identify relevant physical systems that exhibit these kinds of symmetry (e.g. chemical molecules with circulant symmetry) and apply these results to simulate these systems efficiently.

For the Szegedy quantum walk, we have presented a scheme for constructing the quantum circuit implementation of a given Markov chain. We identified explicitly the requirements for efficiency under this scheme, and then generalised these requirements for the more general case of allowing for a partitioning of the vertex

set. This formalism was then applied to a wide variety of graph types: firstly, the class of graphs generated by a cyclic permutation (a generalisation of the class of circulant graphs); next, the class of complete bipartite graphs (which are necessarily non-sparse); then, the class of graphs formed by a tensor product; and lastly, some types of weighted interdependent networks. Lastly, we demonstrated that the Szegedy quantum walk required to implement the quantum Pagerank algorithm [9, 10] can be implemented for some types of graphs using our scheme, for which we provide explicit quantum circuit implementations. Given the generality of the formalism presented in section 4.2, there are almost certainly more classes of Markov chains which can be simulated efficiently than those found thus far—finding these classes of Markov chains is the focus of future work.

We then investigated the application and optimisation of the cosine-sine decomposition, which is a numerical technique that can be applied recursively to generate a quantum circuit implementation of any given unitary matrix. We pose the optimisation of the quantum circuit as a minimisation problem with permutation matrices applied to the unitary matrix as the variable. By using this and parallelising the implementation using MPI, we build on the existing software package *Qcompiler* [48] to develop the optimised quantum compiler *OptQC*, which we apply to a number of different unitary matrices, including to the step operator of the coined quantum walk. For future work, a characterisation of the structure of optimal solutions can shed some light on whether there exists some common preferential structure for the cosine-sine decomposition—this can be used to implement a guided search for the optimal solution, which speeds up the search for the global minimum.

Lastly, we reviewed efficient quantum circuit implementations of the three quantum walk models (including the coined quantum walk model), and derive design principles for each, which conveys some sense of intuition on how such implementations are developed. It is left as an open question whether other degrees of freedom apart from eigenvalue degeneracy can be exploited in designing efficient quantum circuits for CTQWs.

One area of interest in quantum circuit design for quantum walks that is not directly discussed in this thesis (with the exception of section 4.5) is multi-particle quantum walks, with or without interactions. One important application of multi-particle quantum walks is to increase the distinguishing power in a quantum-walk based graph isomorphism algorithm [55] over the single-particle version of the algo-

rithm. For CTQWs, a multi-particle CTQW without interactions corresponds to a Cartesian product of the Hamiltonian with itself (as in section 3.4.2). Interactions can be introduced by modifying the Hamiltonian at sites where there is more than one particle at a single vertex [88]. From section 3.4.2, it follows easily that the multi-particle case without interactions can be efficiently implemented if the single-particle case can be implemented efficiently. However, it remains to be seen if this can be modified for the multi-particle case, since typically introducing any additional terms to the Hamiltonian changes the eigenspectrum. For the Szegedy (and coined) quantum walk, similar considerations apply, in that if the single-particle case can be implemented efficiently, the multi-particle case without interaction follows easily. However, it also remains to be seen that the case with interactions can be implemented efficiently by modifying the implementation for the case without interactions.

References

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

[2] H. Buhrman et al. "Quantum Fingerprinting". en. *Physical Review Letters* 87.16 (Sept. 2001).

[3] D. Gottesman and I. Chuang. "Quantum digital signatures". *arXiv preprint quant-ph/0105032* (2001).

[4] P. P. Rohde, J. F. Fitzsimons, and A. Gilchrist. "Quantum Walks with Encrypted Data". *Physical Review Letters* 109.15 (Oct. 2012), p. 150501.

[5] A. Harrow, A. Hassidim, and S. Lloyd. "Quantum Algorithm for Linear Systems of Equations". en. *Physical Review Letters* 103.15 (Oct. 2009).

[6] S. Aaronson. "Read the fine print". en. *Nature Physics* 11.4 (Apr. 2015), pp. 291–293.

[7] A. M. Childs, R. Kothari, and R. D. Somma. "Quantum linear systems algorithm with exponentially improved dependence on precision". *arXiv:1511.02306 [quant-ph]* (Nov. 2015). arXiv: 1511.02306.

[8] B. L. Douglas and J. B. Wang. "A classical approach to the graph isomorphism problem using quantum walks". *Journal of Physics A: Mathematical and Theoretical* 41.7 (Feb. 2008), p. 075303.

[9] G. D. Paparo and M. A. Martin-Delgado. "Google in a Quantum Network". *Scientific Reports* 2 (June 2012), p. 444.

[10] G. D. Paparo et al. "Quantum Google in a Complex Network". *Scientific Reports* 3 (Oct. 2013).

[11] S. Lloyd. "Universal Quantum Simulators". *Science* 273.5278 (Aug. 1996), pp. 1073–1078.

[12] C. Zalka. "Simulating quantum systems on a quantum computer". *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 454.1969 (Jan. 1998), pp. 313–322.

[13] I. Kassal et al. "Simulating Chemistry Using Quantum Computers". *Annual Review of Physical Chemistry* 62.1 (2011), pp. 185–207.

[14] I. Georgescu, S. Ashhab, and F. Nori. "Quantum simulation". *Reviews of Modern Physics* 86.1 (Mar. 2014), pp. 153–185.

[15] L. Lamata et al. "Efficient quantum simulation of fermionic and bosonic models in trapped ions". en. *EPJ Quantum Technology* 1.1 (Dec. 2014).

[16] Y. Salath et al. "Digital Quantum Simulation of Spin Models with Circuit Quantum Electrodynamics". *Physical Review X* 5.2 (June 2015), p. 021027.

[17] D. Deutsch. "Quantum Computational Networks". en. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 425.1868 (Sept. 1989), pp. 73–90.

[18] A. C.-C. Yao. "Quantum circuit complexity". Nov. 1993, pp. 352–361.

[19] A. Barenco et al. "Elementary gates for quantum computation". *Physical Review A* 52.5 (Nov. 1995), pp. 3457–3467.

[20] D. P. DiVincenzo. "Quantum gates and circuits". *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 261–276.

[21] E. Farhi et al. "Quantum Computation by Adiabatic Evolution". *arXiv:quant-ph/0001106* (Jan. 2000). arXiv: quant-ph/0001106.

[22] D. Aharonov and A. Ta-Shma. "Adiabatic quantum state generation and statistical zero knowledge". *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM, 2003, pp. 20–29.

[23] D. Aharonov et al. "Adiabatic quantum computation is equivalent to standard quantum computation". *45th Annual IEEE Symposium on Foundations of Computer Science, 2004. Proceedings*. Oct. 2004, pp. 42–51.

[24] M. Freedman et al. "Topological quantum computation". *Bulletin of the American Mathematical Society* 40.1 (2003), pp. 31–38.

[25] S. Das Sarma, M. Freedman, and C. Nayak. "Topologically Protected Qubits from a Possible Non-Abelian Fractional Quantum Hall State". *Physical Review Letters* 94.16 (Apr. 2005), p. 166802.

[26] C. Nayak et al. "Non-Abelian anyons and topological quantum computation". *Reviews of Modern Physics* 80.3 (Sept. 2008), pp. 1083–1159.

[27] D. Deutsch. "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer". en. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 400.1818 (July 1985), pp. 97–117.

[28] E. Bernstein and U. Vazirani. "Quantum Complexity Theory". *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*. STOC '93. New York, NY, USA: ACM, 1993, pp. 11–20.

[29] Y. Aharonov, L. Davidovich, and N. Zagury. "Quantum random walks". *Physical Review A* 48.2 (Aug. 1993), pp. 1687–1690.

[30] E. Farhi and S. Gutmann. "Quantum computation and decision trees". *Physical Review A* 58.2 (Aug. 1998), pp. 915–928.

[31] D. Aharonov et al. "Quantum walks on graphs". *Proceedings of the thirty-third annual ACM symposium on Theory of computing*. ACM, 2001, pp. 50–59.

[32] A. M. Childs et al. "Exponential Algorithmic Speedup by a Quantum Walk". *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*. STOC '03. New York, NY, USA: ACM, 2003, pp. 59–68.

[33] M. Szegedy. "Quantum speed-up of Markov chain based algorithms". *45th Annual IEEE Symposium on Foundations of Computer Science, 2004. Proceedings*. Oct. 2004, pp. 32–41.

[34] M. Szegedy. "Spectra of Quantized Walks and a $\sqrt{\delta\epsilon}$ rule". *arXiv:quant-ph/0401053* (Jan. 2004). arXiv: quant-ph/0401053.

[35] S. P. Jordan. "Quantum computation beyond the circuit model". PhD thesis. Massachusetts Institute of Technology, 2008.

[36] A. M. Childs and J. Goldstone. "Spatial search by quantum walk". *Physical Review A* 70.2 (Aug. 2004), p. 022314.

[37] A. M. Childs, E. Farhi, and S. Gutmann. "An Example of the Difference Between Quantum and Classical Random Walks". *Quantum Information Processing* 1.1-2 (Apr. 2002), pp. 35–43.

[38] A. M. Childs. "Quantum information processing in continuous time". PhD thesis. Massachusetts Institute of Technology, 2004.

[39] B. Douglas and J. Wang. "Efficient quantum circuit implementation of quantum walks". *Physical Review A* 79.5 (May 2009), p. 052335.

[40] C.-F. Chiang, D. Nagaj, and P. Wocjan. "Efficient Circuits for Quantum Walks". *Quantum Info. Comput.* 10.5 (May 2010), pp. 420–434.

[41] T. Loke and J. B. Wang. "Efficient circuit implementation of quantum walks on non-degree-regular graphs". *Physical Review A* 86.4 (Oct. 2012), p. 042338.

[42] T. Loke, J. B. Wang, and Y. H. Chen. "OptQC: An optimized parallel quantum compiler". *Computer Physics Communications* 185.12 (Dec. 2014), pp. 3307–3316.

[43] X. Qiang et al. "Efficient quantum walk on a quantum processor". *Nature Communications* 7 (May 2016), p. 11511.

[44] T. Loke and J. B. Wang. "Efficient quantum circuits for continuous-time quantum walks on composite graphs". en. *Journal of Physics A: Mathematical and Theoretical* 50.5 (2017), p. 055303.

[45] G. Cybenko. "Reducing quantum computations to elementary unitary operations". *Computing in Science Engineering* 3.2 (Mar. 2001), pp. 27–32.

[46] C. M. Dawson and M. A. Nielsen. "The Solovay-Kitaev algorithm". *Quantum Information and Computation* 6.1 (2005), pp. 81–95.

[47] M. Amy et al. "A Meet-in-the-Middle Algorithm for Fast Synthesis of Depth-Optimal Quantum Circuits". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32.6 (June 2013), pp. 818–830.

[48] Y. G. Chen and J. B. Wang. "Qcompiler: Quantum compilation with the CSD method". *Computer Physics Communications* 184.3 (Mar. 2013), pp. 853–865.

[49] C.-C. Lin. "Reversible and Quantum Circuit Synthesis". PhD thesis. Princeton University, July 2014.

[50] O. Di Matteo. "Parallelizing quantum circuit synthesis". en. PhD thesis. Apr. 2015.

[51] V. Kliuchnikov, D. Maslov, and M. Mosca. "Practical approximation of single-qubit unitaries by single-qubit quantum Clifford and T circuits". *IEEE Transactions on Computers* (2015). arXiv: 1212.6964, pp. 1–1.

[52] M. Hillery, J. Bergou, and E. Feldman. "Quantum walks based on an interferometric analogy". *Physical Review A* 68.3 (Sept. 2003), p. 032314.

[53] A. M. Childs. “Quantum algorithms: Lecture 14 - Discrete-time quantum walk”. University of Waterloo, 2008.

[54] R. Portugal. “Establishing the equivalence between Szegedys and coined quantum walks using the staggered model”. en. *Quantum Information Processing* 15.4 (Jan. 2016), pp. 1387–1409.

[55] J. K. Gamble et al. “Two-particle quantum walks applied to the graph isomorphism problem”. *Physical Review A* 81.5 (May 2010), p. 052313.

[56] K. Rudinger et al. “Noninteracting multiparticle quantum random walks applied to the graph isomorphism problem for strongly regular graphs”. *Physical Review A* 86.2 (Aug. 2012), p. 022334.

[57] D. W. Berry et al. “Efficient Quantum Algorithms for Simulating Sparse Hamiltonians”. *Communications in Mathematical Physics* 270.2 (Mar. 2007), pp. 359–371.

[58] R. Kothari. “Efficient simulation of Hamiltonians”. PhD thesis. University of Waterloo, 2010.

[59] A. M. Childs and R. Kothari. “Simulating Sparse Hamiltonians with Star Decompositions”. *Theory of Quantum Computation, Communication, and Cryptography*. Ed. by W. v. Dam, V. M. Kendon, and S. Severini. Lecture Notes in Computer Science 6519. Springer Berlin Heidelberg, 2011, pp. 94–103.

[60] N. Wiebe et al. “Simulating quantum dynamics on a quantum computer”. *Journal of Physics A: Mathematical and Theoretical* 44.44 (Nov. 2011), p. 445308.

[61] D. W. Berry et al. “Exponential Improvement in Precision for Simulating Sparse Hamiltonians”. *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. STOC ’14. New York, NY, USA: ACM, 2014, pp. 283–292.

[62] D. W. Berry, A. M. Childs, and R. Kothari. “Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters”. *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*. Oct. 2015, pp. 792–809.

[63] D. W. Berry et al. “Simulating Hamiltonian Dynamics with a Truncated Taylor Series”. *Physical Review Letters* 114.9 (Mar. 2015), p. 090502.

[64] A. M. Childs and R. Kothari. “Limitations on the simulation of non-sparse Hamiltonians”. *Quantum Information & Computation* 10.7 (July 2010), pp. 669–684.

[65] A. M. Childs. “On the Relationship Between Continuous- and Discrete-Time Quantum Walk”. *Communications in Mathematical Physics* 294.2 (Mar. 2010), pp. 581–603.

[66] L. Grover and T. Rudolph. “Creating superpositions that correspond to efficiently integrable probability distributions”. *arXiv:quant-ph/0208112* (Aug. 2002). arXiv: quant-ph/0208112.

[67] B. Giles and P. Selinger. “Exact synthesis of multiqubit Clifford+T circuits”. *Physical Review A* 87.3 (Mar. 2013), p. 032332.

[68] V. Kliuchnikov, D. Maslov, and M. Mosca. “Fast and Efficient Exact Synthesis of Single-qubit Unitaries Generated by Clifford and T Gates”. *Quantum Information & Computation* 13.7-8 (July 2013), pp. 607–630.

[69] D. Deutsch, A. Barenco, and A. Ekert. "Universality in Quantum Computation". *Proceedings: Mathematical and Physical Sciences* 449.1937 (June 1995), pp. 669–677.

[70] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge ; New York: Cambridge University Press, Jan. 2011.

[71] C. C. Paige and M. Wei. "History and generality of the CS decomposition". *Linear Algebra and its Applications* 208209 (Sept. 1994), pp. 303–326.

[72] R. R. Tucci. "A Rudimentary Quantum Compiler (2cnd Ed.)" *arXiv preprint quant-ph/9902062* (2006).

[73] M. Mottonen et al. "Quantum Circuits for General Multiqubit Gates". *Physical Review Letters* 93.13 (Sept. 2004), p. 130502.

[74] K. Manouchehri and J. Wang. *Physical Implementation of Quantum Walks*. Springer, 2014.

[75] S. L. Braunstein, A. Mann, and M. Revzen. "Maximal violation of Bell inequalities for mixed states". *Physical Review Letters* 68.22 (June 1992), pp. 3259–3261.

[76] J. Sakurai and J. J. Napolitano. *Sakurai - Modern Quantum Mechanics*. Pearson, July 2010.

[77] A. Imamoglu and Y. Yamamoto. "Turnstile device for heralded single photons: Coulomb blockade of electron and hole tunneling in quantum confined p-i-n heterojunctions". *Physical Review Letters* 72.2 (Jan. 1994), pp. 210–213.

[78] I. L. Chuang and Y. Yamamoto. "Simple quantum computer". *Physical Review A* 52.5 (Nov. 1995), pp. 3489–3496.

[79] J. I. Cirac and P. Zoller. "Quantum Computations with Cold Trapped Ions". *Physical Review Letters* 74.20 (May 1995), pp. 4091–4094.

[80] D. G. Cory, A. F. Fahmy, and T. F. Havel. "Ensemble quantum computing by NMR spectroscopy". en. *Proceedings of the National Academy of Sciences* 94.5 (Mar. 1997), pp. 1634–1639.

[81] N. A. Gershenfeld and I. L. Chuang. "Bulk Spin-Resonance Quantum Computation". en. *Science* 275.5298 (Jan. 1997), pp. 350–356.

[82] D. Loss and D. P. DiVincenzo. "Quantum computation with quantum dots". *Physical Review A* 57.1 (Jan. 1998), pp. 120–126.

[83] J. Kempe. "Quantum random walks: an introductory overview". *Contemporary Physics* 44.4 (2003), pp. 307–327.

[84] A. Chakrabarti, C. Lin, and N. K. Jha. "Design of Quantum Circuits for Random Walk Algorithms". 2012 IEEE Computer Society Annual Symposium on VLSI, Aug. 2012, pp. 135–140.

[85] A. Ahmadi et al. "On Mixing in Continuous-time Quantum Walks on Some Circulant Graphs". *Quantum Information and Computation* 3.6 (Nov. 2003), pp. 611–618.

[86] B. Tregenna et al. "Controlling discrete quantum walks: coins and initial states". *New Journal of Physics* 5.1 (July 2003), p. 83.

[87] A. M. Childs. "Universal Computation by Quantum Walk". *Physical Review Letters* 102.18 (May 2009), p. 180501.

[88] A. M. Childs, D. Gosset, and Z. Webb. "Universal Computation by Multiparticle Quantum Walk". *Science* 339.6121 (Feb. 2013), pp. 791–794.

[89] D. W. Berry and A. M. Childs. "Black-box Hamiltonian simulation and unitary implementation". *Quantum Information & Computation* 12.1-2 (2012), pp. 29–62.

[90] N. Shenvi, J. Kempe, and K. B. Whaley. "Quantum random-walk search algorithm". *Physical Review A* 67.5 (May 2003), p. 052307.

[91] F. Magniez et al. "Search via Quantum Walk". *SIAM Journal on Computing* 40.1 (Jan. 2011), pp. 142–164.

[92] L. Hogben, ed. *Handbook of Linear Algebra*. Vol. 33. Discrete Mathematics and Its Applications. Chapman and Hall/CRC, Nov. 2006.

[93] R. M. Gray. "Toeplitz and Circulant Matrices: A Review". *Foundations and Trends in Communications and Information Theory* 2.3 (2005), pp. 155–239.

[94] A. M. Childs and Y. Ge. "Spatial search by continuous-time quantum walks on crystal lattices". *Physical Review A* 89.5 (May 2014), p. 052337.

[95] M. K. Ng. *Iterative Methods for Toeplitz Systems (Numerical Mathematics and Scientific Computation)*. New York, NY, USA: Oxford University Press, Inc., 2004.

[96] D. Shepherd and M. J. Bremner. "Temporally unstructured quantum computation". en. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* (Jan. 2009), rspa.2008.0443.

[97] M. J. Bremner, R. Jozsa, and D. J. Shepherd. "Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy". en. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 467.2126 (Feb. 2011), pp. 459–472.

[98] C. L. Nehaniv, P. Dini, and P. Van Mieghem. *Construction of Interdependent Networks and Efficiency Improvements in their Stability*. Technical FP7-288021. Information and Communication Technologies, Sept. 2014.

[99] F. Harary, J. P. Hayes, and H.-J. Wu. "A survey of the theory of hypercube graphs". *Computers & Mathematics with Applications* 15.4 (1988), pp. 277–289.

[100] J. A. Gallian. "Dynamic Survey DS6: Graph Labeling". *Electronic Journal of Combinatorics* (Nov. 1997).

[101] E. Snchez-Burillo et al. "Quantum Navigation and Ranking in Complex Networks". *Scientific Reports* 2 (Aug. 2012).

[102] M. Mohseni et al. "Environment-assisted quantum walks in photosynthetic energy transfer". *The Journal of Chemical Physics* 129.17 (Nov. 2008), p. 174106.

[103] H. Krovi et al. "Quantum Walks Can Find a Marked Element on Any Graph". *Algorithmica* 74.2 (Mar. 2015), pp. 851–907.

[104] H. Buhrman and R. Spalek. "Quantum Verification of Matrix Products". *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*. SODA '06. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2006, pp. 880–889.

[105] F. Magniez and A. Nayak. "Quantum Complexity of Testing Group Commutativity". *Algorithmica* 48.3 (June 2007), pp. 221–232.

[106] G. Wang. "Quantum Algorithms for Approximating the Effective Resistances in Electrical Networks". *arXiv:1311.1851 [quant-ph]* (Nov. 2013).

[107] M. Mosca and P. Kaye. "Quantum Networks for Generating Arbitrary Quantum States". en. OSA, 2001, PB28.

[108] E. Sampathkumar. "On tensor product graphs". *Journal of the Australian Mathematical Society (Series A)* 20.03 (Nov. 1975), pp. 268–273.

[109] A. E. Brouwer, A. M. Cohen, and A. Neumaier. *Distance-Regular Graphs*. en. Vol. 18. Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer Berlin Heidelberg, 1989.

[110] K. Bryan and T. Leise. "The 25,000,000,000 Eigenvector: The Linear Algebra behind Google". *SIAM Review* 48.3 (Jan. 2006), pp. 569–581.

[111] T. Loke and J. B. Wang. "OptQC v1.3: An (updated) optimized parallel quantum compiler". *Computer Physics Communications* 207 (Oct. 2016), pp. 531–532.

[112] V. Bergholm et al. "Quantum circuits with uniformly controlled one-qubit gates". *Physical Review A* 71.5 (May 2005), p. 052330.

[113] F. S. Khan and M. Perkowski. "Synthesis of multi-qudit hybrid and d-valued quantum logic circuits by decomposition". *Theoretical Computer Science* 367.3 (Dec. 2006), pp. 336–346.

[114] A. De Vos and Y. Van Rentergem. "Multiple-valued reversible logic circuits". eng. *Multiple-Valued Logic and Soft Computing* 15.5-6 (2009), pp. 489–505.

[115] M. B. Alexis De Vos. "Reversible Computation, Quantum Computation, and Computer Architectures in Between." *Multiple-Valued Logic and Soft Computing* 18 (2012), pp. 67–81.

[116] K. Manouchehri. "Quantum Walks: Theory and Implementation". PhD thesis. University of Western Australia, Dec. 2010.

[117] B. D. Sutton. "Computing the complete CS decomposition". *Numerical Algorithms* 50.1 (Jan. 2009), pp. 33–65.

[118] E. Anderson et al. *LAPACK Users' Guide, Third Edition*. Society for Industrial and Applied Mathematics, 1999.

[119] S. D. Berry and J. B. Wang. "Two-particle quantum walks: Entanglement and graph isomorphism testing". *Physical Review A* 83.4 (Apr. 2011), p. 042317.

[120] P. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509.

[121] R. Cleve et al. "Quantum algorithms revisited". en. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (Jan. 1998), pp. 339–354.

[122] A. Montanaro. "Quantum walks on directed graphs". *arXiv:quant-ph/0504116* (Apr. 2005). arXiv: quant-ph/0504116.

[123] G. H. Golub and C. F. V. Loan. *Matrix Computations*. 3rd edition. Baltimore: Johns Hopkins University Press, Oct. 1996.

[124] C. Moler and C. Van Loan. "Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later". *SIAM Review* 45.1 (Jan. 2003), pp. 3–49.

[125] D. Lu et al. “Chiral quantum walks”. *Physical Review A* 93.4 (Apr. 2016), p. 042302.

[126] G. R. Carson, T. Loke, and J. B. Wang. “Entanglement dynamics of two-particle quantum walks”. en. *Quantum Information Processing* 14.9 (June 2015), pp. 3193–3210.

Appendix A

Journal articles

Here, I include all papers that have been published during my candidature (not including submitted manuscripts), in the order outlined in the publication list. For completeness, I also include a paper published in Quantum Information Processing with Georgina Carson and Jingbo Wang on the topic of "Entanglement dynamics of two-particle quantum walks" [126], which while not directly relevant to the subject of this thesis, was nonetheless produced during the duration of my candidature. For completeness, I also include a paper detailing the quantum circuit implementation of coined quantum walks, published prior to my candidature.

Computer Physics Communications 185 (2014) 3307–3316

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

OptQC: An optimized parallel quantum compiler☆

T. Loke, J.B. Wang*, Y.H. Chen

School of Physics, The University of Western Australia, 6009 Perth, Australia

ARTICLE INFO

Article history:
Received 4 June 2013
Received in revised form
22 July 2014
Accepted 28 July 2014
Available online 7 August 2014

Keywords:
Quantum computation
Quantum gates
Quantum circuit
Quantum compiler
Optimization
Stimulated annealing

ABSTRACT

The software package *Qcompiler* (Chen and Wang 2013) provides a general quantum compilation framework, which maps any given unitary operation into a quantum circuit consisting of a sequential set of elementary quantum gates. In this paper, we present an extended software *OptQC*, which finds permutation matrices P and Q for a given unitary matrix U such that the number of gates in the quantum circuit of $U = Q^T P^T U' P Q$ is significantly reduced, where U' is equivalent to U up to a permutation and the quantum circuit implementation of each matrix component is considered separately. We extend further this software package to make use of high-performance computers with a multiprocessor architecture using MPI. We demonstrate its effectiveness in reducing the total number of quantum gates required for various unitary operators.

Program summary

Program title: OptQC

Catalogue identifier: AEUA_v1_0

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEUA_v1_0.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html

No. of lines in distributed program, including test data, etc.: 178435

No. of bytes in distributed program, including test data, etc.: 491574

Distribution format: tar.gz

Programming language: Fortran, MPI.

Computer: Any computer with Fortran compiler and MPI library.

Operating system: Linux.

Classification: 4.15.

Nature of problem: It aims to minimize the number of quantum gates required to implement a given unitary operation.

Solution method: It utilizes a threshold-based acceptance strategy for simulated annealing to select permutation matrices P and Q for a given unitary matrix U such that the number of gates in the quantum circuit of $U = Q^T P^T U' P Q$ is minimized, where U' is equivalent to U up to a permutation. The decomposition of a unitary operator is performed by recursively applying the cosine–sine decomposition.

Running time: Running time increases with the size of the unitary matrix, as well as the prescribed maximum number of iterations for qubit permutation selection and the subsequent simulated annealing algorithm. Running time estimates are provided for each example in Section 4. All simulation results presented in this paper are obtained from running the program on the Fornax supercomputer managed by iVEC@UWA with Intel Xeon X5650 CPUs.

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author. Tel.: +61 8 64883790; fax: +61 8 64883790.
E-mail address: jingbo.wang@uwa.edu.au (J.B. Wang).

http://dx.doi.org/10.1016/j.cpc.2014.07.022
0010-4655/

1. Introduction

Quantum computation aims to solve problems that are classically intractable by harnessing intricate quantum correlations between densely encoded states in quantum systems [1]. A well-known example is Shor's algorithm for the factorization of numbers [2,3]. Quantum algorithms are designed to be implemented on quantum computers by means of a quantum circuit, which consists of qubits and quantum gates. It is therefore of vital importance to be able to obtain a quantum circuit representation for any given quantum algorithm (which is always described by a unitary matrix) in terms of an elementary set of quantum gates—this role is that of a quantum compiler.

Barenco et al. and Deutsch et al. [4,5] proved that any arbitrarily complex unitary operation can be implemented by a quantum circuit involving only one- or two-qubit elementary quantum logic gates. Earlier studies applied the standard triangularization or QR-factorization scheme with Givens rotations and Gray codes to map a quantum algorithm to a series of elementary gate operations [4–6,1]. Several research groups examined a more efficient and versatile scheme based on the cosine–sine decomposition was proposed and utilized [7–11]. De Vos et al. [12,13] looked into another decomposition scheme, namely the Birkhoff decomposition, which was found to provide simpler quantum circuits for certain types of unitary matrices than the cosine–sine decomposition. However, the Birkhoff decomposition does not work for general unitary matrices.

More recently, Chen and Wang [14] developed a general quantum compiler package written in Fortran, entitled the *Qcompiler*, which is based on the cosine–sine decomposition scheme and works for arbitrary unitary matrices. The number of gates required to implement a general 2^n-by-2^n unitary matrix using the CSD method scales as $O(4^n)$ [8,11]. In other words, the number of gates scales exponentially with the number of qubits. Thus, in any practical application of the CSD method to decomposing matrices, it is of considerable interest to reduce the number of gates required as much as possible.

In this work, we adopt the CSD method due to the reasons outlined above, and we split the unitary matrix U into an equivalent sequence of unitaries with the aim of reducing the number of gates required to implement the entire sequence of unitaries. In general, this means writing U as a sequence of s unitaries, i.e. $U = U_s U_{s-1} \ldots U_1$. At first glance, this seems counterintuitive, since if we were to apply the CSD to each unitary, this would increase the scaling of the number of gates required to $O(4^n s)$, which is undesirable. However, we note that (1) certain U_i can be decomposed more efficiently than CSD such as qubit permutation matrices; and (2) some matrices requires only a few gates when separately decomposed using the CSD method.

This paper is organized as follows. Section 2 describes in detail our approach for reducing the number of gates required to implement any given unitary matrix U. Section 3 details our developed program, called *OptQC*, that uses the methods described in Section 2 to reduce the number of gates required to implement any given unitary matrix. Some sample results using the program are given in Section 4, and then we discuss our conclusions and possible future work in Section 5.

2. Our approach

Suppose we are given an m-by-m (where $m = 2^n$) unitary matrix U. As mentioned above, we are interested in splitting U into a sequence of unitaries with the aim of reducing the total number of gates required to implement the entire sequence. One means of splitting U into an equivalent sequence of unitaries is by using permutation matrices. A permutation matrix is a square binary matrix that contains, in each row and column, precisely a single 1 with 0s everywhere else. For any permutation matrix P, its corresponding inverse is $P^{-1} = P^T$. For convenience, we also define an equivalent representation of permutations using lists—a permutation list p (lowercase) is equivalent to the permutation matrix P (uppercase) by the relation:

$$(P)_{i,j} = \delta_{p[\![i]\!],j}, \tag{1}$$

where δ is the Kronecker delta function, and $p[\![i]\!]$ denotes the ith list element of p. For example, the permutation list $p = \{2, 1, 4, 3\}$ corresponds to the 4-by-4 permutation matrix:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Now define $\mathrm{CSD}(M)$ to be the number of gates required to implement the unitary matrix M according to the CSD method. If we were to write U as $U = P^T U' P$ (where U' is equivalent to U up to a permutation), then we find that $\mathrm{CSD}(U) \neq \mathrm{CSD}(U') + \mathrm{CSD}(P) + \mathrm{CSD}(P^T)$ in general (note also that $\mathrm{CSD}(P) \neq \mathrm{CSD}(P^T)$). The general aim is thus to find a P that minimizes the total cost function $\mathrm{CSD}(U') + \mathrm{CSD}(P) + \mathrm{CSD}(P^T)$, with the obvious restriction that it has to be less than $\mathrm{CSD}(U)$.

In our approach, we write U as $U = Q^T P^T U' P Q$, where P and Q are both permutation matrices, and U' is equivalent to U up to a permutation. In general, P is allowed to be any permutation matrix ($m! = (2^n)!$ permutations possible), but Q is restricted to a class of permutation matrices that correspond to qubit permutations (only $n!$ permutations possible). The advantage of this approach is that qubit permutations can be easily implemented using a sequence of swap gates—so for a system with n qubits, it requires at most $n - 1$ swap gates to implement any qubit permutation. This also enables the program (in the parallel version) to start different threads at different points in the search space of $m!$ permutations by using different qubit permutations.

Let $s_{num}(Q)$ be the number of swap gates required to implement a qubit permutation matrix Q. Note that $s_{num}(Q) = s_{num}(Q^T)$, since the reverse qubit permutation would just be the same swap gates applied in reverse order. The total cost function c_{num} of implementing a given unitary $U = Q^T P^T U' P Q$ is then:

$$c_{num}(U) = \mathrm{CSD}(U') + \mathrm{CSD}(P) + \mathrm{CSD}(P^T) + 2s_{num}(Q). \tag{2}$$

To make the dependencies in this function clear, we write this as:

$$c_{num}(U, P, Q) = \mathrm{CSD}(PQUQ^T P^T) + \mathrm{CSD}(P) + \mathrm{CSD}(P^T) + 2s_{num}(Q), \tag{3}$$

which is the function that we aim to minimize with respect to P and Q.

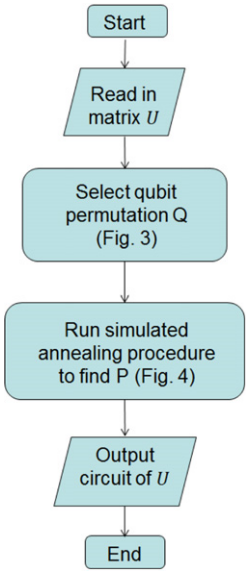


Fig. 1. Flowchart overview of the serial version of *OptQC*.

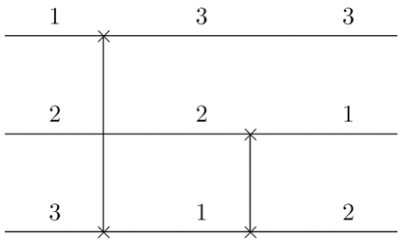


Fig. 2. Example implementation of qubit permutation $q = \{3, 1, 2\}$ using swap gates.

3. Program outline

We have developed a Fortran program, called *OptQC*, which reads in a unitary matrix U, minimizes the total cost function $c_{num}(U, P, Q)$, and outputs a quantum circuit that implements U. A significant portion of this program is based on the CSD code provided by the LAPACK library [15] and the recursive procedure implemented in *Qcompiler*, developed by Chen and Wang [14]. As with *Qcompiler*, the new *OptQC* program has two different branches, one treating strictly real unitary (i.e. orthogonal) matrices, and another treating arbitrary complex unitary matrices, with the former generally providing a circuit that is half in size of the latter [14].

Note that the CSD procedure requires the round up of the matrix dimension to the closest power of two, i.e. the dimension used is

$$m' = 2^{\lceil \log_2 m \rceil}. \tag{4}$$

The expanded unitary operator $\bar{U}$ is an m'-by-m' matrix, where

$$\left(\bar{U}\right)_{i,j} = \begin{cases} (U)_{i,j} & : i \leq m, j \leq m \\ \delta_{i,j} & : \text{otherwise} \end{cases} \tag{5}$$

which we will subsequently treat as the unitary U to be optimized via permutations.

In the following subsections we describe the key procedures in *OptQC*, depicted in Fig. 1, which serve to progressively reduce the total cost function $c_{num}(U, P, Q)$. We first detail the serial version of the program, followed by an extension to a parallel architecture using MPI.

3.1. Selection of qubit permutation

Qubit permutations are a class of permutations that are expressible in terms of a reordering of qubits, which can be efficiently implemented using swap gates that serve to interchange qubits. Recalling that U is of dimensions m-by-m (where $m = 2^n$), this implies that there are only $n!$ qubit permutations possible for a given U. A qubit permutation can be expressed as a list q (lowercase) of length n, or as a permutation matrix Q (uppercase) of dimensions m-by-m. A qubit permutation of length n requires at most $n - 1$ swap gates.

The selection of the qubit permutation matrix Q is done by varying Q and computing the corresponding change in the cost function c_{num}, while holding P constant as the identity matrix I. An example implementation of the $n = 3$ qubit permutation $q = \{3, 1, 2\}$ is shown in Fig. 2. By considering how the basis states are mapped to each other by q, a regular permutation list $\bar{q}$ of length m can be readily constructed from q, and then we use the relation between permutation lists and permutation matrices (see Eq. (1)) to obtain Q from $\bar{q}$.

We start the program with an identity qubit permutation q, i.e. $q[[i]] = i$ (corresponding to $Q = I$), and compute the corresponding cost of implementation $c_{num}(U, I, Q)$. Then, for some prescribed number of iterations j_{max}, we generate a random qubit permutation q' each time and compute the new cost as $c_{num}(U, I, Q')$. If the new cost is lower than the initial cost (recorded by $c_{num}(U, I, Q)$), the current

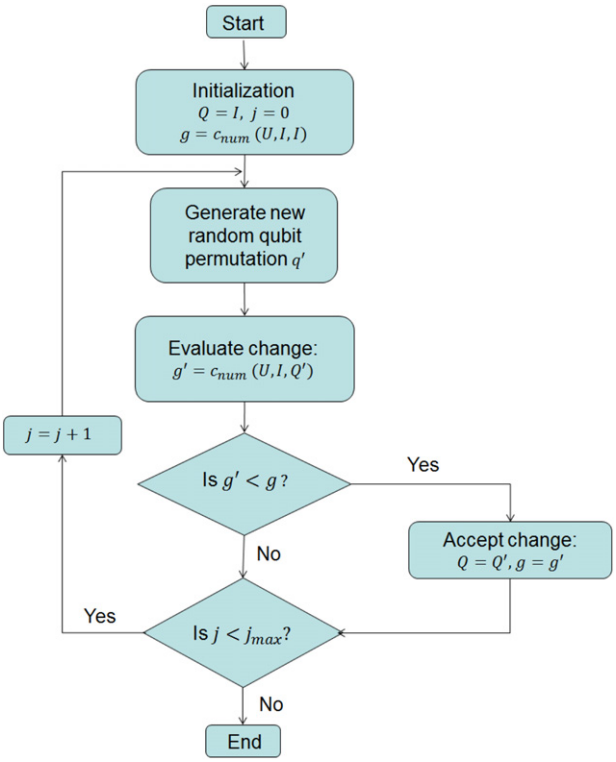


Fig. 3. Flowchart overview of the qubit selection procedure.

qubit permutation q is replaced by q'. Fig. 3 shows a flowchart overview of the qubit selection procedure. After this procedure, we have an optimized qubit permutation matrix Q, which will remain unchanged while we find the unrestricted permutation matrix P in the next section through a simulated annealing process.

3.2. Simulated annealing

Here, we aim to find an optimal permutation p' such that $c_{num}(U, P', Q) < c_{num}(U, P, Q)$ in the discrete search space of all $m!$ permutations. Given the massive size of the search space, use of a heuristic optimization method is practically necessary. Simulated annealing is one such method for finding a minimum in a discrete search space. In the OptQC program, we adopt a threshold acceptance based simulated annealing algorithm. There are three key components to the algorithm:

1. Cost function: the function to be minimized, i.e. $c_{num}(U, P, Q)$.
2. Neighborhood operator: the procedure that alters the current solution slightly by altering the current permutation p to a slightly different permutation p'. Our neighborhood operator acts to interchange any two *random* positions in p to form p'.
3. Threshold value: any 'bad' trades (increase in cost function) that are below some threshold value β are accepted, otherwise they are rejected. We define the threshold value as $\beta(P, Q) = \min(\lceil \alpha c_{num}(U, P, Q) \rceil, \lceil \alpha c_{num}(U, I, Q) \rceil)$, where $0 \leq \alpha < 1$. As such, the threshold value is taken to be the proportion α of the current number of gates (with a fixed maximum value of the proportion α of the initial number of gates to ensure that $\beta(P, Q)$ cannot grow arbitrarily large).

We start with p as the identity permutation. By iterating the neighborhood operator and evaluating the subsequent change in the number of gates, we accept the change in the permutation if it reduces the number of gates, or if the increase in the number of gates is below the threshold β. After some prescribed number of iterations i_{max}, we terminate the simulated annealing procedure, returning the permutation p_{min} that provides the minimum number of gates. Fig. 4 shows a flowchart overview of the simulated annealing procedure. Note that p_{min} is not necessarily the permutation p at the end of i_{max} iterations—rather, we keep track of p_{min} separately during the procedure.

3.3. Gate reduction procedure

Here, we focus on reducing the number of gates in some prescribed quantum circuit by combining 'similar' gates. In a quantum circuit, we can combine CUGs (controlled unitary gates) that apply the same unitary operation U_{op} to the same qubit, with all but one of the conditionals of the CUGs being the same. This reduction process is carried out after every application of the CSD method to a matrix—in particular, it is applied three times (to $PQUQ^TP^T$, P and P^T respectively) when computing $c_{num}(U, P, Q)$ (see Eq. (3)). While it does impose a significant computational overhead, it gives a better reflection of the true cost function, since the reduced circuit is the circuit that one would use for implementation. Fig. 5 shows an example result of applying the reduction procedure to a quantum circuit.

3.4. MPI parallelization

The program described above can be readily extended to a parallel architecture using MPI. Since the neighborhood operator in the simulated annealing procedure acts to interchange any two random positions, it follows that if the random number generator is seeded

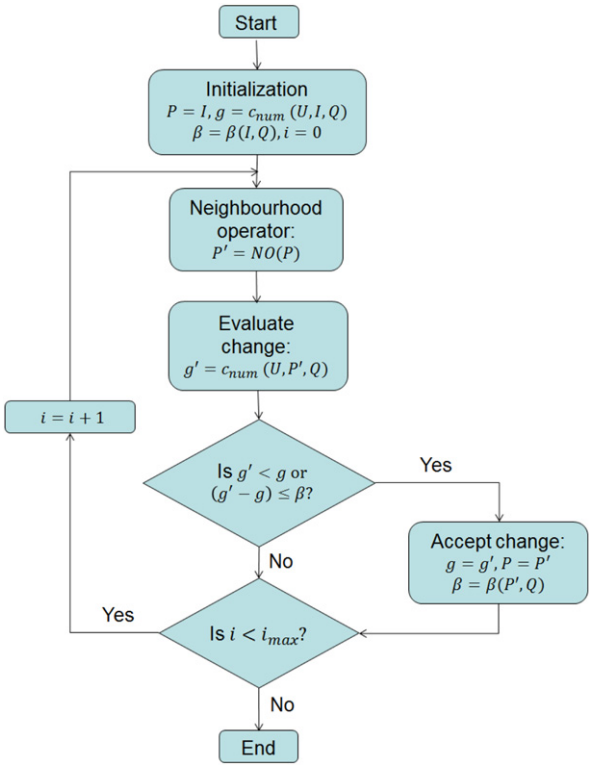


Fig. 4. Flowchart overview of the simulated annealing procedure.

differently, then a different set of positions would be interchanged, i.e. a different search through the space of permutations would be conducted. Similarly, the qubit permutation that is generated would also change when seeded differently, which enables the program to start threads at multiple locations in the search space of $m!$ permutations, so that the search procedure explores as much of the permutation space as possible. We do, however, restrict the root thread (thread index 0) of the program to use the identity qubit permutation for comparison purposes. Hence, using MPI, we can spawn a team of threads that simultaneously searches through the space of permutations independently and differently (by seeding the random number generator of each thread differently), and then collate the results to pick out the thread with the most optimal permutation, that is, it has the lowest $c_{num}(U, P, Q)$ value.

4. Results

We now apply the software program *OptQC* to various unitary operations to obtain corresponding optimized quantum circuits. All the results shown here are obtained using parameters $i_{max} = 40,000$, $j_{max} = 1000$ and $\alpha = 0.01$ we choose this α value because it provides, on average, the best results for the unitary operators being considered in this paper. Using these parameters, we run *OptQC* on the supercomputer Fornax with Intel Xeon X5650 CPUs, managed by iVEC@UWA, using 8 nodes with 12 cores on each (i.e. 96 threads).

4.1. Real unitary matrix

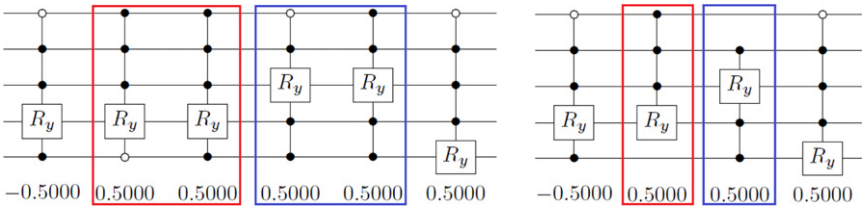
A random real unitary (i.e. orthogonal) matrix is given below:

$$U = \begin{pmatrix} 0.0438 & 0 & 0 & 0 & 0.9990 & 0 & 0 & 0 \\ 0.1297 & 0.8689 & -0.2956 & 0 & -0.0057 & 0.1538 & -0.3423 & 0 \\ -0.2923 & 0 & 0.6661 & 0 & 0.0128 & 0 & -0.6861 & 0 \\ -0.0061 & -0.0412 & 0.0140 & 0.7058 & 0.0003 & 0.3008 & 0.0162 & -0.6397 \\ 0.9147 & 0 & 0.4021 & 0 & -0.0401 & 0 & 0 & 0 \\ 0.0185 & 0.1242 & -0.0422 & 0.3961 & -0.0008 & -0.9073 & -0.0489 & 0 \\ 0.2424 & -0.4762 & -0.5524 & 0 & -0.0106 & 0 & -0.6397 & 0 \\ 0.0051 & 0.0343 & -0.0117 & -0.5874 & -0.0002 & -0.2503 & -0.0135 & -0.7686 \end{pmatrix}.$$

Note that this matrix is not completely filled, otherwise no reduction via permutations would generally be possible. By using *OptQC*, the reduction process gives the following results for the thread which achieves the optimal solution:

- No optimization: $c_{num}(U, I, I) = 29$ gates.
- After selection of an optimized qubit permutation q: $c_{num}(U, I, Q) = 1 + 26 + 1 = 28$ gates.
- After simulated annealing process for the permutation p: $c_{num}(U, P, Q) = 1 + 2 + 16 + 2 + 1 = 22$ gates.

Hence, we achieve a reduction of ∼25% from the original number of gates. Fig. 6 shows a comparison between the original and optimized circuit for U. Runtime for this calculation is ∼14.5 s.



(a) Circuit before reduction procedure. (b) Circuit after reduction procedure.

Fig. 5. Example of applying the reduction procedure to a quantum circuit.

4.2. Quantum walk operators

One important class of unitary operators are quantum walk operators—in particular, discrete-time quantum walk (DTQW) step operators [16–18]. For a given undirected graph $G(V, E)$, defined by a vertex set V and edge set E, we can define the DTQW step operator $U = SC$, where S and C are the shifting and coin operators respectively. The shifting operator acts to swap coin states that are connected by an edge, and the coin operator acts to mix the coin states at each individual vertex.

4.2.1. 8-star graph

The 8-star graph (shown in Fig. 7) is a graph with 1 center vertex connected to 8 leaf vertices by undirected edges. Using the Grover coin operator, the resulting quantum walk operator on this graph corresponds to a 16-by-16 real unitary matrix, as given below:

$$U = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}.$$

By using *OptQC*, the reduction process gives the following results for the thread which achieves the optimal solution:

- No optimization: $c_{num}(U, I, I) = 34$ gates.
- After selection of an optimized qubit permutation: $c_{num}(U, I, Q) = 27$ gates.
- After simulated annealing process to select a permutation p: $c_{num}(U, P, Q) = 0 + 2 + 19 + 2 + 0 = 23$ gates.

Hence, we achieve a reduction of ∼32% from the original number of gates. Fig. 8 shows the optimized circuit obtained for U. Runtime for this calculation is ∼47 s.

4.2.2. 3rd generation 3-Cayley tree

The 3rd generation 3-Cayley tree (abbreviated as the 3CT3 graph) is a tree of 3 levels in which all interior nodes have degree 3, as shown in Fig. 9a. The corresponding quantum walk operator using the Grover coin operator is shown in Fig. 9b—the quantum walk operator U is a 42-by-42 real unitary matrix (which is fairly sparse), which, for the purposes of the decomposition, is expanded to a 64-by-64 unitary matrix as per Eq. (5).

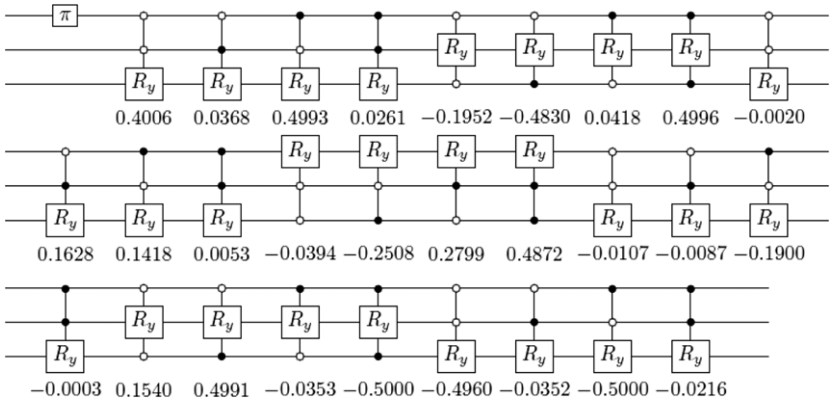
By using *OptQC*, the reduction process gives the following results for the thread which achieves the optimal solution:

- No optimization: $c_{num}(U, I, I) = 996$ gates.
- After selection of an optimized qubit permutation: $c_{num}(U, I, Q) = 3 + 345 + 3 = 351$ gates.
- After simulated annealing process to select a permutation p: $c_{num}(U, P, Q) = 3 + 33 + 231 + 30 + 3 = 300$ gates.

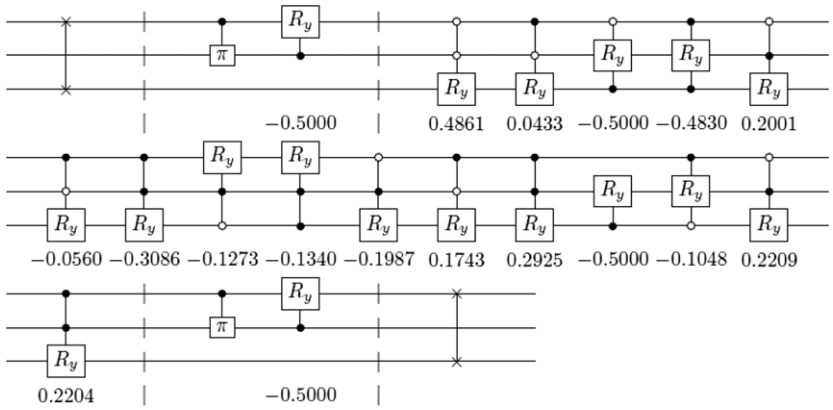
Hence, we achieve a reduction of ∼70% from the original number of gates. Runtime for this calculation is ∼12 min. Fig. 10 shows the time-series for $c_{num}(U, P, Q)$ during both the qubit permutation selection phase and the simulated annealing process (separated by a dotted line) to achieve the above result.

4.3. Quantum Fourier transform

Quantum Fourier transform is the quantum counterpart of the discrete Fourier transform in classical computing. It is an essential ingredient in several well-known quantum algorithms, such as Shor's factorization algorithm [2] and the quantum phase estimation



(a) Original circuit—29 gates.



(b) Optimized circuit—22 gates.

Fig. 6. Result of quantum circuit optimization as performed by *OptQC* on a random real unitary matrix. In (b), the dashed vertical lines separate the circuit for each matrix—from left to right, this corresponds to Q, P, U', P^T and Q^T respectively.

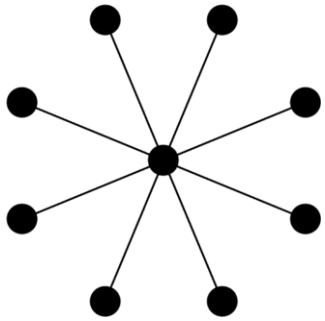


Fig. 7. The 8-star graph.

algorithm [19]. The matrix representation of the quantum Fourier transform on n dimensions is given by:

$$(\mathrm{QFT})_{jk} = \frac{1}{\sqrt{n}}\omega^{jk}, \quad \text{where } \omega = \exp(2\pi i/n). \tag{6}$$

An efficient quantum circuit implementation of the quantum Fourier transform is given in [1], which scales logarithmically as $O(\log(n)^2)$. Such a circuit implementation for $n = 2^6 = 64$ is shown in Fig. 11.

Now, let us apply *OptQC* to the corresponding 64-by-64 complex unitary operator, given by Eq. (6). With $\alpha = 0.002$, the reduction process gives the following results for the thread achieving an optimal solution:

- No optimization: $c_{num}(U, I, I) = 4095$ gates.
- After selection of an optimized qubit permutation: $c_{num}(U, I, Q) = 5 + 3577 + 5 = 3587$ gates.
- After simulated annealing process to select a permutation p: $c_{num}(U, P, Q) = 5 + 69 + 3359 + 70 + 5 = 3508$ gates.

Hence, we achieve a reduction of ∼14% from the original number of gates. Runtime of this calculation is ∼20 min. Fig. 12 shows the time-series for $c_{num}(U, P, Q)$ during both the qubit permutation selection phase and the simulated annealing process (separated by a dotted line) to achieve the above result. Clearly, this result is by far inferior to the quantum circuit of only 24 gates shown in Fig. 11. Similarly, the *OptQC* package would not be able to provide quantum circuits as efficient as those presented in [18,20] for the implementation of quantum walks

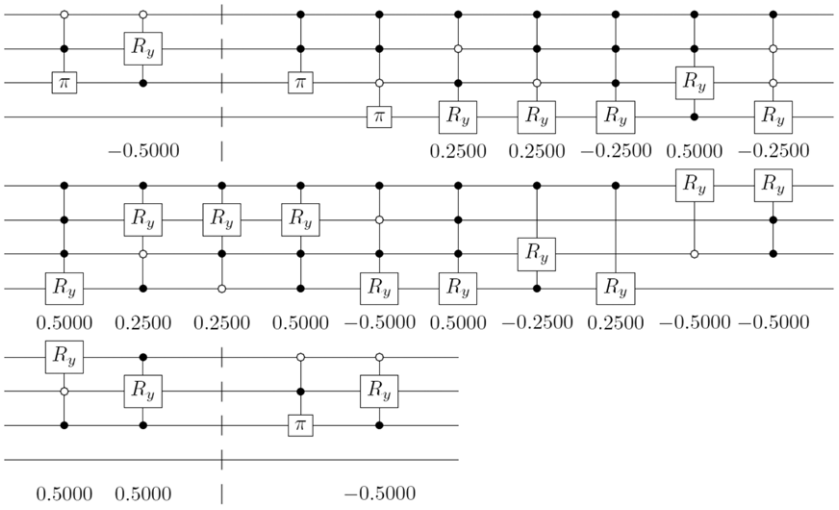
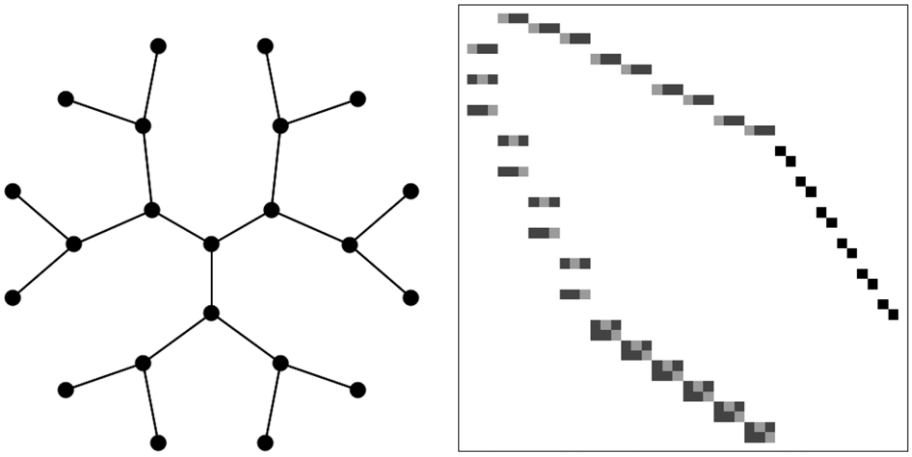


Fig. 8. Optimized circuit (with 23 gates) for the quantum walk operator of the 8-star graph.



(a) 3CT3 graph. (b) Quantum walk operator for the 3CT3 graph.

Fig. 9. The 3CT3 graph and its corresponding quantum walk operator using the Grover coin operator. The colors/shades in (b) denote the matrix entries for −1/3 (light gray), 2/3 (dark gray) and 1 (black)—all other matrix entries are 0 (white).

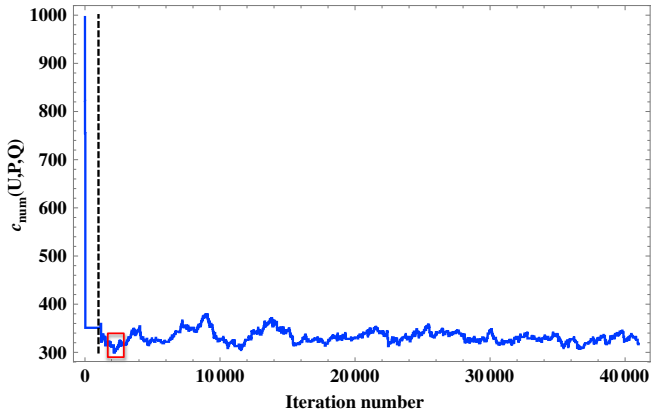


Fig. 10. Time-series of $c_{num}(U, P, Q)$ during the simulated annealing process for the thread which obtains the optimal solution. The original number of gates required by the CSD method to implement U is 996; selection of a qubit permutation reduces this cost to 351 gates, which is used as the starting point for the simulated annealing process. The red box indicates the region where the optimal solution of 300 gates is achieved. The iterations before the dotted line indicate the qubit permutation selection phase, and the subsequent iterations show the simulated annealing process.

on highly symmetric graphs. This is to be expected, since the CS decomposition is a general technique that decomposes a given unitary into a fixed circuit structure using many conditional gates, with an upper bound of $O(4^n)$. This algorithm is performed without foreknowledge or explicitly exploiting the structure of the unitary, which would clearly be crucial in achieving the lowest possible number of gates for a given unitary, as exemplified by the above examples. Instead, the *OptQC* package is designed to work for any arbitrary unitary operator

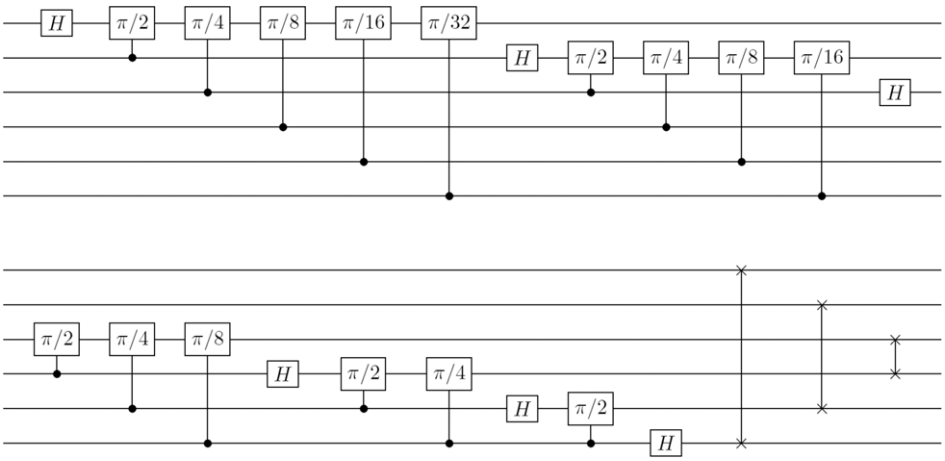


Fig. 11. Circuit implementation of quantum Fourier transform for $n = 64$.

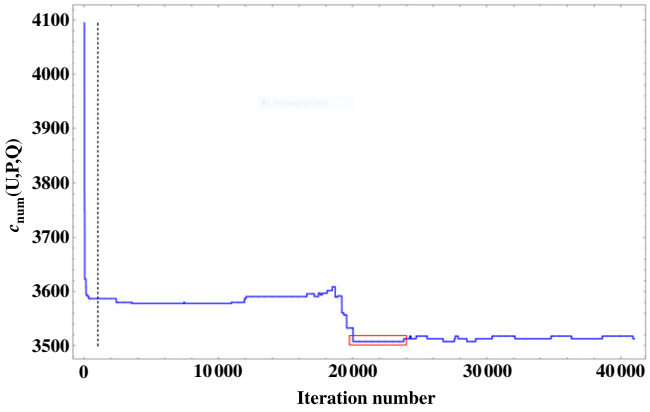


Fig. 12. Time-series of $c_{num}(U, P, Q)$ during the simulated annealing process for the thread which obtains the optimal solution. The original number of gates required by the CSD method to implement U is 4095; selection of a qubit permutation reduces this cost to 3587 gates, which is used as the starting point for the simulated annealing process. The red box indicates the region where the optimal solution of 3508 gates is achieved. The iterations before the dotted line indicate the qubit permutation selection phase, and the subsequent iterations show the simulated annealing process.

for which we do not already have an efficient quantum circuit implementation of, for example, quantum walk operators on arbitrarily complex graphs. In such cases, we have demonstrated that the *OptQC* package provides optimized quantum circuits that are far more efficient than the original *Qcompiler*.

5. Conclusion and future work

We have developed an optimized quantum compiler, named as *OptQC*, that runs on a parallel architecture to minimize the number of gates in the resulting quantum circuit of a unitary matrix U. This is achieved by finding permutation matrices Q and P such that $U = Q^T P^T U' P Q$ requires less total number of gates to be implemented, where the implementation for each matrix is considered separately. Decompositions of unitary matrices is done using the CSD subroutines provided in the LAPACK library [15] and adapted from *Qcompiler* [14]. *OptQC* utilizes an optimal selection of qubit permutations Q, a simulated annealing procedure to find P, and a combination of similar gates in order to reduce the total number of gates required as much as possible. We find that for many different types of unitary operators, *OptQC* is able to reduce the number of gates required by a significant amount, but its efficacy does vary depending on the unitary matrix given. In particular, this optimization procedure works well for sparse unitary matrices.

For future work, we hope to look at characterizing the optimal solutions reached to see if the matrix U' (and the associated permutation P) have some common preferential structure that leads to a reduced cost of implementation using the CSD method. Such information could be used to implement a guided search for the optimal solution, rather than using random adjustments of the permutation matrix. We also want to characterize 'bad' permutations (that is, permutations with a large cost) and avoid them in the search procedure, perhaps by eliminating the conjugacy class of 'bad' permutations from the search space.

Acknowledgments

Our work was supported through the use of advanced computing resources located at iVEC@UWA, as well as funding for a summer internship by iVEC. The authors would like to acknowledge valuable discussions with Chris Harris at iVEC@UWA, and also thank the referees for their constructive comments and suggestions. T.L. is supported by the International Postgraduate Research Scholarship, Australian Postgraduate Award and the Bruce and Betty Green Postgraduate Research Top-Up Scholarship.

References

[1] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, New York, 2011.
[2] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. 26 (1997) 1484–1509.
[3] L.M.K. Vandersypen, M. Steffen, G. Breyta, C.S. Yannoni, M.H. Sherwood, I.L. Chuang, Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance, Nature 414 (2001) 883–887.
[4] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin, H. Weinfurter, Elementary gates for quantum computation, Phys. Rev. A 52 (1995) 3457–3467.
[5] D. Deutsch, A. Barenco, A. Ekert, Universality in quantum computation, Proc. Math. Phys. Sci. 449 (1995) 669–677.
[6] G. Cybenko, Reducing quantum computations to elementary unitary operations, Comput. Sci. Eng. 3 (2001) 27–32.
[7] R.R. Tucci, A Rudimentary Quantum Compiler, second ed., 1999, arXiv preprint quant-ph/9902062.
[8] M. Mttnen, J.J. Vartiainen, V. Bergholm, M.M. Salomaa, Quantum circuits for general multiqubit gates, Phys. Rev. Lett. 93 (2004) 130502.
[9] V. Bergholm, J.J. Vartiainen, M. Mttnen, M.M. Salomaa, Quantum circuits with uniformly controlled one-qubit gates, Phys. Rev. A 71 (2005) 052330.
[10] F.S. Khan, M. Perkowski, Synthesis of multi-qudit hybrid and d-valued quantum logic circuits by decomposition, Theoret. Comput. Sci. 367 (2006) 336–346.
[11] K. Manouchehri, J.B. Wang, Quantum random walks without walking, Phys. Rev. A 80 (2009) 060304.
[12] A. De Vos, Y. Van Rentergem, Multiple-valued reversible logic circuits, Mult.-Valued Logic Soft Comput. 15 (2009) 489–505.
[13] M.B. Alexis De Vos, Reversible computation, quantum computation, and computer architectures in between, Mult.-Valued Logic Soft Comput. 18 (2012) 67–81.
[14] Y.G. Chen, J.B. Wang, Qcompiler: quantum compilation with the CSD method, Comput. Phys. Commun. 184 (2013) 853–865.
[15] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, LAPACK Users' Guide, third ed., Society for Industrial and Applied Mathematics, 1999.
[16] J. Kempe, Quantum random walks: an introductory overview, Contemp. Phys. 44 (2003) 307–327.
[17] S.D. Berry, J.B. Wang, Two-particle quantum walks: entanglement and graph isomorphism testing, Phys. Rev. A 83 (2011).
[18] T. Loke, J.B. Wang, Efficient circuit implementation of quantum walks on non-degree-regular graphs, Phys. Rev. A 86 (2012).
[19] R. Cleve, A. Ekert, C. Macchiavello, M. Mosca, Quantum algorithms revisited, Proc. R. Soc. Lond. Ser. A: Math. Phys. Eng. Sci. 454 (1998) 339–354.
[20] B. Douglas, J. Wang, Efficient quantum circuit implementation of quantum walks, Phys. Rev. A 79 (2009).

nature COMMUNICATIONS

ARTICLE

Received 29 Oct 2015 | Accepted 4 Apr 2016 | Published 5 May 2016

DOI: 10.1038/ncomms11511 OPEN

Efficient quantum walk on a quantum processor

Xiaogang Qiang[1,*], Thomas Loke[2,*], Ashley Montanaro[3], Kanin Aungskunsiri[1], Xiaoqi Zhou[1,4], Jeremy L. O'Brien[1], Jingbo B. Wang[2] & Jonathan C.F. Matthews[1]

The random walk formalism is used across a wide range of applications, from modelling share prices to predicting population genetics. Likewise, quantum walks have shown much potential as a framework for developing new quantum algorithms. Here we present explicit efficient quantum circuits for implementing continuous-time quantum walks on the circulant class of graphs. These circuits allow us to sample from the output probability distributions of quantum walks on circulant graphs efficiently. We also show that solving the same sampling problem for arbitrary circulant quantum circuits is intractable for a classical computer, assuming conjectures from computational complexity theory. This is a new link between continuous-time quantum walks and computational complexity theory and it indicates a family of tasks that could ultimately demonstrate quantum supremacy over classical computers. As a proof of principle, we experimentally implement the proposed quantum circuit on an example circulant graph using a two-qubit photonics quantum processor.

[1] Centre for Quantum Photonics, H.H. Wills Physics Laboratory and Department of Electrical and Electronic Engineering, University of Bristol, Bristol BS8 1UB, UK. [2] School of Physics, The University of Western Australia, Crawley, WA 6009, Australia. [3] School of Mathematics, University of Bristol, Bristol BS8 1TW, UK. [4] State Key Laboratory of Optoelectronic Materials and Technologies and School of Physics, Sun Yat-sen University, Guangzhou 510275, China. * These authors contributed equally to this work. Correspondence and requests for materials should be addressed to J.B.W. (email: jingbo.wang@uwa.edu.au) or to J.C.F.M. (email: jonathan.matthews@bristol.ac.uk).

Quantum walks are the quantum mechanical analogue of the well-known classical random walk and they have established roles in quantum information processing[1–3]. In particular, they are central to quantum algorithms created to tackle database search[4], graph isomorphism[5–7], network analysis and navigation[8,9], and quantum simulation[10–12], as well as modelling biological processes[13,14]. Meanwhile, physical properties of quantum walks have been demonstrated in a variety of systems, such as nuclear magnetic resonance[15,16], bulk[17] and fibre[18] optics, trapped ions[19–21], trapped neutral atoms[22] and photonics[23,24]. Almost all physical implementations of quantum walk so far followed an analogue approach as for quantum simulation[25], whereby the apparatus is dedicated to implement specific instances of Hamiltonians without translation onto quantum logic. However, there is no existing method to implement analogue quantum simulations with error correction or fault tolerance, and they do not scale efficiently in resources when simulating broad classes of large graphs. Some exceptions of demonstrations of quantum walks, such as ref. 15, adopted the qubit model, but did not discuss potentially efficient implementation of quantum walks.

Efficient quantum circuit implementations of continuous-time quantum walks (CTQWs) have been presented for sparse and efficiently row-computable graphs[26,27], and specific non-sparse graphs[28,29]. However, the design of quantum circuits for implementing CTQWs is in general difficult, since the time-evolution operator is time dependent and non-local[1]. A subset of circulant graphs have the property that their eigenvalues and eigenvectors can be classically computed efficiently[30,31]. This enables construction of a scheme that efficiently outputs the quantum state $|\psi(t)\rangle$, which corresponds to the time-evolution state of a CTQW on corresponding graphs. One can then either implement further quantum circuit operations or perform direct measurements on $|\psi(t)\rangle$ to extract physically meaningful information. For example the 'SWAP test'[32] can be used to estimate the similarity of dynamical behaviours of two circulant Hamiltonians operating on two different initial states, as shown in Fig. 1a. This procedure can also be adapted to study the stability of quantum dynamics of circulant molecules (for example, the DNA Möbius strips[33]) in a perturbational environment[34,35]. When measuring $|\psi(t)\rangle$ in the computational basis we can sample the probability distribution

$$p(x) := |\langle x|\psi(t)\rangle|^2 \qquad (1)$$

that describes the probability of observing the quantum walker at position $x \in \{0, 1\}^n$—an n-bit string, labelling one of the 2^n vertices of the given graph, as shown in Fig. 1b. Sampling of this form is sufficient to solve various search and characterization problems[4,9], and can be used to deduce critical parameters of the quantum walk, such as mixing time[2].

Here we present efficient quantum circuits for implementing CTQWs on circulant graphs with an eigenvalue spectrum that can be classically computed efficiently. These quantum circuits provide the time-evolution states of CTQWs on circulant graphs exponentially faster than best previously known methods[30]. We report a proof-of-principle experiment, where we implement CTQWs on an example circulant graph (namely the complete graph of four vertices) using a two-qubit photonics quantum processor to sample the probability distributions and perform state tomography on the output state of a CTQW. We also provide evidence from computational complexity theory that the probability distributions $p(x)$ that are output from the circuits of this circulant form are in general hard to sample from using a classical computer, implying our scheme also provides an exponential speedup for sampling. We adapt the methodology of refs 36–38 to show that if there did exist a classical sampler for a somewhat more general class of circuits, then this would have the following unlikely complexity-theoretic implication: the infinite tower of complexity classes known as the polynomial hierarchy would collapse. This evidence of hardness exists despite the classical efficiency with which properties of the CTQW, such as the eigenvalues of circulant graphs, can be computed on a classical machine.

Results

Quantum circuit for CTQW on circulant graph. For an undirected graph G of N vertices, a quantum particle (or 'quantum walker') placed on G evolves into a superposition $|\psi(t)\rangle$ of states in the orthonormal basis $\{|1\rangle, |2\rangle, \ldots, |N\rangle\}$ that correspond to vertices of G. The exact evolution of the CTQW is governed by connections between the vertices of G: $|\psi(t)\rangle = \exp(-itH)|\psi(0)\rangle$ where the Hamiltonian is given by $H = \gamma A$ for hopping rate per edge per unit time γ and where A is the N-by-N symmetric adjacency matrix, whose entries are $A_{jk} = 1$, if vertices j and k are connected by an edge in G, and $A_{jk} = 0$ otherwise[1]. The dynamics of a CTQW on a graph with N vertices can be evaluated in time poly(N) on a classical computer. When a CTQW takes place on a graph G of exponential size, that is, $N = 2^n$ for an input of size n, it becomes interesting to use quantum processors to simulate dynamics.

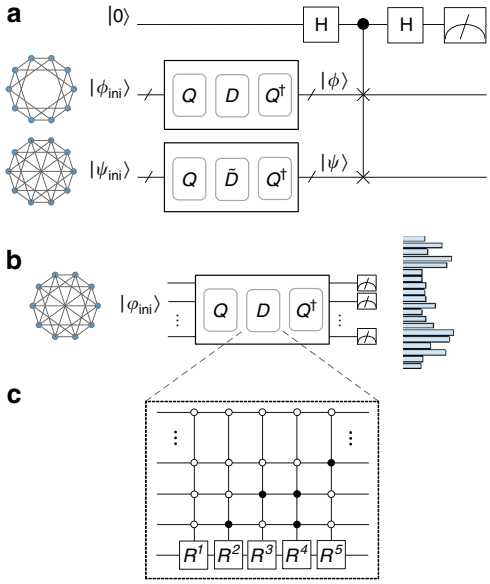


Figure 1 | Applications for generating the time-evolution state of circulant Hamiltonians. (**a**) The SWAP test[32] can be used to estimate the similarity of two evolution states of two similar circulant systems, or when one of the Hamiltonians is non-circulant but efficiently implementable. In brief, an ancillary qubit is entangled with the output states ψ and ϕ of two compared processes according to $\frac{1}{2}|0\rangle[|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle] + \frac{1}{2}|1\rangle[|\phi\rangle|\psi\rangle - |\psi\rangle|\phi\rangle]$. On measuring the ancillary qubit we obtain outcome '1' with probability $\frac{1}{2}(1 - |\langle\phi|\psi\rangle|^2)$—the probability of observing '1' indicates the similarity of dynamical behaviours of the two processes. See its complexity analysis in Supplementary Note 1. (**b**) Probability distributions are sampled by measuring the evolution state in a complete basis, such as the computational basis. (**c**) An example of the quantum circuit for implementing diagonal unitary operator $D = \exp(-it\Lambda)$, where the circulant Hamiltonian has 5 non-zero eigenvalues. The open and solid circles represent the control qubits as 'if $|0\rangle$' and 'if $|1\rangle$', respectively. $R^i = [1, 0; 0, \exp(-it\lambda_i)](i = 1, \cdots, 5)$, where λ_i is the corresponding eigenvalue.

Circulant graphs are defined by symmetric circulant adjacency matrices for which each row j when right rotated by one element, equals the next row $j+1$—for example, complete graphs, cycle graphs and Mobius ladder graphs are all subclasses of circulant graphs, and further examples are shown in Supplementary Note 2. It follows that Hamiltonians for CTQWs on any circulant graph have a symmetric circulant matrix representation, which can be diagonalized by the unitary Fourier transform[31], that is, $H=Q^{\dagger}\Lambda Q$, where

$$Q_{jk}=\frac{1}{\sqrt{N}}\omega^{jk},\ \omega=\exp(2\pi i/N) \qquad (2)$$

and Λ is a diagonal matrix containing eigenvalues of H, which are all real and whose order is determined by the order of the eigenvectors in Q. Consequently, we have $\exp(-itH)=Q^{\dagger}\exp(-it\Lambda)Q$, where the time dependence of $\exp(-itH)$ is confined to the diagonal unitary operator $D=\exp(-it\Lambda)$.

The Fourier transformation Q can be implemented efficiently by the well-known QFT quantum circuit[39]. For a circulant graph that has $N=2^n$ vertices, the required QFT of N dimensions can be implemented with $O((\log N)^2)=O(n^2)$ quantum gates acting on $O(n)$ qubits. To implement the inverse QFT, the same circuit is used in reverse order with phase gates of opposite sign. D can in general be implemented using at most $N=2^n$ controlled-phase gates with phase values being a linear function of t, because an arbitrary phase can be applied to an arbitrary basis state, conditional on at most n–1 qubits. However, given a circulant graph that has $O(\mathrm{poly}(n))$ non-zero eigenvalues, only $O(\mathrm{poly}(n))$ controlled-phase gates are needed to implement D. If the given circulant graph has $O(2^n)$ distinct eigenvalues, which can be characterized efficiently (such as the cycle graphs and Mobius ladder graphs), then we are still able to implement the diagonal unitary operator D using polynomial quantum resources. A general construction of efficient quantum circuits for D was given by Childs[40], and is shown in Supplementary Fig. 1 and Supplementary Note 3 for completeness. Thus, the quantum circuit implementations of CTQWs on circulant graphs can be constructed, which have an overall complexity of $O(\mathrm{poly}(n))$, and act on at most $O(n)$ qubits. Compared with the best-known classical algorithm based on fast Fourier transform, that has the computational complexity of $O(n2^n)$ (ref. 30), the proposed quantum circuit implementation generates the evolution state $|\psi(t)\rangle$ with an exponential advantage in speed.

Experimental demonstration. To demonstrate implementation of our scheme with two qubits, we have built photonic quantum logic to simulate CTQWs on the K_4 graph—a complete graph with self loops on four vertices (Fig. 2a). The family of complete graphs K_N are a special kind of circulant graph, with an adjacency matrix A where $A_{jk}=1$ for all j, k. Their Hamiltonian has only 2 distinct eigenvalues, 0 and $N\gamma$. Therefore, the diagonal matrix of eigenvalues of K_4 is $\Lambda=\mathrm{diag}(\{4\gamma,0,0,0\})$. We can readily construct the quantum circuit for implementing CTQWs on K_4 based on diagonalization, using the QFT matrix. However, the choice of using the QFT matrix as the eigenbasis of Hamiltonian is not strictly necessary—any equivalent eigenbasis can be selected. Through the diagonalization using Hadamard eigenbasis, an alternative efficient quantum circuit for implementing CTQWs on K_4 is shown in Fig. 2b, which can be easily extended to K_N.

We built a configurable two-qubit photonics quantum processor (Fig. 2c), adapting the entanglement-based technique presented in ref. 41, and implemented CTQWs on K_4 graph with various evolving times and initial states. Specifically, we prepared two different initial states $|\varphi_{\mathrm{ini}}\rangle_1=[1,0,0,0]'$ and $|\varphi_{\mathrm{ini}}\rangle_2=\frac{1}{\sqrt{2}}[1,1,0,0]'$, which represent the quantum walker starting from vertex 1, and the superposition of vertices 1 and 2, respectively. We chose the evolution time following the list $\{0,\frac{1}{8}\pi,\frac{2}{8}\pi,\frac{3}{8}\pi,\frac{4}{8}\pi,\frac{5}{8}\pi,\frac{6}{8}\pi,\frac{7}{8}\pi,\pi\}$, which covers the whole

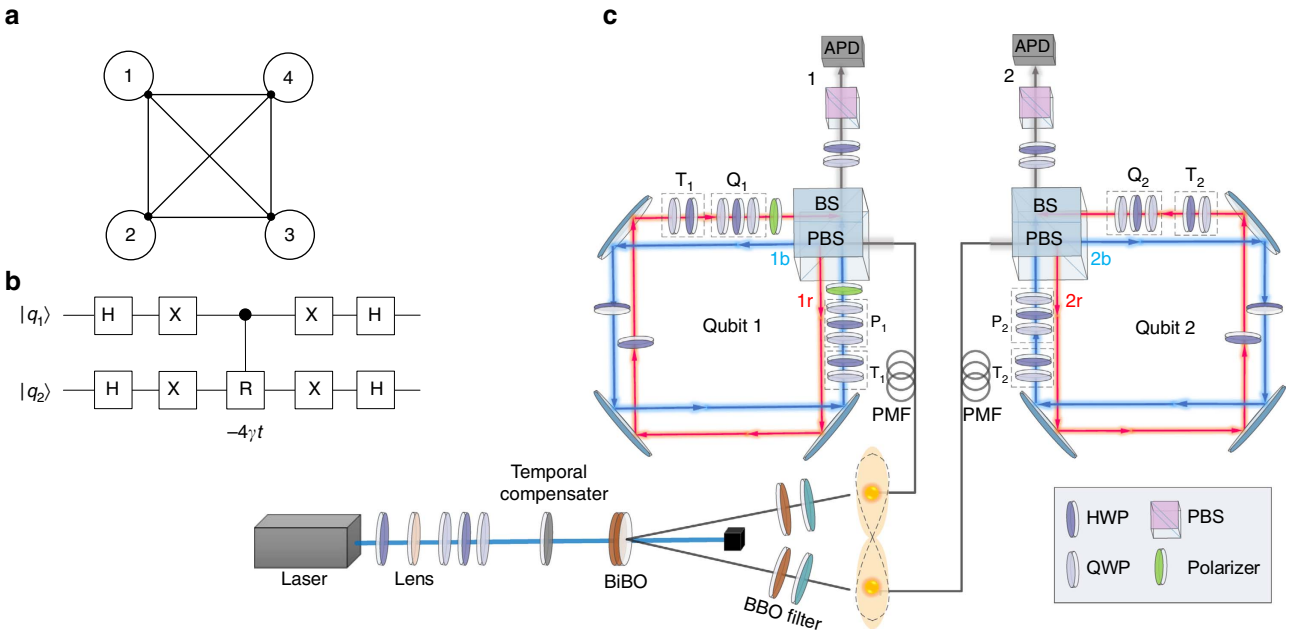


Figure 2 | The schematic diagram and set-up of experimental demonstration. (**a**) The K_4 graph. (**b**) The quantum circuit for implementing CTQW on the K_4 graph. This can also be used to implement CTQW on the K_4 graph without self-loops, up to a global phase factor $\exp(i\gamma t)$. H and X represent the Hadamard and Pauli-X gate, respectively. $R=(1,0;0,\exp(-i4\gamma t))$ is a phase gate. (**c**) The experimental set-up for a reconfigurable two-qubit photonics quantum processor, consisting of a polarization-entangled photon source using paired type-I BiBO crystal in the sandwich configuration and displaced Sagnac interferometers. See further details in Methods.

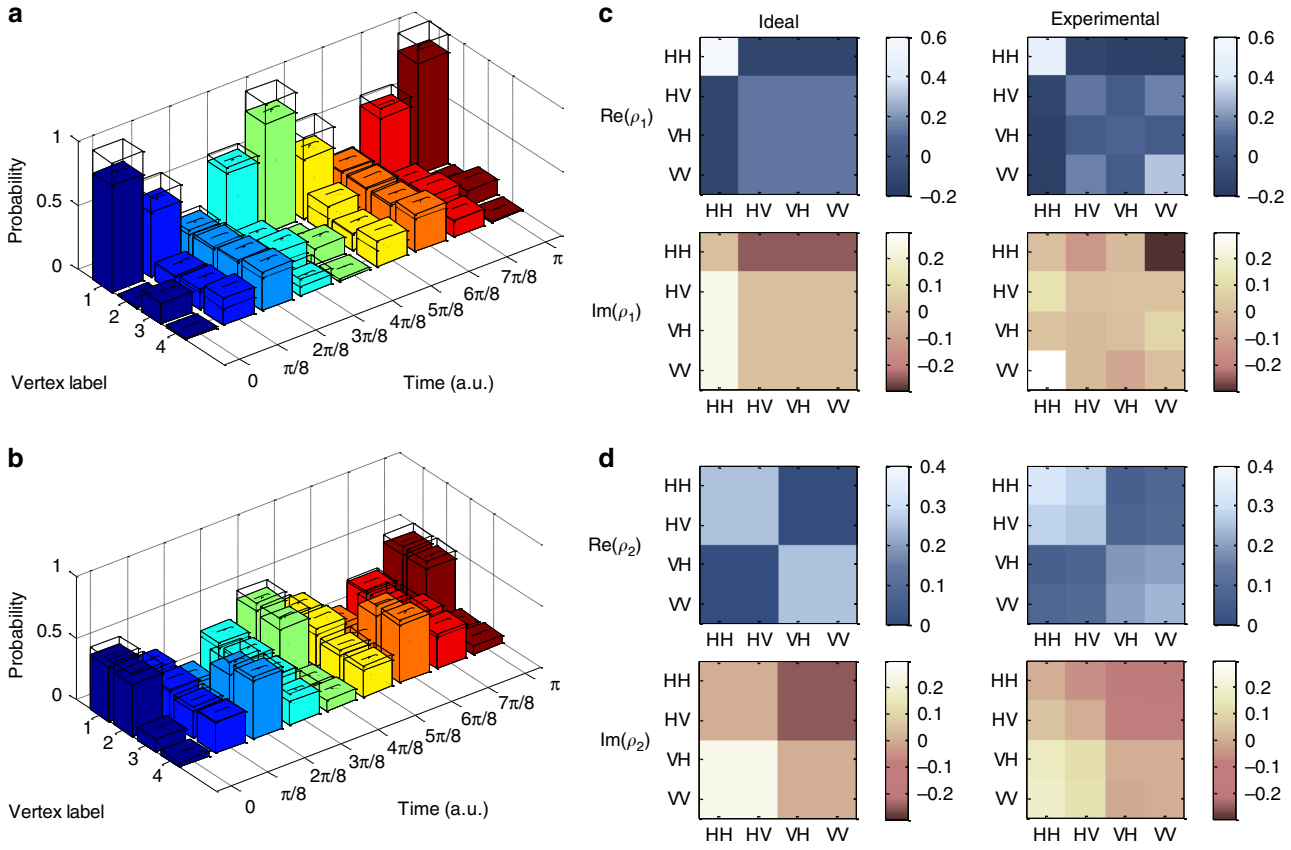


Figure 3 | Experimental results for simulating CTQWs on K_4. (**a**,**b**) The experimental sampled probability distributions with ideal theoretical distributions overlaid, for CTQWs on K_4 graph with initial states $|\varphi_{\rm ini}\rangle_1=[1,0,0,0]'$ and $|\varphi_{\rm ini}\rangle_2=\frac{1}{\sqrt{2}}[1,1,0,0]'$. The s.d. of each individual probability is also plotted, which is calculated by propagating error assuming Poissonian statistics. (**c**,**d**) The ideal theoretical and experimentally reconstructed density matrices for the states $|\varphi_{\rm out}\rangle_1=[0.75+0.25i,\ -0.25+0.25i,\ -0.25+0.25i,\ -0.25+0.25i]'$ (corresponding to ρ_1) and $|\varphi_{\rm out}\rangle_2=[0.3536+0.3536i, 0.3536+0.3536i,\ -0.3536+0.3536i,\ -0.3536+0.3536i]'$ (corresponding to ρ_2). Both of the real and imaginary parts of the density matrices are obtained through the maximum likelihood estimation technique, and is shown as Re(ρ) and Im(ρ), respectively. Further results are shown in Supplementary Table 1, Supplementary Fig. 2 and Supplementary Note 4.

periodical characteristics of CTQWs on K_4 graph. For each evolution, we sampled the corresponding probability distribution with fixed integration time, shown in Fig. 3a,b. To measure how close the experimental and ideal probability distributions are, we calculated the average fidelities defined as $F_{\rm average}=\frac{1}{9}\sum_{n=1}^{9}\sum_{i=1}^{4}\sqrt{P_{\rm ideal,n}(i)P_{\rm exp,n}(t)}$. The achieved average fidelities for the samplings with two distinct initial states are $96.68\pm0.27\%$ and $95.82\pm0.25\%$, respectively. Through the proposed circuit implementation, we are also able to examine the evolution states using quantum state tomography, which is generally difficult for the analogue simulations. For two specific evolution states $|\varphi_{\rm out}\rangle_1=\exp(-iH\frac{7}{8}\pi)|\varphi_{\rm ini}\rangle_1$ and $|\varphi_{\rm out}\rangle_2=\exp(-iH\frac{7}{8}\pi)|\varphi_{\rm ini}\rangle_2$, we performed quantum state tomography and reconstructed the density matrices using the maximum likelihood estimation technique. The two reconstructed density matrices achieve fidelities of $85.81\pm1.08\%$ and $88.44\pm0.97\%$, respectively, shown in Fig. 3c,d.

Here we have chosen to use K_4 in our experiment because it is simple enough to be implementable with state of the art photonics capability, while it provides an example to demonstrate our protocol for simulating CTQW on a circulant graph with controlled quantum logic. As the size of graph increases, the simplicity of K_N implies that CTQWs on this family of graphs can easily be simulated classically for arbitrary N—for CTQW on a complete graph of size N, an arbitrary output probability amplitude $\langle y|\exp(-itH)|x\rangle$ can be readily obtained as $(N-1+\exp(-itN\gamma))N^{-1}$ if $x=y$, and $(-1+\exp(-itN\gamma))N^{-1}$ otherwise, where $|x\rangle$ and $|y\rangle$ represent the initial state and evolution state, respectively. However, our outlined quantum circuit implementation (Fig. 1) extends to implement CTQW on far more complicated circulant graphs.

Hardness of the sampling problem. To provide evidence that simulating CTQW on general circulant graphs is likely to be hard classically, we consider a circuit of the form $Q^{\dagger}DQ$, where D is a diagonal matrix made up of poly(n) controlled-phase gates and Q is the quantum Fourier transform. Define $p_{\rm D}$ to be the probability of measuring all qubits to be 0 in the computational basis after $Q^{\dagger}DQ$ is applied to the input state $|0\rangle^{\otimes n}$. It is readily shown that

$$\begin{aligned} p_{\rm D} &= \left|\langle 0|^{\otimes n}Q^{\dagger}DQ|0\rangle^{\otimes n}\right|^2 \\ &= \left|\langle +|^{\otimes n}D|+\rangle^{\otimes n}\right|^2 \\ &= \left|\langle 0|^{\otimes n}H^{\otimes n}DH^{\otimes n}|0\rangle^{\otimes n}\right|^2. \end{aligned} \tag{3}$$

This implies that $p_{\rm D}$ can also be obtained through a circuit of form $H^{\otimes n}DH^{\otimes n}$ with D unchanged—this represents a class of circuits known as instantaneous quantum polynomial time (IQP), which has the following structure: each qubit line begins and ends with a Hadamard (H) gate, and, in between, every gate is diagonal

in the computational basis[37,42]. As such, p_D is a probability that is classically hard to compute—it is known that computing p_D for arbitrary diagonal unitaries D made up of circuits of poly(n) gates, even if each acts on $O(1)$ qubits, is #P-hard[38,43,44]. This hardness result even holds for approximating p_D up to any relative error strictly less than 1/2 (refs 38,43,44), where $\widetilde{p_D}$ is said to approximate p_D up to relative error ϵ if

$$\left|\widetilde{p_D} - p_D\right| \leq \epsilon p_D. \quad (4)$$

Note that other output probabilities $p(x)$ cannot be achieved using IQP circuits since a general circulant graph cannot be diagonalized by Hadamard matrices but rather by more heterogeneous Fourier matrices.

Towards a contradiction, assume that there exists a polynomial-time randomized classical algorithm, which samples from p, as defined in equation (1). Then a classic result of Stockmeyer[45] states that there is an algorithm in the complexity class $\mathrm{FBPP}^{\mathrm{NP}}$, which can approximate any desired probability $p(x)$ to within relative error $O(1/\mathrm{poly}(n))$. This complexity class $\mathrm{FBPP}^{\mathrm{NP}}$—described as polynomial-time randomized classical computation equipped with an oracle to solve arbitrary NP problems—sits within the infinite tower of complexity classes known as the polynomial(-time) hierarchy[46]. Combining with the above hardness result of approximating p_D, we find that the assumption implies that an $\mathrm{FBPP}^{\mathrm{NP}}$ algorithm solves a #P-hard problem, so $\mathrm{P}^{\#\mathrm{P}}$ would be contained within $\mathrm{FBPP}^{\mathrm{NP}}$, and therefore the polynomial hierarchy would collapse to its third level. This consequence is considered very unlikely in computational complexity theory[46]. A similar methodology has been used to prove the hardness of IQP and boson sampling[36–38].

We therefore conclude that, in general, a polynomial-time randomized classical sampler from the distribution p is unlikely to exist. Further, this even holds for classical algorithms which sample from any distribution $\widetilde{p}$ which approximates p up to relative error strictly $<1/2$ in each probability $p(x)$. It is worth noting that if the output distribution results from measurements on only $O(\mathrm{poly}(\log n))$ qubits[47], or obeys the sparsity promise that only a poly(n)-sized, and *a priori* unknown, subset of the measurement probabilities are non-zero[48], it could be classically efficiently sampled. It was shown in ref. 38 that assuming certain conjectures in complexity theory, it is classically hard to sample from distributions that are close in total variation distance to arbitrary IQP probability distributions. The differences between circulant and IQP circuits imply that this result does not go through immediately in our setting. Therefore, it remains open to prove hardness of approximate simulation of CTQWs on circulant graphs, which specifically requires to show that computing most of the output probabilities of circulant circuits is hard, assuming some conjectures in complexity theory.

Discussion

In this paper, we have described how CTQWs on circulant graphs can be efficiently implemented on a quantum computer, if the eigenvalues of the graphs can be characterized efficiently classically. In fact, we can construct an efficient quantum circuit to implement CTQWs on any graph whose adjacency matrix is efficiently diagonalisable, in other words, as long as the matrix of column eigenvectors Q and the diagonal matrix of the eigenvalue exponentials D can be implemented efficiently. To demonstrate our implementation scheme, we simulated CTQWs on an example 4-vertex circulant graph, K_4, using a two-qubit photonic quantum logic circuit. We have shown that the problem of sampling from the output probability distributions of quantum circuits of the form $Q^{\dagger}DQ$ is hard for classical computers, based on a highly plausible conjecture that the polynomial hierarchy does not collapse. This observation is particularly interesting from both perspectives of CTQW and computational complexity theory, as it provides new insights into the CTQW framework and also helps to classify and identify new problems in computational complexity theory. For the CTQWs on the circulant graphs of poly(n) non-zero eigenvalues, the proposed quantum circuit implementations do not need a fully universal quantum computer, and thus can be viewed as an intermediate model of quantum computation. Meanwhile, the evidence we provided for hardness of the sampling problem indicates a promising candidate for experimentally establishing quantum supremacy over classical computers, and further evidence against the extended Church–Turing thesis. To claim in an experiment super-classical performance based on the conjecture outlined in this work, future demonstrations would need to consider circulant graphs that are more general than K_N and that are of sufficient size to be outside the capabilities of a classical computer. For photonics, the biggest challenges remain increasing the number of indistinguishable photons and controlled gate operations. For any platform, quantum circuit implementation of CTQWs could be more appealing due to available methods in fault tolerance and error correction, which are difficult to implement for other intermediate models like boson sampling[49] and for analogue quantum simulation. Our results may also lead to other practical applications through the use of CTQWs for quantum algorithm design.

Methods

Experimental set-up. A diagonally polarized, 120 mW, continuous-wave laser beam with central wavelength of 404 nm is focused at the centre of paired type-I BiBO crystals with their optical axes orthogonally aligned to each other, to create the polarization entangled photon-pairs[50]. Through the spontaneous parametric downconversion process, the photon pairs are generated in the state of $\frac{1}{\sqrt{2}}(|H_1H_2\rangle + |V_1V_2\rangle)$, where H and V represent horizontal and vertical polarization, respectively. The photons pass through the polarization beam-splitter (PBS) part of the dual PBS/beam-splitter cubes on both arms to generate two-photon four-mode state of the form $\frac{1}{\sqrt{2}}(|H_{1b}H_{2b}\rangle + |V_{1r}V_{2r}\rangle)$ (where r and b labels the red and blue paths shown in Fig. 2c, respectively). Rotations T_1 and T_2 on each path, consisting of half wave-plate (HWP) and quarter wave plate (QWP), convert the state into $\frac{1}{\sqrt{2}}(|\phi_{1b}\phi_{2b}\rangle + |\phi_{1r}\phi_{2r}\rangle)$, where $|\phi_1\rangle$ and $|\phi_2\rangle$ can be arbitrary single-qubit states. The four spatial modes 1b, 2b, 1r and 2r pass through four single-qubit quantum gates P_1, P_2, Q_1 and Q_2, respectively, where each of the four gates is implemented through three wave plates: QWP, HWP and QWP. The spatial modes 1b and 1r (2b and 2r) are then mixed on the beam-splitter part of the cube. By post-selecting the case where the two photons exit at ports 1 and 2, we obtain the state $(P_1 \otimes P_2 + Q_1 \otimes Q_2)|\phi_1\phi_2\rangle$. In this way, we implement a two-qubit quantum operation of the form $P_1 \otimes P_2 + Q_1 \otimes Q_2$ on the initialized state $|\phi_1\phi_2\rangle$.

As shown in Fig. 2b, the quantum circuit for implementing CTQW on the K_4 graph consists of Hadamard gates (H), Pauli-X gates (X) and controlled-phase gate (CP). CP is implemented by configuring $P_1 = |H\rangle\langle H|$, $P_2 = I$, $Q_1 = |V\rangle\langle V|$, $Q_2 = R(= [1, 0; 0, e^{-i4\gamma t}])$, where P_1 and Q_1 are implemented by polarizers. Altogether with combining the operation $(\mathrm{H}\cdot\mathrm{X}) \otimes (\mathrm{H}\cdot\mathrm{X})$ before CP with state preparation and the operation $(\mathrm{X}\cdot\mathrm{H}) \otimes \mathrm{X}\cdot\mathrm{H}$ after CP with measurement setting, we implement the whole-quantum circuit on the experimental set-up. The evolution time of CTQW is controlled by the phase value of R, which is determined by setting the three wave plates of Q_2 in Fig. 2c to $\mathrm{QWP}(\frac{\pi}{4})$, $\mathrm{HWP}(\omega)$, $\mathrm{QWP}(\frac{\pi}{4})$, where the angle ω of HWP equals to the phase of R: -4γ t. The evolution time t is then given by $t = -\omega/(4\gamma)$.

References

1. Farhi, E. & Gutmann, S. Quantum computation and decision trees. *Phys. Rev. A* **58,** 915 (1998).
2. Kempe, J. Quantum random walks: an introductory overview. *Contemp. Phys.* **44,** 307–327 (2003).
3. Childs, A. M., Gosset, D. & Webb, Z. Universal computation by multiparticle quantum walk. *Science* **339,** 791–794 (2013).
4. Childs, A. M. & Goldstone, J. Spatial search by quantum walk. *Phys. Rev. A* **70,** 022314 (2004).
5. Douglas, B. L. & Wang, J. B. A classical approach to the graph isomorphism problem using quantum walks. *J. Phys. A* **41,** 075303 (2008).

6. Gamble, J. K., Friesen, M., Zhou, D., Joynt, R. & Coppersmith, S. N. Two-particle quantum walks applied to the graph isomorphism problem. *Phys. Rev. A* **81,** 052313 (2010).
7. Berry, S. D. & Wang, J. B. Two-particle quantum walks: entanglement and graph isomorphism testing. *Phys. Rev. A* **83,** 042317 (2011).
8. Berry, S. D. & Wang, J. B. Quantum-walk-based search and centrality. *Phys. Rev. A* **82,** 042333 (2010).
9. Sánchez-Burillo, E., Duch, J., Gómez-Gardeñes, J. & Zueco, D. Quantum navigation and ranking in complex networks. *Sci. Rep.* **2,** 605 (2012).
10. Lloyd, S. Universal quantum simulators. *Science* **273,** 1073–1078 (1996).
11. Berry, D. W. & Childs, A. M. Black-box hamiltonian simulation and unitary implementation. *Quantum Inf. Comput.* **12,** 29–62 (2012).
12. Schreiber, A. *et al.* A 2D quantum walk simulation of two-particle dynamics. *Science* **336,** 55–58 (2012).
13. Engel, G. S. *et al.* Evidence for wavelike energy transfer through quantum coherence in photosynthetic systems. *Nature* **446,** 782–786 (2007).
14. Rebentrost, P. *et al.* Environment-assisted quantum transport. *New J. Phys.* **11,** 033003 (2009).
15. Du, J. *et al.* Experimental implementation of the quantum random-walk algorithm. *Phys. Rev. A* **67,** 042316 (2003).
16. Ryan, C. A. *et al.* Experimental implementation of a discrete-time quantum random walk on an NMR quantum-information processor. *Phys. Rev. A* **72,** 062317 (2005).
17. Do, B. *et al.* Experimental realization of a quantum quincunx by use of linear optical elements. *J. Opt. Soc. Am. B* **22,** 499–504 (2005).
18. Schreiber, A. *et al.* Photons walking the line: a quantum walk with adjustable coin operations. *Phys. Rev. Lett.* **104,** 050502 (2010).
19. Xue, P., Sanders, B. C. & Leibfried, D. Quantum walk on a line for a trapped ion. *Phys. Rev. Lett.* **103,** 183602 (2009).
20. Schmitz, H. *et al.* Quantum walk of a trapped ion in phase space. *Phys. Rev. Lett.* **103,** 090504 (2009).
21. Zähringer, F. *et al.* Realization of a quantum walk with one and two trapped ions. *Phys. Rev. Lett.* **104,** 100503 (2010).
22. Karski, M. *et al.* Quantum walk in position space with single optically trapped atoms. *Science* **325,** 174–177 (2009).
23. Perets, H. B. *et al.* Realization of quantum walks with negligible decoherence in waveguide lattices. *Phys. Rev. Lett.* **100,** 170506 (2008).
24. Carolan, J. *et al.* On the experimental verification of quantum complexity in linear optics. *Nat. Photon.* **8,** 621 (2014).
25. Manouchehri, K. & Wang, J. B. *Physical Implementation of Quantum Walks* (Springer-Verlag, 2014).
26. Aharonov, D. & Ta-Shma, A. in *Proceedings of the 35th Annual ACM Symposium on Theory of Computing* 20–29 (ACM, New York, NY, USA, 2003).
27. Berry, D. W., Ahokas, G., Cleve, R. & Sanders, B. C. Efficient quantum algorithms for simulating sparse hamiltonians. *Commun. Math. Phys.* **270,** 359–371 (2007).
28. Childs, A. M. & Kothari, R. Limitations on the simulation of non-sparse hamiltonians. *Quantum Inf. Comput.* **10,** 669–684 (2009).
29. Childs, A. M. On the relationship between continuous-and discrete-time quantum walk. *Commun. Math. Phys.* **294,** 581–603 (2010).
30. Ng, M. K. *Iterative Methods for Toeplitz Systems* (Oxford Univ. Press, 2004).
31. Gray, R. M. *Toeplitz and Circulant Matrices: A Review* (Now Publishers Inc., 2006).
32. Buhrman, H., Cleve, R., Watrous, J. & De Wolf, R. Quantum fingerprinting. *Phys. Rev. Lett.* **87,** 167902 (2001).
33. Han, D. *et al.* Folding and cutting DNA into reconfigurable topological nanostructures. *Nat. Nanotechnol.* **5,** 712–717 (2010).
34. Peres, A. Stability of quantum motion in chaotic and regular systems. *Phys. Rev. A* **30,** 1610 (1984).
35. Prosen, T. & Znidaric, M. Stability of quantum motion and correlation decay. *J. Phys. A Math. Gen.* **35,** 1455 (2002).
36. Aaronson, S. & Arkhipov, A. in *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing* 333-342 (ACM Press, New York, NY, USA, 2011).
37. Bremner, M. J., Jozsa, R. & Shepherd, D. J. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proc. R. Soc. A Math. Phys. Engineering Science* **467,** 459–472 (2010).
38. Bremner, M. J., Montanaro, A. & Shepherd, D. J. Average-case complexity versus approximate simulation of commuting quantum computations. Preprint at http://arxiv.org/abs/1504.07999 (2015).
39. Nielsen, N. A. & Chuang, I. L. *Quantum Computation and Quantum Information* (Cambridge Univ. Press, 2010).
40. Childs, A. M. *Quantum Information Processing in Continuous Time* (PhD thesis, Massachusetts Institute of Technology, 2004).
41. Zhou, X. Q. *et al.* Adding control to arbitrary unknown quantum operations. *Nat. Commun.* **2,** 413 (2011).
42. Shepherd, D. & Bremner, M. J. Temporally unstructured quantum computation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* **465,** 1413–1439 (2009).
43. Fujii, K. & Morimae, T. Quantum commuting circuits and complexity of Ising partition functions. Preprint at http://arxiv.org/abs/1311.2128 (2013).
44. Goldberg, L. A. & Guo, H. The complexity of approximating complex-valued Ising and Tutte partition functions. Preprint at http://arxiv.org/abs/1409.5627 (2014).
45. Stockmeyer, L. J. On approximation algorithms for #P. *SIAM J. Comput.* **14,** 849–861 (1985).
46. Papadimitriou, C. *Computational Complexity* (Addison-Wesley, 1994).
47. Nest, M. Simulating quantum computers with probabilistic methods. *Quantum Inf. Comput.* **11,** 784–812 (2011).
48. Schwarz, M. & Nest, M. Simulating quantum circuits with sparse output distributions. Preprint at http://arxiv.org/abs/1310.6749 (2013).
49. Rohde, P. P. & Ralph, T. C. Error tolerance of the boson-sampling model for linear optics quantum computing. *Phys. Rev. A* **85,** 022332 (2012).
50. Rangarajan, R., Goggin, M. & Kwiat, P. Optimizing type-I polarization-entangled photons. *Opt. Express* **17,** 18920–18933 (2009).

Acknowledgements

We thank Anthony Laing and Peter J. Shadbolt for helpful discussions. We acknowledge support from the Engineering and Physical Sciences Research Council (EPSRC), the European Research Council (ERC)—including BBOI, QUCHIP (H2020-FETPROACT-3-2014: Quantum simulation) and PIQUE (FP7-PEOPLE-2013-ITN)—the Centre for Nanoscience and Quantum Information (NSQI), the US Army Research Office (ARO) grant W911NF-14-1-0133. X.Q. acknowledges support from University of Bristol and China Scholarship Council. T.L. is supported by an International Postgraduate Research Scholarship, Australian Postgraduate Award and the Bruce and Betty Green Postgraduate Research Top-Up Scholarship. J.L.O.B. acknowledges a Royal Society Wolfson Merit Award and a Royal Academy of Engineering Chair in Emerging Technologies. J.B.W. acknowledges the Benjamin Meaker visiting professorship provided by IAS at University of Bristol. J.C.F.M. was supported by a Leverhulme Trust Early Career Fellowship and an EPSRC Early Career Fellowship.

Author contributions

X.Q., T.L., J.L.O.B., J.B.W. and J.C.F.M. conceived and designed the project. T.L. and J.B.W. proposed the circuit construction scheme. X.Q., K.A. and X.Z. built the experiment set-up. X.Q. performed the experiments and analysed the data. A.M. provided the complexity theory proofs. X.Q., T.L., A.M., J.L.O.B., J.B.W. and J.C.F.M. wrote the manuscript. J.L.O.B., J.B.W. and J.C.F.M. supervised the project.

Additional information

Supplementary Information accompanies this paper at http://www.nature.com/naturecommunications

Competing financial interests: The authors declare no competing financial interests.

Reprints and permission information is available online at http://npg.nature.com/reprintsandpermissions/

How to cite this article: Qiang, X. *et al.* Efficient quantum walk on a quantum processor. *Nat. Commun.* 7:11511 doi: 10.1038/ncomms11511 (2016).

Computer Physics Communications 207 (2016) 531–532

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

OptQC v1.3: An (updated) optimized parallel quantum compiler

T. Loke, J.B. Wang*

School of Physics, The University of Western Australia, Crawley, WA 6009, Australia

ARTICLE INFO

Article history:
Received 25 May 2016
Accepted 26 May 2016
Available online 16 June 2016

Keywords:
Quantum circuit
Quantum compiler
Optimization

ABSTRACT

We present a revised version of the *OptQC* program of Loke et al. (2014) [1]. We have removed the simulated annealing process in favour of a descending random walk. We have also introduced a new method for iteratively generating permutation matrices during the random walk process, providing a reduced total cost for implementing the quantum circuit. Lastly, we have also added a synchronization mechanism between threads, giving quicker convergence to more optimal solutions.

New version program summary

Program title: OptQC v1.3

Catalogue identifier: AEUA_v1_3

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEUA_v1_3.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html

No. of lines in distributed program, including test data, etc.: 240903

No. of bytes in distributed program, including test data, etc.: 632395

Distribution format: tar.gz

Programming language: Fortran, MPI.

Computer: Any computer with Fortran compiler (not gfortran4.9 or earlier) and MPI library.

Operating system: Linux.

Classification: 4.15.

Catalogue identifier of previous version: AEUA_v1_3

Journal reference of previous version: Comput. Phys. Comm. 185(2014)3307

External routines: Intel MKL LAPACK routines and MPI routines.

Does the new version supersede the previous version?: Yes

Nature of problem:
It aims to minimize the number of quantum gates required to implement a given unitary operation.

Solution method:
It utilizes a descending random walk to select permutation matrices P and Q for a given unitary matrix U such that the number of gates in the quantum circuit of $U = Q^T P^T U' P Q$ is minimized, where U' is equivalent to U up to a permutation. The decomposition of a unitary operator is performed by recursively applying the cosine–sine decomposition.

Reasons for new version:
Simulated annealing process was found to give suboptimal results compared to a normal descending random walk. Computation time was also bloated by the necessity of running the CS decomposition thrice (for U', P and P^T) for each iteration of the optimization process.

* Corresponding author.
E-mail address: jingbo.wang@uwa.edu.au (J.B. Wang).

http://dx.doi.org/10.1016/j.cpc.2016.05.028

Summary of revisions:

- Simulated annealing process was replaced by a descending random walk (equivalent to setting the threshold value β to 0), due to poor convergence of the simulated annealing process, and due to the lack of pathological instances of local minima in this particular search space.
- Introduced an iterative method for generating the general permutation matrix P, in which we start with $P = I$ and build up the quantum circuit (and modify P correspondingly) by adding gates onto the existing circuit. This removes the requirement of running the CS decomposition on P and P^T to find the quantum circuit implementation, since it is already known by construction.
- Added a synchronization mechanism between threads (after some prescribed number of iterations) in which the current state of the top 10% processes with the fittest solutions is copied over to the remaining 90%. This works so as to discard the less fit solutions and focuses the searching algorithm in the state space with the fittest solutions.

Additional comments:

The program contains some Fortran2003 features and will not compile with gcc4.9 or earlier.

Running time:

As before, running time increases with the size of the unitary matrix, as well as the prescribed maximum number of iterations for qubit permutation selection and the descending random walk. All simulation results presented in this paper are obtained from running the program on the Fornax supercomputer managed by iVEC@UWA with Intel Xeon X5650 CPUs. A comparison of running times is also given in Table 1.

Table 1
Result summary for the decomposition of various matrices: m is the dimension of the given unitary matrix U, N_0 denotes the number of gates required to implement U, and $N_{\min}$ is the total number of gates obtained after the optimization process (post-reduction).

Matrix *	m *	N_0 *	Before $N_{\min}$	Update CPU (min)	After $N_{\min}$	Update CPU (min)
Random real unitary	8	29	22	0.333	18	0.250
S_8graph	16	34	23	1.083	21	0.467
3rd generation 3-Cayley tree	42	996	300	13.583	216	8.233
Quantum Fourier transform	64	4095	3508	20.95	3144	13.250
Shor's algorithm 8	128	8285	5621	75.700	4085	42.917

References:

[1] T. Loke, J.B. Wang, Y.H. Chen, Comput. Phys. Comm. 185 (2014) 3307.

IOP Publishing Journal of Physics A: Mathematical and Theoretical

J. Phys. A: Math. Theor. **50** (2017) 055303 (11pp) doi:10.1088/1751-8121/aa53a9

Efficient quantum circuits for continuous-time quantum walks on composite graphs

T Loke and J B Wang

School of Physics, The University of Western Australia, 6009 Perth, Australia

E-mail: jingbo.wang@uwa.edu.au

Received 26 June 2016, revised 7 December 2016
Accepted for publication 13 December 2016
Published 6 January 2017

Abstract

In this paper, we investigate the simulation of continuous-time quantum walks on specific classes of graphs, for which it is possible to fast-forward the time-evolution operator to achieve constant-time simulation complexity and to perform the simulation exactly, i.e. $\epsilon = 0$, while maintaining $\mathrm{poly}(\log(n))$ efficiency. In particular, we discuss two classes of composite graphs, commuting graphs and Cartesian product of graphs, that contain classes of graphs which can be simulated in this fashion. This allows us to identify new families of graphs that we can efficiently simulate in a quantum circuit framework, providing practical and explicit means to explore quantum-walk based algorithms in laboratories.

Keywords: quantum walk, quantum circuit, composite graph

(Some figures may appear in colour only in the online journal)

1. Introduction

Quantum walks are currently a subject of intense theoretical and experimental investigation due to its established role in quantum computation and quantum simulation [1–5]. In fact, any dynamical simulation of a Hamiltonian system in quantum physics and quantum chemistry can be discretized and mapped onto a continuous-time quantum walk on specific graphs [6–8]. The primary difficulty of such a numerical simulation lies in the exponential scaling of the Hilbert space as a function of the system size, making real-world size problems intractable on classical computers. However, in order to run quantum walk-based algorithms on a quantum computer, we require an efficient quantum circuit that implements the required quantum walk. Some examples of quantum walk-based algorithms are searching for a marked element on a graph [9–11], determining the relative importance of nodes in a graph [12, 13], and testing graph isomorphism [14].

There are two distinct types of quantum walks: the continuous-time quantum walk (CTQW) [1] and the discrete-time quantum walk (DTQW) [15]. In previous studies, DTQWs on several

classes of graphs have been shown to be efficiently implementable in terms of a sequence of quantum logic gate operations [16–19]. The implementation of the DTQW time-evolution operator $U(t) = (S \cdot C)^t$ for any $t \in \mathbb{R}_+$ is simplified by the fact that the time t is discrete and the evolution is repetitive. Here S and C are the shift and coin operator, respectively. As a result, if we can implement a single time-step $U = S \cdot C$ in a quantum circuit, the implementation for $U(t)$ can be generated by repeating the same circuit t times. Another property of DTQWs that is exploited in quantum circuit design is that the single time-step operator U acts locally on the vertex-coin states encoding the graph. In other words, applying U to the vertex-coin states associated with a particular vertex will only propagate the corresponding amplitudes to adjacent vertices, so vertices that are a distance of two or more apart do not affect each other in the single time-step. This means that the local structure of the graph is the primary consideration in implementing DTQWs. Taking these into account, quantum circuits to implement DTQWs on sparse graphs [17, 18], graphs with a high degree of symmetry, such as vertex-transitive graphs [16], and certain types of non-degree-regular graphs [19] have been constructed.

However for CTQWs, the above two properties of DTQWs do not carry over, which makes the design of quantum circuits substantially more difficult. First, the time-evolution operator $U(t)$ for CTQW is defined as a continuous function of $t \in \mathbb{R}_+$, requiring a time-dependent quantum circuit implementation. Second, the CTQW does not act locally on vertices—any two even distantly connected vertices on a graph will propagate amplitudes to each other in a CTQW. Consequently, the global structure of a graph must be taken into account in designing quantum circuits to implement CTQWs.

A substantial amount of work has been done on implementing CTQWs efficiently on quantum computers, typically considered under the more general problem of Hamiltonian simulation. Two classes of Hamiltonians are considered separately: sparse Hamiltonians [20–25] and dense Hamiltonians [22, 26]. Simulation of an n-dimensional Hamiltonian H is said to be efficient if there exists a quantum circuit using at most $\mathrm{poly}(\log(n), \|Ht\|, 1/\epsilon)$ one- and two- qubit gates (where $\mathrm{poly}(\ldots)$ denotes a polynomial scaling in terms of the listed parameters) that approximates the evolution operator $\exp(-\mathrm{i}tH)$ with error at most ε [22]. Here $\|\cdot\cdot\|$ denotes the spectral norm of a matrix.

Under this definition of efficiency, the complexity of quantum circuits would, in general, scale at least linearly with the time t. Given the periodic nature of a unitary system, this is undesirable although necessary in general, due to the no-fast-forwarding theorem [21]. As such, simulation of general sparse Hamiltonians in sublinear time is not possible (this result has been extended to non-sparse Hamiltonians as well in [26]). However, there still exists classes of Hamiltonians that can be simulated in sublinear (or even constant) time, i.e. Hamiltonians for which their time-evolution can be fast-forwarded, as pointed out in [26].

In this paper, we identify classes of graphs for which their Hamiltonian (given by the adjacency matrix) can be simulated efficiently in constant time (i.e. the complexity of the quantum circuit does not scale with the parameter t). We also focus on exact simulations of these graphs, that is, $\epsilon = 0$. Hence, in the rest of this paper, we term a quantum circuit implementation of a CTQW as efficient if it uses at most $\mathrm{poly}(\log(n))$ one- and two- qubit gates to implement the CTQW time-evolution operator $U(t) = \exp(-\mathrm{i}tH)$ of a graph on n vertices with $\epsilon = 0$, with the complexity being t-independent. This paper mirrors the efforts of previous studies in the DTQW literature to implement exactly the DTQW step operator for certain classes of graphs [16, 19], where it was found that only graphs that are highly symmetric are amenable to exact efficient implementations. This study is also motivated by the need for highly efficient quantum circuit implementations of CTQWs, since additional operations or a complete redesign of quantum circuits for different values of t would be prohibitively expensive given current experimental capabilities.

There are a select few classes of graphs for which their CTQW can be fast-forwarded to obtain an efficient quantum circuit implementation. In particular, it has been pointed out previously that the glued tree [20], complete graph, complete bipartite graph and star graph [27, 28] can be simulated efficiently using diagonalization. Recently, the class of circulant graphs (with some restrictions) have also been identified as being efficiently implementable [29], since circulant graphs can be diagonalized using the quantum Fourier transform.

The paper is organized as follows. In section 2, we review the background theory of CTQWs on graphs, and discuss diagonalization of matrices, which maps to quantum circuit implementations that have time-independent complexity. We discuss commuting graphs in section 3, which yield new graphs that can be efficiently implemented using known implementations of the subgraphs. In section 4, we discuss the Cartesian product of graphs, which further extends the classes of graphs we can efficiently implement. We then draw our conclusions in section 6.

2. Background theory

Consider a general undirected graph $G(V,E)$, with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E = \{(v_i, v_j), (v_k, v_l), \ldots\}$ being unordered pairs connecting the vertices. The n-by-n adjacency matrix A is defined as $A_{jk} = 1$, if $(v_j, v_k) \in E$ and 0 otherwise. For an undirected graph, its adjacency matrix A is symmetric. The degree d_i of a vertex v_i in an undirected graph is the number of undirected edges connected to v_i, which we denote by $\deg(A)_{v_i} = d_i$. A degree-regular graph with degree d is then a graph that has $\deg(A)_{v_i} = d \forall\, i = 1, \ldots, N$, i.e. every vertex has the same degree d.

The CTQW is described by a state vector $|\psi(t)\rangle$ in the Hilbert space of dimension n spanned by the orthonormal basis states $\{|1\rangle, |2\rangle, \ldots, |n\rangle\}$ corresponding to vertices in the graph. The time-evolution of the state $|\psi(t)\rangle$ is governed by the time-dependent Schrödinger equation

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = H|\psi(t)\rangle, \tag{1}$$

where the Hamiltonian H is a Hermitian operator, i.e. $H = H^\dagger$. The formal solution to this equation is $|\psi(t)\rangle = U(t)|\psi(0)\rangle$, where $U(t) = \exp(-\mathrm{i}tH)$ is the time-evolution operator. Choice of H varies in the literature between $H = \gamma(D - A)$ [2, 30] and $H = \gamma A$ [1, 31], where γ is the hopping rate per edge per unit time and D is an n-by-n (diagonal) degree matrix defined by $D_{ij} = d_i\delta_{ij}$, where δ_{ij} is the Kronecker delta. For degree-regular graphs, the only difference between the two choices is a global phase factor and a sign flip in t, which does not change observable quantities [32]. However, the two choices will result in different dynamics for non-degree-regular graphs [10]. In this paper, we use $H = \gamma A$.

It is well-known that for a Hermitian matrix H, the spectral theorem guarantees that H can be diagonalized using its eigenbasis, that is $H = Q^\dagger \Lambda Q$ [33]. Here Q is a unitary matrix whose column vectors are eigenvectors of H, and Λ is a diagonal matrix of eigenvalues of H, which are all real and whose order is determined by the order of the eigenvectors in Q. From this, we can express the time-evolution operator as $U(t) = Q^\dagger \exp(-\mathrm{i}t\Lambda)Q$.

The diagonalization approach confines the time-dependence of $U(t)$ to the diagonal matrix $\exp(-\mathrm{i}t\Lambda)$, which can be readily implemented by a sequence of at most n controlled-phase gates with phase values being linear functions of t. Experimentally, this corresponds to a sequence of tunable controlled-phase gates, where the phase values are determined by t.

Two quantum gates that we will use heavily in the following sections are the Hadamard gate:

$$\mathcal{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2}$$

and the general 2-phase rotation gate:

$$R(\theta_1, \theta_2) = \begin{pmatrix} e^{i\theta_1} & 0 \\ 0 & e^{i\theta_2} \end{pmatrix}. \tag{3}$$

In principle, a quantum circuit implementation for $U(t)$ that has time-independent complexity can always be obtained by using a general quantum compiler (as discussed in [34, 35]) to obtain a quantum circuit implementation for Q and $Q^\dagger$—however this is almost never efficient, since such methods typically scale exponentially in terms of complexity. In order to be able to implement $U(t)$ efficiently as a whole, we require that at most $\mathrm{poly}(\log(n))$ one- and two- qubit gates are used in implementing Q, $\exp(-it\Lambda)$ and $Q^\dagger$ individually (which is not always possible, as per the no-fast-forwarding theorem). In the following sections, we will cover some classes of graphs that satisfy this criteria.

3. Commuting graphs

Suppose we have two matrices H_1 and H_2. In general, when H_1 and H_2 do not commute, their sum can be simulated by the Lie product formula [36]

$$\exp(-it(H_1 + H_2)) = \lim_{m\to\infty} (\exp(-itH_1/m)\exp(-itH_2/m))^m \tag{4}$$

which, in practice, requires high-order approximations to achieve a bounded error that scales with t. However, in the case where H_1 and H_2 commute, we can write the expression exactly as

$$\exp(-it(H_1 + H_2)) = \exp(-itH_1)\exp(-itH_2). \tag{5}$$

Taking $H_1 = \gamma A$ and $H_2 = \gamma B$, where γ is constant, A and B are the adjacency matrices of two commuting graphs, i.e. $[A, B] = 0$. It follows that if the individual time-evolution operators $\exp(-it\gamma A)$ and $\exp(-it\gamma B)$ can be efficiently implemented, then the time-evolution operator for the graph $A + B$, that is, $\exp(-it\gamma(A + B))$, can also be efficiently implemented, provided $[A, B] = 0$.

The general criteria for commuting graphs is studied in [37]. One particular class of graphs is the interdependent networks, defined by

$$A = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \text{and } B = \begin{pmatrix} 0 & B_0 \\ B_0^T & 0 \end{pmatrix},$$

where the interlink graph B connects two subgraphs A_1 and A_2, which are both symmetric, i.e. $A_1 = A_1^T$ and $A_2 = A_2^T$. In this instance, the condition for commutativity becomes

$$A_1 B_0 = B_0 A_2. \tag{6}$$

Suppose Q_1 and Q_2 diagonalize A_1 and A_2 respectively, we have

$$\Lambda_1 = Q_1^\dagger A_1 Q_1 \text{ and } \Lambda_2 = Q_2^\dagger A_2 Q_2.$$

Then the following matrix

$$Q = \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix}$$

diagonalizes A, and give the eigenvalue matrix

$$\Lambda = \begin{pmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{pmatrix}.$$

Suppose B_0 is diagonalized by Q_0, i.e. $\zeta_0 = Q_0 B_0 Q_0^\dagger$, then it can be shown that if $B_0 = B_0^T$ (namely a symmetric interconnection), the diagonalizing matrix for B is

$$Q' = \frac{1}{\sqrt{2}} \begin{pmatrix} Q_0 & Q_0 \\ Q_0 & -Q_0 \end{pmatrix} = \mathcal{H} \otimes Q_0,$$

where $\mathcal{H}$ is the Hadamard matrix as defined above. The corresponding eigenvalue matrix

$$\zeta = \begin{pmatrix} \zeta_0 & 0 \\ 0 & -\zeta_0 \end{pmatrix} = \sigma_z \otimes \zeta_0,$$

where σ_z is the Pauli-z matrix. Hence, in the case where $[A, B] = 0$, we expand the CTQW time-evolution operator as

$$\exp(-\mathrm{i}t(H_1 + H_2)) = Q^\dagger \exp(-\mathrm{i}t\Lambda) Q Q'^\dagger \exp(-\mathrm{i}t\zeta) Q'. \tag{7}$$

3.1. Case 1: identity interconnections between two copies of a graph

Next, we examine some explicit examples of interdependent networks in which $[A, B] = 0$ is satisfied and equation (6) holds. One special case is that of identity interconnections between two disjoint copies of a graph with n vertices, namely $A_1 = A_2$ and $B_0 = I_n$. The diagonalizing matrices for A and B are, respectively

$$Q = \begin{pmatrix} Q_1 & 0 \\ 0 & Q_1 \end{pmatrix} = I_2 \otimes Q_1 \text{ and } Q' = \mathcal{H} \otimes I_n,$$

giving the eigenvalue matrices

$$\Lambda = I_2 \otimes \Lambda_1 \text{ and } \zeta = \sigma_z \otimes I_n.$$

Hence, if we are able to implement A_1 efficiently, it follows that the interdependent network with $A_1 = A_2$ and $B_0 = I_n$ can be implemented efficiently. An equivalent result can be achieved by noting that $A + B = \sigma_x \oplus A_1$, where σ_x is the Pauli-x matrix and $\oplus$ denotes the Cartesian product (defined in section 4), and then applying the methods of section 4. Figure 1 shows one class of graphs (complete graphs with identity interconnections) that can be constructed as above, together with its corresponding circuit implementation.

3.2. Case 2: complete interconnections between degree-regular graphs

Another case where $[A, B] = 0$ is using complete interconnections between two disjoint degree-regular graphs (with n_1 and n_2 vertices respectively) of same degree. That is, $\deg(A_1) = \deg(A_2) = d$ and $B_0 = J_{n_1,n_2}$, where J_{n_1,n_2} is the n_1-by-n_2 matrix with all 1's. Although in general $B_0 \neq B_0^T$, the interconnection matrix

$$B = \begin{pmatrix} 0 & J_{n_1,n_2} \\ J_{n_2,n_1} & 0 \end{pmatrix} \tag{8}$$

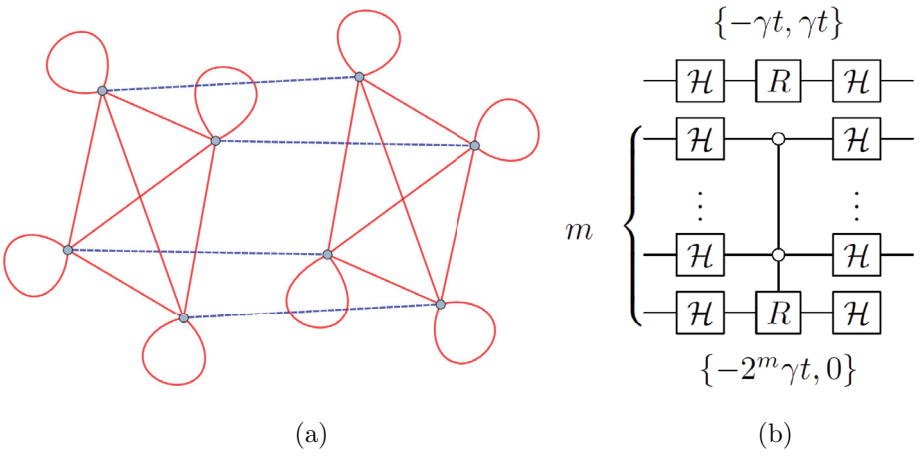


Figure 1. (a) Two disjoint K_{2^m} graphs with identity interconnections (solid red lines and dashed blue lines indicate edges belonging to A and B respectively), where $m = 2$; (b) the corresponding quantum circuit implementation for the CTQW time-evolution operator on this graph.

can still be diagonalized easily in the case where $n_1 = 2^{m_1}$ and $n_2 = 2^{m_2}$, for non-negative integers m_1 and m_2. For convenience, we assume $n_1 \geqslant n_2$, and note that B is the complete bipartite graph K_{n_1,n_2}. The diagonalization operator for B can be written mathematically as

$$\begin{aligned} Q' = (I_{2^{m_1+1}} + (H - I_2) \otimes P_0^{\otimes m_1})(I_{2^{m_1+1}} + P_0 \otimes (H^{\otimes m_1} - I_{2^{m_1}}) \\ + P_1 \otimes P_0^{\otimes (m_1-m_2)} \otimes (H^{\otimes m_2} - I_{2^{m_2}})), \end{aligned} \tag{9}$$

where $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$ are the 2-dimensional projection operators. The corresponding eigenvalue matrix of B is then given by

$$\zeta = \mathrm{diag}\left(\left\{(+\sqrt{n_1 n_2})^1, 0^{n_1-1}, (-\sqrt{n_1 n_2})^1, 0^{n_2-1}\right\}\right). \tag{10}$$

Figure 2 shows the complete bipartite graph K_{n_1,n_2} together with its corresponding quantum circuit implementation. As a corollary, the star graph S_{2^m+1} can also be implemented using the same method, since S_{2^m+1} is equivalent to $K_{2^m,1}$.

Hence, if we have degree-regular graphs A_1 and A_2 satisfying

$$\deg(A_1)_v = \deg(A_2)_v = d \forall\, v \in V, \tag{11}$$

where $n_1 = 2^{m_1}$ and $n_2 = 2^{m_2}$, which can both be efficiently implemented, then it follows that the interdependent network built from A_1, A_2 and $B_0 = J_{n_1,n_2}$ can be efficiently implemented. Figure 3(a) gives an example of this kind of graphs, where vertices 1–16 belong to the Q_4 graph (hypercube graph of dimension 4—refer to section 4), and 17–24 belong to the $K_{4,4}$ graph. The quantum circuit implementation of the composite graph shown in figure 3(a) is given by figure 3(b), where the $K_{16,8}$ circuit is already described above and given by figure 2(b). Note that in general, the K_{n_1,n_2} graph and by extension, the resulting interdependent network with complete interconnections is not degree-regular - so this provides an example of a class of graphs that is not degree-regular but still has an efficient quantum circuit implementation for the CTQW time-evolution operator.

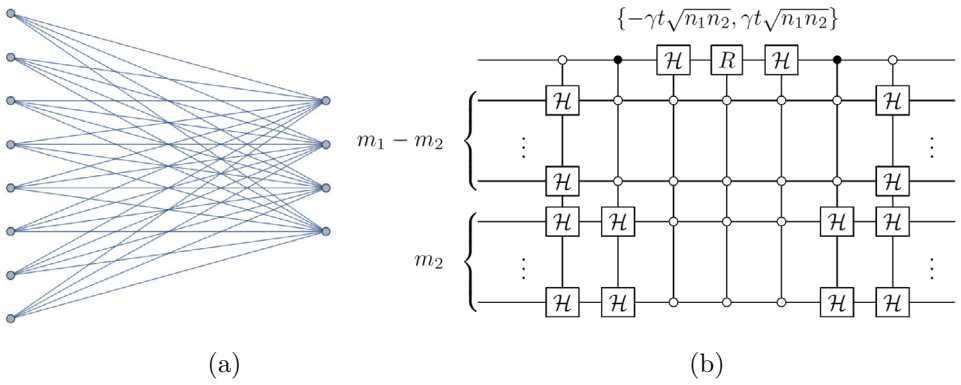


Figure 2. (a) An example of the complete bipartite graph K_{n_1,n_2}, where $n_1 = 8$ and $n_2 = 4$; (b) quantum circuit implementation of the CTQW time-evolution operator for the complete bipartite graph K_{n_1,n_2}, where $n_1 = 2^{m_1}$ and $n_2 = 2^{m_2}$.

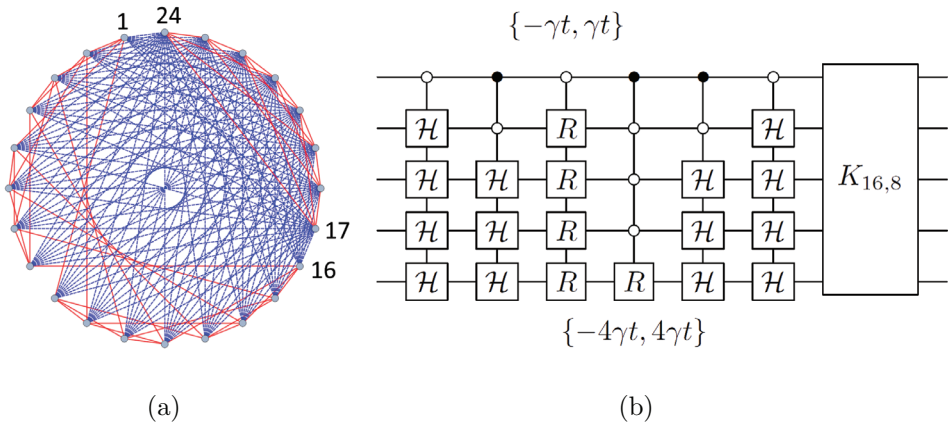


Figure 3. (a) Disjoint Q_4 and $K_{4,4}$ with complete interconnections (solid red lines and dashed blue lines indicate edges belonging to A and B respectively); (b) its corresponding quantum circuit implementation.

4. Cartesian product of graphs

Given two matrices H_1 and H_2 of dimension n_1-by-n_1 and n_2-by-n_2 respectively, the Cartesian product of H_1 and H_2 are given by

$$H_1 \oplus H_2 = H_1 \otimes I_{n_2} + I_{n_1} \otimes H_2, \tag{12}$$

which is a matrix of dimension n_1n_2-by-n_1n_2. In particular, if we define $H = H_1 \oplus H_2$, we have

$$\exp(-\mathrm{i}tH) = \exp(-\mathrm{i}tH_1) \otimes \exp(-\mathrm{i}tH_2) \tag{13}$$

or, more compactly, $U(t) = U_1(t) \otimes U_2(t)$. Again we set $H_1 = \gamma A$ and $H_2 = \gamma B$, i.e. the Hamiltonians H_1 and H_2 correspond to graphs A and B respectively. The physical significance of this composition is that the quantum walks on the two graphs A and B are performed simultaneously and separately, with no interaction between the two walks. This implies that if we have an efficient quantum circuit implementation for the individual CTQW time-evolution

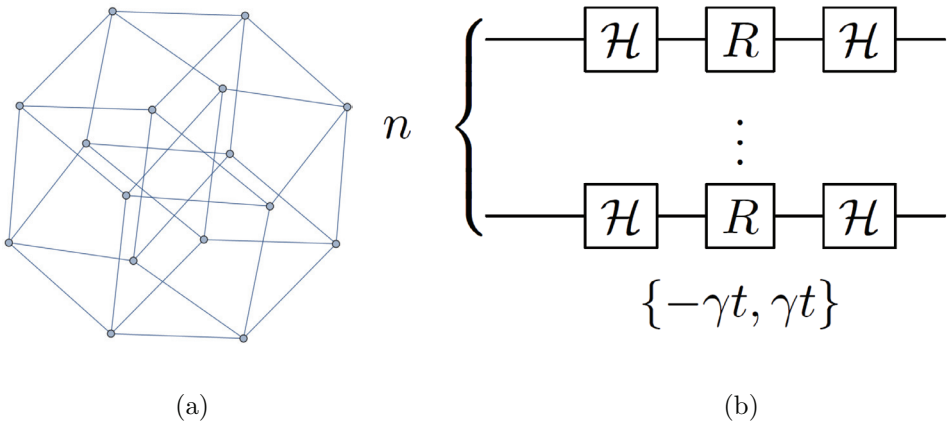


Figure 4. (a) The hypercube graph Q_n, where $n = 4$; (b) its corresponding quantum circuit implementation for the CTQW time-evolution operator on Q_n.

operators on the graphs A and B, the implementation for $U(t)$ (which is the time-evolution operator that corresponds to the CTQW on the graph $A \oplus B$) is easily formed by stacking the individual quantum circuit implementations in parallel. As a corollary, in the case where $A = B$, then $H = \gamma A \oplus A$ corresponds to the Hamiltonian for the non-interacting two-particle quantum walk on A. In particular, the two-particle quantum walk has been applied to the graph isomorphism problem, as seen in [38, 39].

4.1. Example 1: hypercube Q_n

One particular class of graphs that is constructed using the Cartesian product of graphs is the hypercube Q_n. Given the path graph P_2 of length 2 with adjacency matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, Q_n is constructed as $Q_n = P_2^{\oplus n}$, i.e. it is the Cartesian product of n copies of P_2 [40]. As such, Q_n is a graph with 2^n vertices and degree-regular with degree n. P_2 is diagonalizable using the Hadamard matrix $\mathcal{H}$, giving its eigenvalue matrix $\Lambda_{P_2} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. Figure 4 shows the hypercube graph Q_n with its corresponding quantum circuit implementation.

4.2. Example 2: book graph B_n

Another example of this class of graphs is the book graph B_n [41], which is constructed as $B_n = S_{n+1} \oplus P_2$, where S_{n+1} is the star graph on $n + 1$ vertices. As we have discussed in section 3, S_{n+1} can be implemented efficiently in a quantum circuit if $n = 2^m$ for some non-negative integer m—hence book graphs of the form B_{2^m} can be efficiently implemented as well, as shown in figure 5.

5. Discussion

As shown in sections 3 and 4, we can construct an efficient implementation of a composite graph using diagonalization, provided that the constituent graphs can themselves be efficiently diagonalized. While we have constructed examples based on constituent graphs that

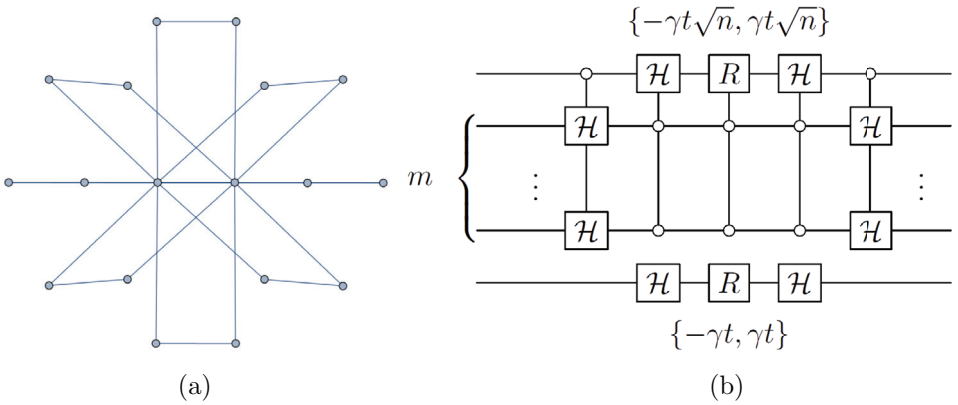


Figure 5. (a) The book graph B_n, where $n = 8$; (b) corresponding quantum circuit implementation for the CTQW time-evolution operator on the book graph for $n = 2^m$.

can be efficiently diagonalized, it remains to be seen in general what graphs can be efficiently diagonalized (which can then be used as constituent graphs for the above classes of composite graphs). That is, given that the time-evolution operator of a graph can be diagonalized in the form $U(t) = Q^\dagger \exp(-\mathrm{i}t\Lambda)Q$ where $H = Q^\dagger \Lambda Q$, in what circumstances can Q, $Q^\dagger$ and $\exp(-\mathrm{i}t\Lambda)$ each be implemented using at most $\mathrm{poly}(\log(n))$ one- and two- qubit gates?

Recall that Q is a unitary matrix whose column vectors are eigenvectors of H, and Λ is a diagonal matrix of eigenvalues of H. From [29], we find that $\exp(-\mathrm{i}t\Lambda)$ can be efficiently implemented in the case where there are at most $\mathrm{poly}(\log(n))$ non-zero eigenvalues. More generally, if the eigenvalue spectrum can be characterized efficiently such that some efficiently implementable function $f(x_i) = \lambda_i$ can generate the eigenvalue spectrum, then $\exp(-\mathrm{i}t\Lambda)$ can be efficiently implemented [29].

As for the implementation of Q (and $Q^\dagger$, which can be obtained by conjugating and reversing the order of gates in Q), it is much more difficult to draw concrete conditions on when it can be efficiently implemented. This is because the choice of matrix eigenvectors is not unique, and as such, there is a continuum of possible Q's that can be selected to diagonalize the Hamiltonian of the graph. For example, as noted in [29], the complete graph K_n with self-loops can be efficiently diagonalized using the QFT matrix $Q_{jk} = \frac{1}{\sqrt{n}}\exp(2\pi \mathrm{i}jk/n)$ or, in the case where $n = 2^m$, by the Hadamard basis $Q = \mathcal{H}^{\otimes m}$. There are, of course, infinitely many different eigenbases that can be selected to diagonalize the K_n graph with self-loops—but not all are efficiently implementable.

In order to help to identify situations where Q is efficiently implementable, we note that the availability of different bases to diagonalize the graph is connected to the degeneracy of the eigenspectrum. In the above example of the K_n graph with self-loops, the eigenspectrum of the graph is $\{n^1, 0^{n-1}\}$, i.e. the eigenvalue 0 is degenerate with algebraic multiplicity $n - 1$. Since the Hamiltonian is Hermitian, this implies that the Hamiltonian is never defective [42], i.e. the geometric and algebraic multiplicities of all eigenvalues are always equal—implying that any orthogonal basis of $n - 1$ vectors belonging to the eigenspace (i.e. the kernel $\ker(H)$) can be chosen as the corresponding eigenvectors. In more general terms, for an eigenvalue λ of the Hamiltonian with algebraic multiplicity m_λ, any orthogonal basis of m_λ vectors for the eigenspace $\ker(H - \lambda I)$ can be chosen as the eigenvectors for the eigenvalue λ. As such, degeneracy of the eigenspectrum can be exploited to select an eigenbasis that can be more

easily implemented in a quantum circuit, as demonstrated by the K_n example. From this, we state the following conjecture:

Conjecture 1. If the eigenspectrum of the Hamiltonian has more degeneracy (i.e. high algebraic multiplicities for eigenvalues), then it is more likely that the Hamiltonian can be efficiently diagonalized.

Another example of the application of this conjecture is to the complete bipartite graph K_{n_1,n_2}. This graph has the eigenvalue spectrum $\{(\pm\sqrt{n_1 n_2})^1, 0^{n_1+n_2-2}\}$, which is highly degenerate since the eigenvalue 0 has algebraic multiplicity $n_1 + n_2 - 2$. As such, according to the conjecture, it is likely that the Hamiltonian corresponding to the graph K_{n_1,n_2} can be efficiently implemented (its implementation was provided in figure 2). The star graph S_n is equivalent to the $K_{n,1}$ graph, and so is also highly degenerate.

6. Conclusion

In summary, we have shown that using diagonalization, we can fast-forward the simulation of some Hamiltonians corresponding to graphs in an efficient manner for some classes of composite graphs (commuting graphs and Cartesian products of graphs) to achieve constant-time complexity with $\epsilon = 0$. The quantum circuit implementations presented here are eminently useful in experimentally implementing CTQWs, since new classes of graphs can be simulated using existing graphs with minimal increase in complexity.

Acknowledgments

TL is supported by the International Postgraduate Research Scholarship, Australian Postgraduate Award and the Bruce and Betty Green Postgraduate Research Top-Up Scholarship.

References

[1] Farhi E and Gutmann S 1998 *Phys. Rev.* A **58** 915
[2] Kempe J 2003 *Contemp. Phys.* **44** 307
[3] Childs A M 2009 *Phys. Rev. Lett.* **102** 180501
[4] Childs A M, Gosset D and Webb Z 2013 *Science* **339** 791
[5] Manouchehri K and Wang J 2014 *Physical Implementation of Quantum Walks* (New York: Springer)
[6] Lloyd S 1996 *Science* **273** 1073
[7] Aharonov D and Ta-Shma A 2003 *Proc. of the 35th Annual ACM Symp. on Theory of Computing* (ACM) pp 20–9
[8] Berry D W and Childs A M 2012 *Quantum Inf. Comput.* **12** 29
[9] Shenvi N, Kempe J and Whaley K B 2003 *Phys. Rev.* A **67** 052307
[10] Childs A M and Goldstone J 2004 *Phys. Rev.* A **70** 022314
[11] Magniez F, Nayak A, Roland J and Santha M 2011 *SIAM J. Comput.* **40** 142
[12] Paparo G D and Martin-Delgado M A 2012 *Sci. Rep.* **2** 444
[13] Paparo G D, Mller M, Comellas F and Martin-Delgado M A 2013 *Sci. Rep.* **3** 2773
[14] Douglas B L and Wang J B 2008 *J. Phys. A: Math. Theor.* **41** 075303
[15] Aharonov Y, Davidovich L and Zagury N 1993 *Phys. Rev.* A **48** 1687
[16] Douglas B and Wang J 2009 *Phys. Rev.* A **79** 052335
[17] Jordan S P and Wocjan P 2009 *Phys. Rev.* A **80** 062301
[18] Chiang C-F, Nagaj D and Wocjan P 2010 *Quantum Inf. Comput.* **10** 420
[19] Loke T and Wang J B 2012 *Phys. Rev.* A **86** 042338

[20] Childs A M, Cleve R, Deotto E, Farhi E, Gutmann S and Spielman D A 2003 *Proc. of the 35th Annual ACM Symp. on Theory of Computing* (New York: ACM) pp 59–68
[21] Berry D W, Ahokas G, Cleve R and Sanders B C 2007 *Commun. Math. Phys.* **270** 359
[22] Kothari R 2010 Efficient simulation of Hamiltonians *PhD Thesis* University of Waterloo
[23] Childs A M and Kothari R 2011 *Theory of Quantum Computation, Communication, and Cryptography* (*Lecture Notes in Computer Science* vol 6519) ed W V Dam *et al* (Berlin: Springer) pp 94–103
[24] Wiebe N, Berry D W, Hyer P and Sanders B C 2011 *J. Phys. A: Math. Theor.* **44** 445308
[25] Berry D W, Childs A M and Kothari R 2015 *2015 IEEE 56th Annual Symp. on Foundations of Computer Science* pp 792–809
[26] Childs A M and Kothari R 2010 *Quantum Inf. Comput.* **10** 669
[27] Xu X-P 2009 *J. Phys. A: Math. Theor.* **42** 115205
[28] Childs A M 2010 *Commun. Math. Phys.* **294** 581
[29] Qiang X, Loke T, Montanaro A, Aungskunsiri K, Zhou X, OBrien J L, Wang J B and Matthews J C F 2016 *Nat. Commun.* **7** 11511
[30] Childs A M, Farhi E and Gutmann S 2002 *Quantum Inf. Process.* **1** 35
[31] Chakrabarti A, Lin C and Jha N K 2012 Design of quantum circuits for random walk algorithms *2012 IEEE Computer Society Annual Symp. on VLSI* pp 135–40
[32] Ahmadi A, Belk R, Tamon C and Wendler C 2003 *Quantum Information and Computation* **3** 611
[33] Hogben L (ed) 2006 *Handbook of Linear Algebra* (*Discrete Mathematics and Its Applications* vol 33) (London: Chapman and Hall)
[34] Chen Y G and Wang J B 2013 *Comput. Phys. Commun.* **184** 853
[35] Loke T, Wang J B and Chen Y H 2014 *Comput. Phys. Commun.* **185** 3307
[36] Childs A M 2004 Quantum information processing in continuous time *PhD Thesis* Massachusetts Institute of Technology
[37] Nehaniv C L, Dini P and Van Mieghem P 2014 Construction of Interdependent Networks and Efficiency Improvements in their Stability *Technical Report* (www.internet-science.eu/publication/1116)
[38] Gamble J K, Friesen M, Zhou D, Joynt R and Coppersmith S N 2010 *Phys. Rev.* A **81** 052313
[39] Berry S D and Wang J B 2011 *Phys. Rev.* A **83** 042317
[40] Harary F, Hayes J P and Wu H-J 1988 *Comput. Math. Appl.* **15** 277
[41] Gallian J A 1997 Graph labeling *Electron. J. Comb.*
[42] Golub G H and Loan C F V 1996 *Matrix Computations* 3rd edn (Baltimore: Johns Hopkins University Press)

Quantum Inf Process (2015) 14:3193–3210
DOI 10.1007/s11128-015-1047-4

Entanglement dynamics of two-particle quantum walks

G. R. Carson[1] · T. Loke[1] · J. B. Wang[1]

Received: 15 October 2014 / Accepted: 3 June 2015 / Published online: 20 June 2015

Abstract This paper explores the entanglement dynamics generated by interacting two-particle quantum walks on degree-regular and degree-irregular graphs. We performed spectral analysis of the time-evolution of both the particle probability distribution and the entanglement between the two particles for various interaction strength. While the particle probability distributions are stable and not sensitive to perturbations in the interaction strength, the entanglement dynamics are found to be much more sensitive to system variations. This property may be utilised to probe small differences in the system parameters.

Keywords Quantum walk · Entanglement dynamics · Two-particle interaction · Spectral analysis

1 Introduction

Entanglement dynamics has been extensively investigated in the context of quantum chaos and quantum-classical correspondence. Several nonlinear models have been examined for their entanglement dynamics, for example, the N-atom Jaynes-Cummings model [1–3], coupled kicked tops [4–7], the Dicke model [8], and Rydberg molecules [9,10]. Such studies have focused on the connection between the dynamics of classically chaotic systems with the dynamical generation of entanglement in their corresponding quantised systems.

In this work, we study the entanglement dynamics in two-particle discrete-time quantum walks on degree-regular and degree-irregular graphs. Single-particle quan-

✉ J. B. Wang
jingbo.wang@uwa.edu.au

[1] School of Physics, The University of Western Australia, Crawley, WA 6009, Australia

tum walks have already emerged as a useful tool in the development of quantum algorithms [11–18]. These algorithms depend on interference between the multiple paths that are simultaneously traversed by the quantum walker. In multi-particle quantum walks, the dimension of the state space increases exponentially with the number of particles and there is the additional resource of interaction and entanglement between particles, which may be utilised to develop efficient quantum algorithms to solve practically significant applications.

The theoretical study of discrete-time quantum walks with more than one particle was initiated by Omar et al. [19], who considered non-interacting two-particle quantum walks on the infinite line. Omar et al. established a role for entanglement in two-particle quantum walks by showing that initial states which are entangled in their coin degrees of freedom can generate two-particle probability distributions in which the positions of the two particles exhibit quantum correlations. Also with non-interacting particles, Štefaňák et al. [20] considered the meeting problem in the discrete-time quantum walk.

In addition to these studies with distinguishable particles, there have also been theoretical investigations of two-photon quantum walks [21,22]. By increasing the dimension of the coin Hilbert space to incorporate both the direction of photon propagation and polarisation, Pathak and Agarwal [21] introduced a quantum walk in which two photons initially in separable Fock states become entangled through the action of linear optical elements. While the quantum walk studied in [19,20] requires entangled initial states to generate spatial correlations, the two-photon walk studied in [21], with its larger coin space, is capable of generating entanglement even from initially separable states. Venegas-Andraca and Bose [23] have also proposed a variant of the two-particle quantum walk in which the particles have a shared coin space. In this system, entanglement is introduced between the spatial degrees of freedom of the particles by performing measurements in the shared coin space. More recently, Berry and Wang [24] introduced two spatial interaction schemes, which have shown to dynamically generate complex entanglement between the two walking particles. Rohde et. al. [25] studied the entanglement dynamics in a single-particle quantum walk between its position states on a finite line with bound and reflecting boundaries. They demonstrated the emergence of quasi-periodicity in the entanglement time series.

In this work, we explore the detailed entanglement dynamics generated by locally interacting two-particle quantum walks on degree-regular and degree-irregular graphs. Our key purpose is to demonstrate that entanglement dynamics arising from two-particle interacting quantum walks on graphs with simple underlying structures can be complex and sensitive to perturbations in the interaction strength parameter. This work provides strong evidence through sufficiently simple examples where the evolution of the system wavefunction is stable and not sensitive to perturbations to the interaction strength, while the entanglement dynamics of the system is much more sensitive to these small changes. Even in cases where the system wavefunction appears to evolve in a more complex manner, the marginal probability distribution is stable compared to the entanglement between the two particles. Both degree-regular and degree-irregular graphs are studied in this work. Degree-regular graphs are graphs for which every vertex has the same number of neighbours. We studies three simple degree-regular graphs, including the complete K_8 graph (without self-loops), the

3-dimensional hypercube Q_3, and the joined second-generation 3-Cayley tree. The degree-irregular graphs studied are variations of these degree-regular graphs, with several edges removed.

This paper is structured as follows. In Sect. 2, we describe the mathematical formalism for the interacting two-particle quantum walks and the evaluation of entanglement between the two particles. In Sect. 3, we perform spectral analysis on the time-evolution of probability distribution and entanglement time series of two-particle discrete-time quantum walks for different values of interaction strength. Also presented are the corresponding Feigenbaum diagrams of the frequencies in the time series. Finally, Sect. 4 contains our conclusions.

2 Interacting two-particle quantum walks and entanglement

As described in [19,24], a two-particle quantum walk takes place in the Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$, where $\mathcal{H}_i = (\mathcal{H}_v \otimes \mathcal{H}_c)_i$ for particle i, with v and c representing the position and coin space, respectively. Denoting the two-particle basis states as

$$\left|v_i, v_k; c_j, c_l\right\rangle = \left|v_i, c_j\right\rangle_1 \otimes |v_k, c_l\rangle_2, \tag{1}$$

we can write the wavefunction describing the two-particle system as a linear combination of basis states

$$|\psi\rangle = \sum_{ik}\sum_{jl} a_{v_i,v_k;c_j,c_l}\left|v_i, v_k; c_j, c_l\right\rangle. \tag{2}$$

The normalization condition is expressible as:

$$\sum_{ik}\sum_{jl} |a_{v_i,v_k;c_j,c_l}|^2 = 1. \tag{3}$$

The time-evolution operator for the discrete-time two-particle quantum walk is

$$U := S \cdot (\mathbb{1} \otimes C), \tag{4}$$

where the shifting operator $S = S_1 \otimes S_2$ acts on the basis state as

$$S\left|v_i, v_k; c_j, c_l\right\rangle = \left|v_j, v_l; c_i, c_k\right\rangle, \tag{5}$$

and the coin operator C_{ik} is a $d_i d_k \times d_i d_k$ unitary matrix given by $C_{ik} = (C_i)_1 \otimes (C_k)_2$. Here d_i is the degree of vertex i, $(C_i)_1$ is the $d_i \times d_i$ coin matrix for the vertex v_i, and $(C_k)_2$ is the $d_k \times d_k$ coin matrix for the vertex v_k. The matrix C_{ik} acts on the $d_i d_k$-dimensional coin space with the basis states ordered in the following manner:

$$\{|v_i, v_k; c_1, c_1\rangle, \ldots, \left|v_i, v_k; c_1, c_{d_k}\right\rangle, |v_i, v_k; c_2, c_1\rangle, \ldots, \left|v_i, v_k; c_{d_i}, c_{d_k}\right\rangle\}.$$

In this work, several initial states are considered. One initial state of the system is chosen to be an equal superposition of position states on a graph with N vertices,

$$|\psi_0\rangle = \sum_{i,k=1}^{n} \sum_{j=1}^{d_i} \sum_{l=1}^{d_k} \frac{1}{N\sqrt{d_i d_k}} \big| v_i, v_k; c_j, c_l \big\rangle. \tag{6}$$

Other initial states studied are random. Each coefficient $a_{v_i, v_m; c_j, c_n}$ is chosen to be a randomly generated number from a uniform distribution over [0,1], and the state is then normalized using the normalization condition described in Eq. (3).

The two-particle Grover coin operator is defined as $C_{ik} = G(d_i) \otimes G(d_k)$, where the one-particle Grover coin operator is given by

$$G_{\iota,\zeta}(d) = \frac{2}{d} - \delta_{\iota,\zeta}. \tag{7}$$

where $\delta_{\iota,\zeta}$ is the Kronecker delta function. Such a two-particle coin operator is separable, and consequently the two particle walkers evolve independently. To dynamically generate entanglement between the two particles, we introduce a local interaction between the two particles, in the form of a modification of the coin operator. A local interaction scheme perturbs the coin operators only when both particles are simultaneously at the same node. We define the effective coin operator as

$$C_{ik} = \begin{cases} G(d_i) \otimes G(d_k) & : i \neq k \\ C' & : i = k \end{cases} \tag{8}$$

where C' is the perturbed coin operator. We consider the ϕ-Grover interaction scheme as introduced in [24], where $C' = e^{i\phi} G(d_i) \otimes G(d_i)$.

The joint probability of particle 1 located at vertex v_i and particle 2 located at vertex v_k simultaneously, after t steps of the quantum walk, is

$$P(i,k,t) = \sum_{jl} |\big\langle v_i, v_k; c_j, c_l \big| (U)^t |\psi_0\rangle|^2. \tag{9}$$

The marginal probabilities for particles 1 and 2, assuming the particles are distinguishable, are obtained by summing the joint probability over the position states of the other particle, i.e.

$$\begin{aligned} P_1(i,t) &= \sum_k P(i,k,t) = \sum_k \sum_{jl} |\big\langle v_i, v_k; c_j, c_l \big| (U)^t |\psi_0\rangle|^2, \\ P_2(k,t) &= \sum_i P(i,k,t) = \sum_i \sum_{jl} |\big\langle v_i, v_k; c_j, c_l \big| (U)^t |\psi_0\rangle|^2. \end{aligned} \tag{10}$$

For a system in the pure state $|\psi\rangle$ defined in Eq. (2), the reduced density matrix ρ_1 is obtained by tracing the density matrix $\rho = |\psi\rangle\langle\psi|$ over subsystem 2 (namely both position and coin states of particle 2). Using orthonormality, we obtain

$$\rho_1 = \sum_{ik} \sum_{jl} b_{v_i,v_k;c_j,c_l} \left| v_i, c_j \right\rangle \langle v_k, c_l|, \tag{11}$$

where $b_{v_i,v_k;c_j,c_l} = \sum_m \sum_n a_{v_i,v_m;c_j,c_n} a^*_{v_k,v_m;c_l,c_n}$. For such bipartite systems, entanglement can be measured by the von Neumann entropy using the reduced density matrix of either subsystem [26]

$$E(|\psi\rangle) = S(\rho_1) = -\mathrm{Tr}\left(\rho_1 \log_2(\rho_1)\right), \tag{12}$$

where $\mathbf{0}\ln(\mathbf{0}) \equiv 0$. Since the reduced density matrix ρ_1 has real non-negative eigenvalues λ_i and the trace is invariant under the similarity transformation, we can compute the entanglement as

$$E(|\psi\rangle) = -\sum_i \lambda_i \log_2(\lambda_i). \tag{13}$$

3 Entanglement time series and spectral analysis

Using the interacting coin operator defined by Eq. (8), the entanglement between the two walking particles varies in time and is also dependent on the interaction strength ϕ. The corresponding power spectra of the entanglement time series are obtained via a discrete Fourier transform of a modified time series. We denote the (normalized) modulus square of the Fourier transform as $|\mathcal{E}|^2$. The linear trend is subtracted from the time series, and then multiplied by a tapered cosine window, defined by

$$w(n) = \begin{cases} \frac{1}{2}\left(1 + \cos\left(\pi\left(\frac{2n}{\alpha(N-1)} - 1\right)\right)\right) & 0 \le n \le \frac{\alpha(N-1)}{2} \\ 1 & \frac{\alpha(N-1)}{2} \le n \le (N-1)(1-\frac{\alpha}{2}) \\ \frac{1}{2}\left(1 + \cos\left(\pi\left(\frac{2n}{\alpha(N-1)} - \frac{2}{\alpha} + 1\right)\right)\right) & (N-1)(1-\frac{\alpha}{2}) \le n \le N-1 \end{cases} \tag{14}$$

where N is the width of the time series, $n = 0, 1, \ldots, N-1$, $\alpha N/2$ is the width of the cosine lobes, and $(1-\alpha/2)N$ is the width of the rectangle window. The window function is used to eliminate the discontinuity at the endpoints of the time series, since the length of the time series is not an exact multiple of the fundamental period, if it exists. We then take the discrete Fourier transform of the modified time series. The dependence of this power series on ϕ can be visualised as a Feigenbaum bifurcation diagram. We plot the most prominent frequencies in the power series as a function of ϕ. Only frequencies with an amplitude in the top 5 % (black dots) and second 5 % (grey dots) of all amplitudes in the spectrum are shown. Frequencies with smaller amplitudes do not appear on this diagram. In this work, the parameter $\alpha = 0.4$ is used, which provides an effective reduction of noise in the frequency spectra. A different value of α leads to a slightly different Feigenbaum diagram, but in any case it does not affect the overall appearance.

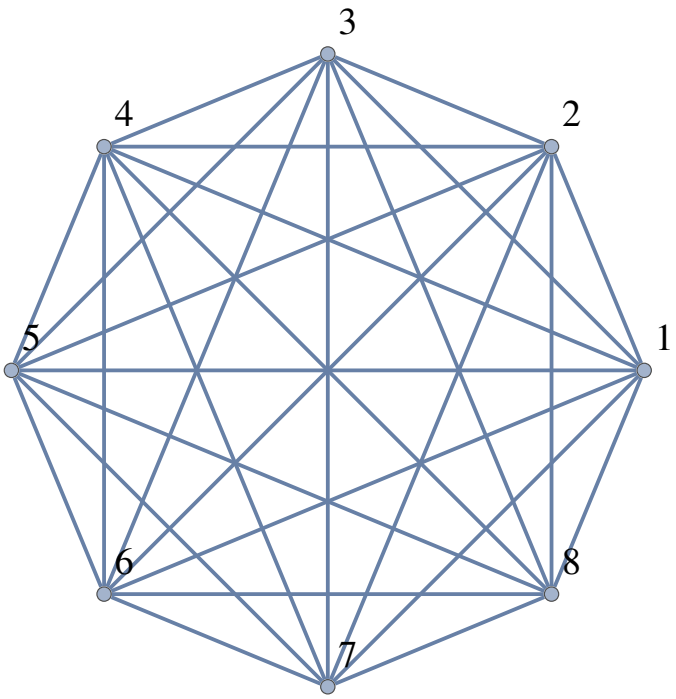


Fig. 1 K_8 graph without self-loops

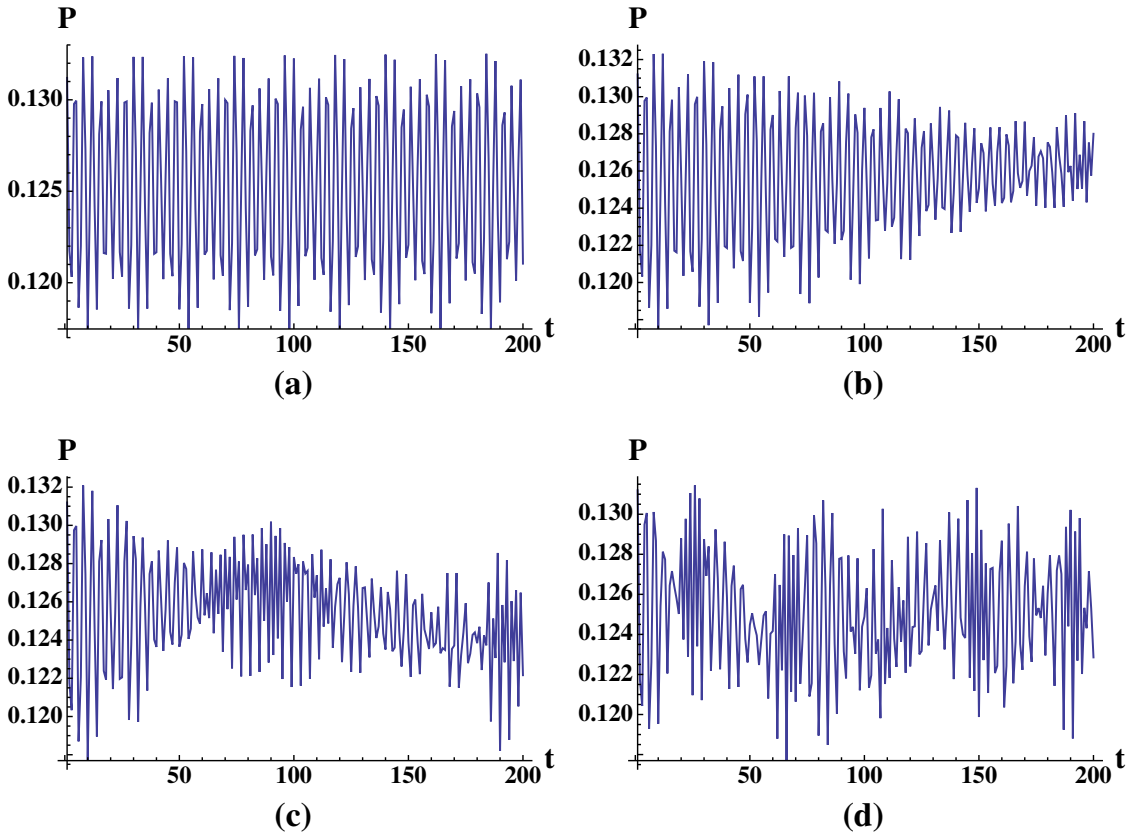


Fig. 2 Marginal probability of a particle at vertex 1 for a two-particle quantum walk on the K_8 graph without self-loops under ϕ-Grover interaction, with the initial state being random. **a** $\phi = 0.0$, **b** $\phi = 0.01\pi$, **c** $\phi = 0.03\pi$, **d** $\phi = 0.1\pi$

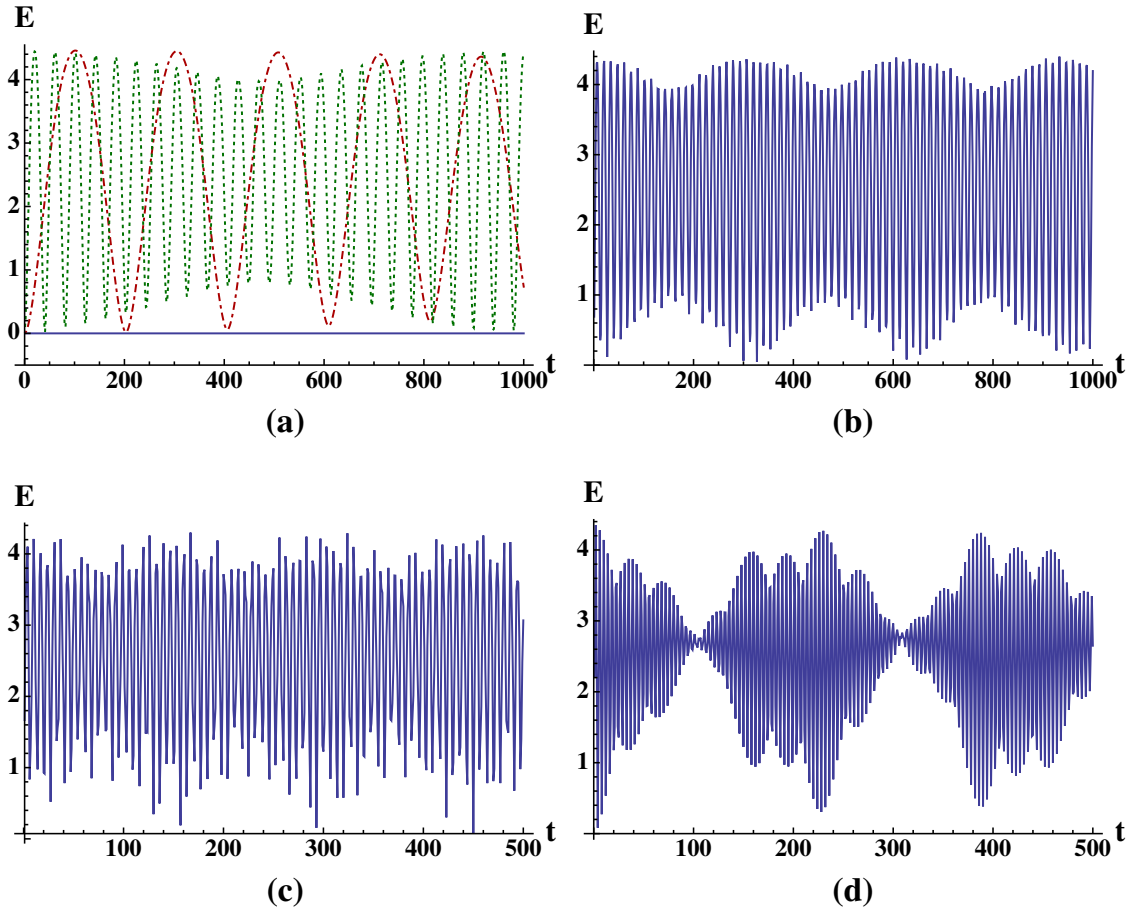


Fig. 3 Entanglement time series for a two-particle quantum walk on the K_8 graph without self-loops under ϕ-Grover interaction, with an equal superposition initial state. **a** $\phi = 0.0$ (*blue solid line*), $\phi = 0.02\pi$ (*red dot-dashed line*), $\phi = 0.1\pi$ (*green dotted line*), **b** $\phi = 0.3\pi$, **c** $\phi = 0.6\pi$, **d** $\phi = 0.99\pi$ (Color figure online)

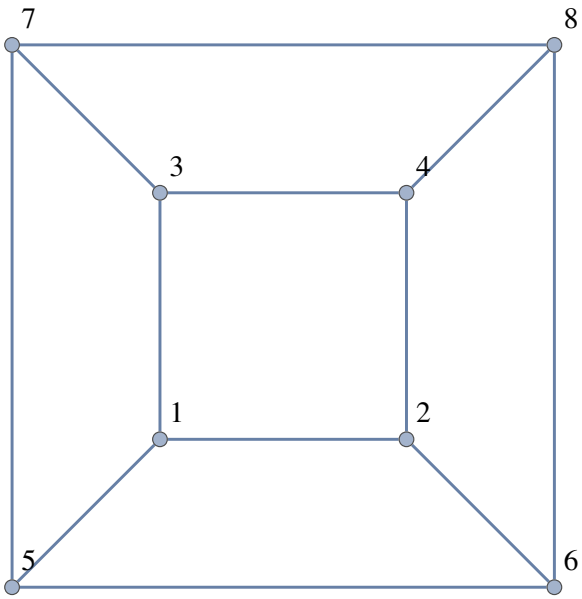


Fig. 4 3-Dimensional hypercube graph Q_3

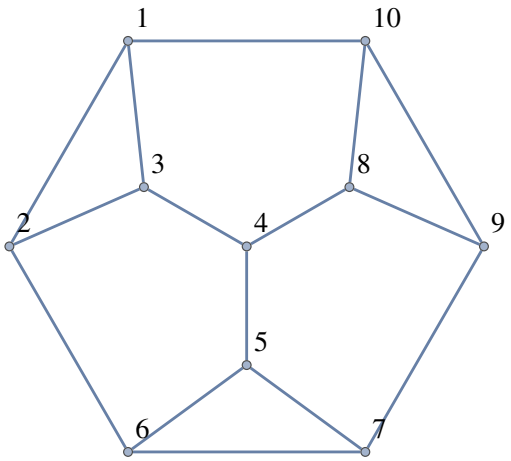


Fig. 5 Joined second-generation 3-Cayley Tree

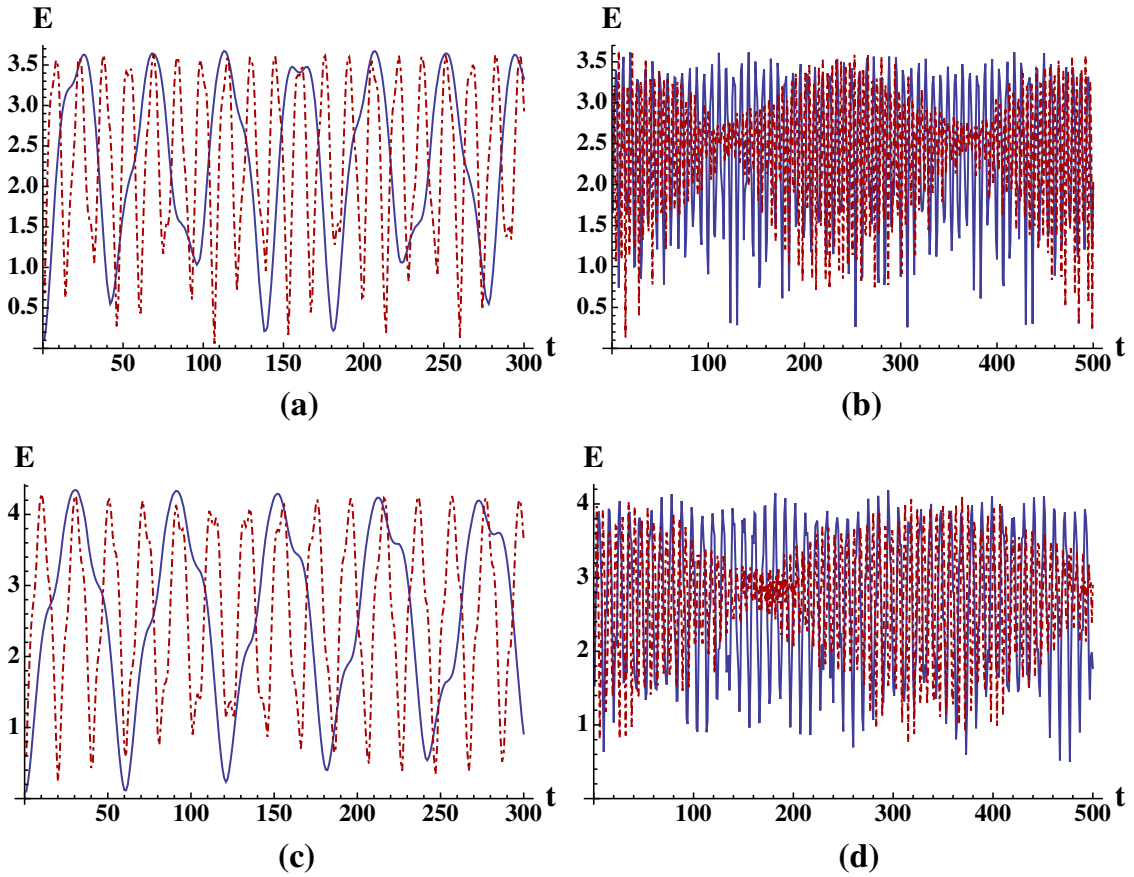


Fig. 6 Entanglement time series for a two-particle quantum walk on the Q_3 and joined 3CT2 graphs under ϕ-Grover interaction, with an equal superposition initial state. **a** Q_3 graph, $\phi = 0.1\pi$ (*blue solid line*), $\phi = 0.3\pi$ (*red dot-dashed line*), **b** Q_3 graph, $\phi = 0.6\pi$ (blue solid line), $\phi = 0.99\pi$ (*red dot-dashed line*), **c** Joined 3CT2 graph, $\phi = 0.1\pi$ (*blue solid line*), $\phi = 0.3\pi$ (*red dot-dashed line*), **d** joined 3CT2 graph, $\phi = 0.6\pi$ (*blue solid line*), $\phi = 0.99\pi$ (*red dot-dashed line*) (Color figure online)

3.1 Degree-regular graphs

Degree-regular graphs have the same number of edges joined to each vertex, and as such tend to be very symmetric. First, we consider the highly symmetric and regular complete graphs on n vertices, namely the K_n graphs, with and without self-loops.

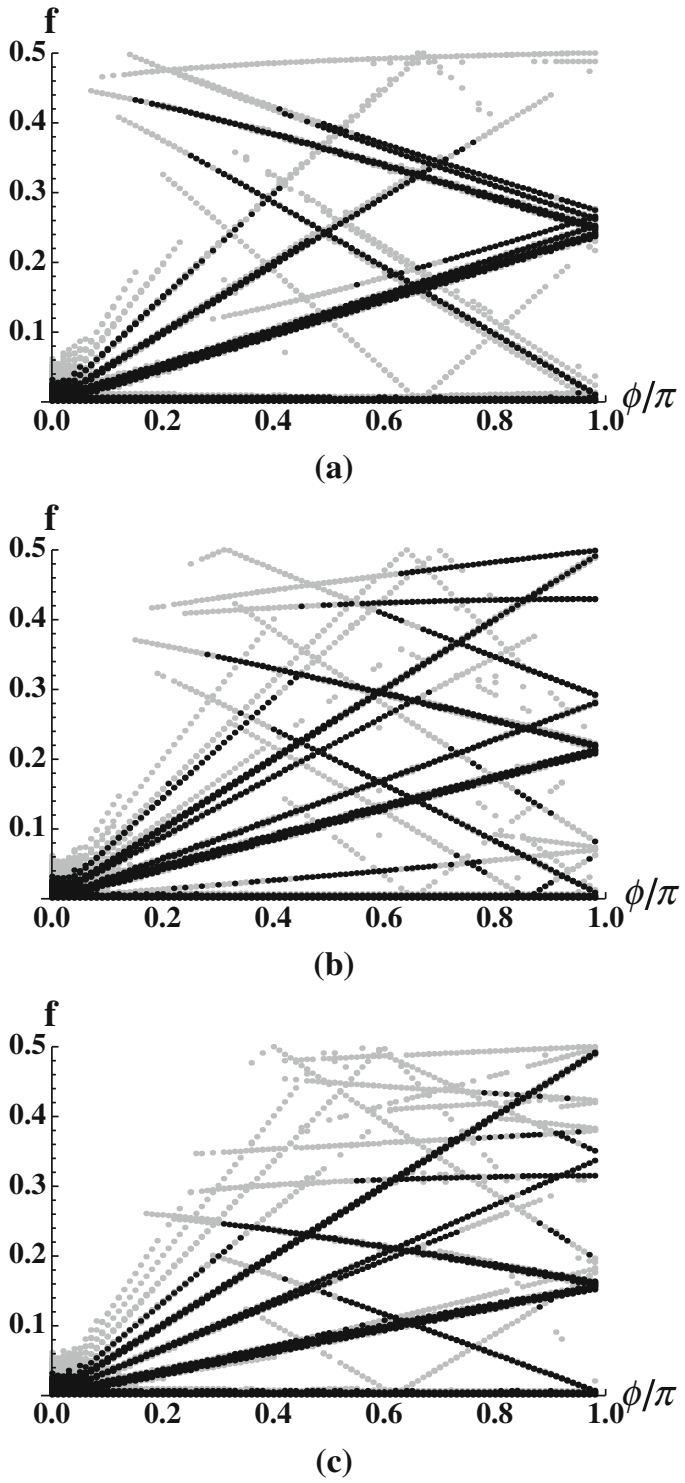


Fig. 7 Feigenbaum diagrams of frequencies present in the entanglement time series of several regular graphs under ϕ-Grover interaction, with an equal superposition initial state. **a** K_8, **b** Q_3, **c** joined 3CT2

Due to the symmetry and regularity of these graphs, when the system starts at an equal superposition of all states, the joint probability distribution defined in Eq. (9) and the marginal probability distributions defined in Eq. (10) remain uniform at all times, which is independent of the ϕ value. That is, for a K_n graph, $P(i,k,t) \equiv 1/n^2$ and $P_1(i,t) \equiv P_2(k,t) \equiv P(j,t) \equiv 1/n$ for all i, k, j and t. This is a reflection of the symmetry of the K_n graphs, which are strongly regular, as well as vertex-, edge-, and distance-transitive. However, the entanglement between the two particles as defined

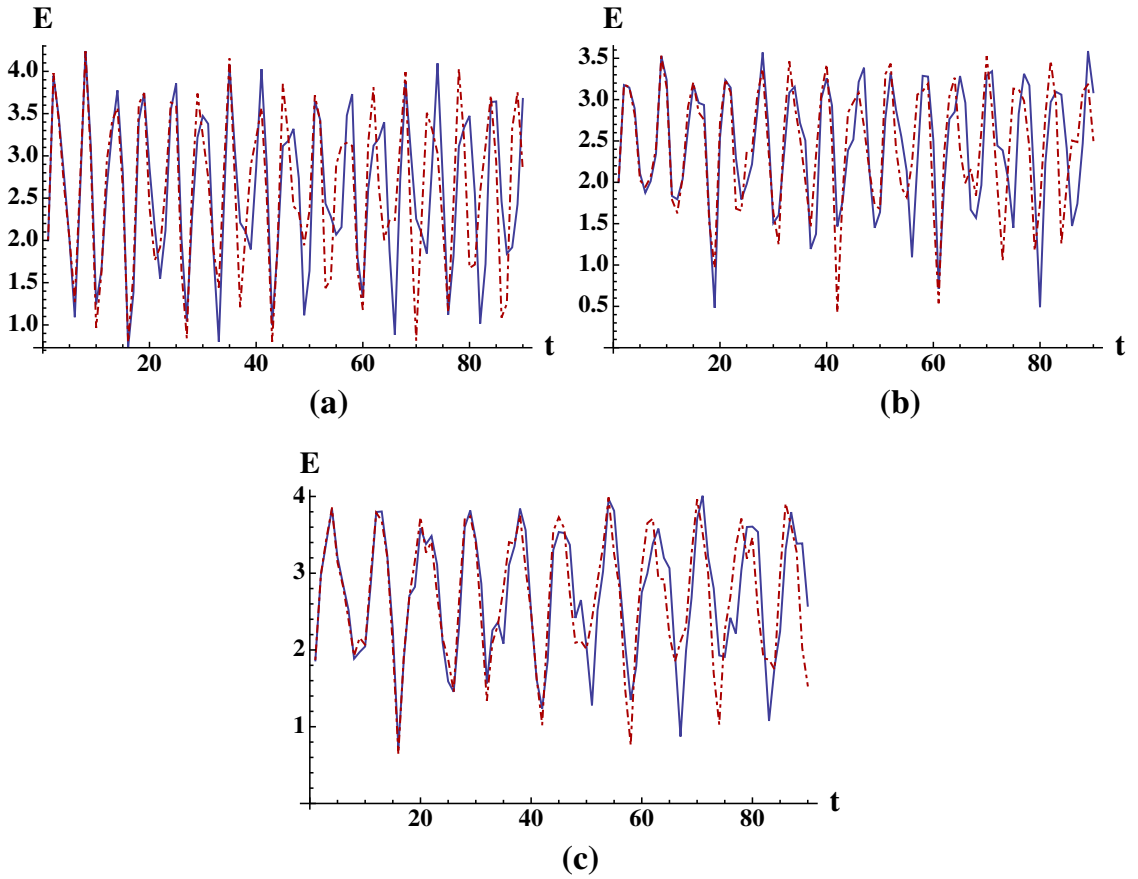


Fig. 8 Time series of entanglement for $\phi = 0.75\pi$ (*blue, solid*) and $\phi = 0.76\pi$ (*green, dotted*) for a two-particle quantum walk under ϕ-Grover interaction beginning in an equal superposition initial state. **a** K_8 graph, **b** Q_3 graph, **c** joined 3CT2 graph (Color figure online)

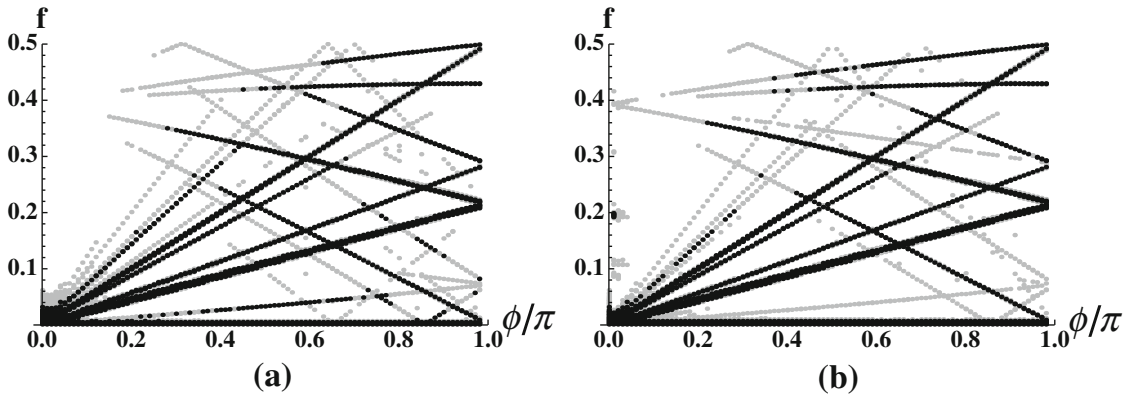


Fig. 9 Feigenbaum diagrams of frequencies present in the entanglement time series of the Q_3 graph under ϕ-Grover interaction. **a** Equal superposition initial state, **b** random initial state

in Eq. (13) changes with time even for an initially equal superposition of all states, if they interact with a nonzero ϕ value.

Since the probability and entanglement dynamics are very similar for these complete graphs, in the following we will only present the results for the K_8 graph without self-loops (Fig. 1). We first show the marginal probability for a random initial state that breaks the symmetry. In this case, the probability distributions vary in time and are

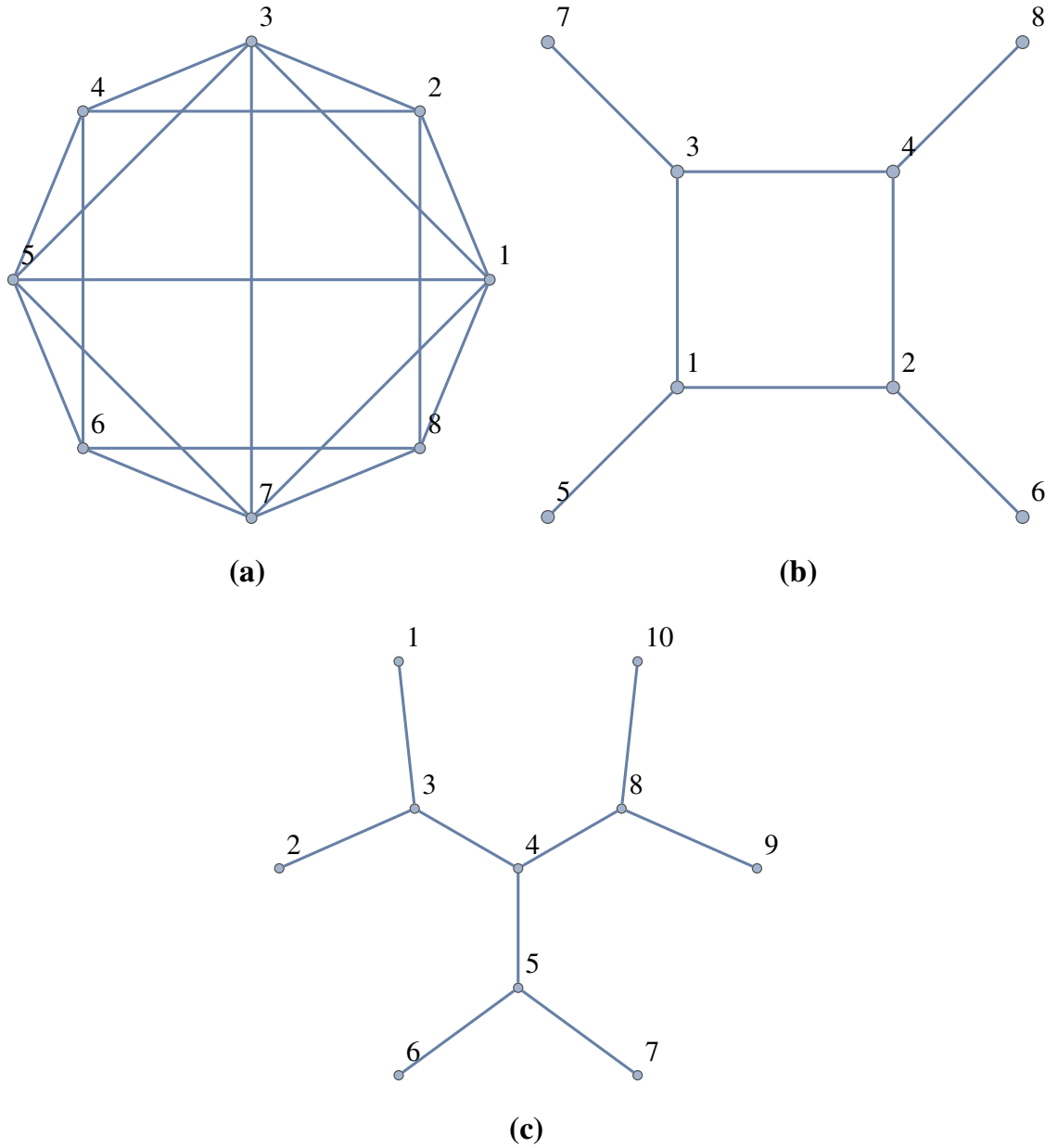


Fig. 10 Non-degree-irregular counterparts of regular graphs. **a** K_8 with 10/28 edges removed, **b** Q_3 with 4/12 edges removed, **c** 2nd generation 3-Cayley tree

different for different ϕ values, becoming more complex for larger ϕ, as shown in Fig. 2.

We observe that the entanglement dynamics for the K_8 graph under the ϕ-Grover interaction scheme are changing with time, even when the system begins at an equal superposition of all states, as shown in Fig. 3. Interesting to note is that for $\phi \to \pi$ the time series appears to form wave packets. Nonetheless, the entanglement time series remain relatively simple and are oscillatory with the overall pattern repeating as $t \to \infty$. The dynamics can be described by a number of frequencies, which increase for increasing ϕ, as best seen in the Feigenbaum diagram shown in Fig. 7a.

Another family of very symmetric regular graphs is the hypercube graphs Q_n. Two-particle quantum walks on the 3-dimensional hypercube graph (Fig. 4) were

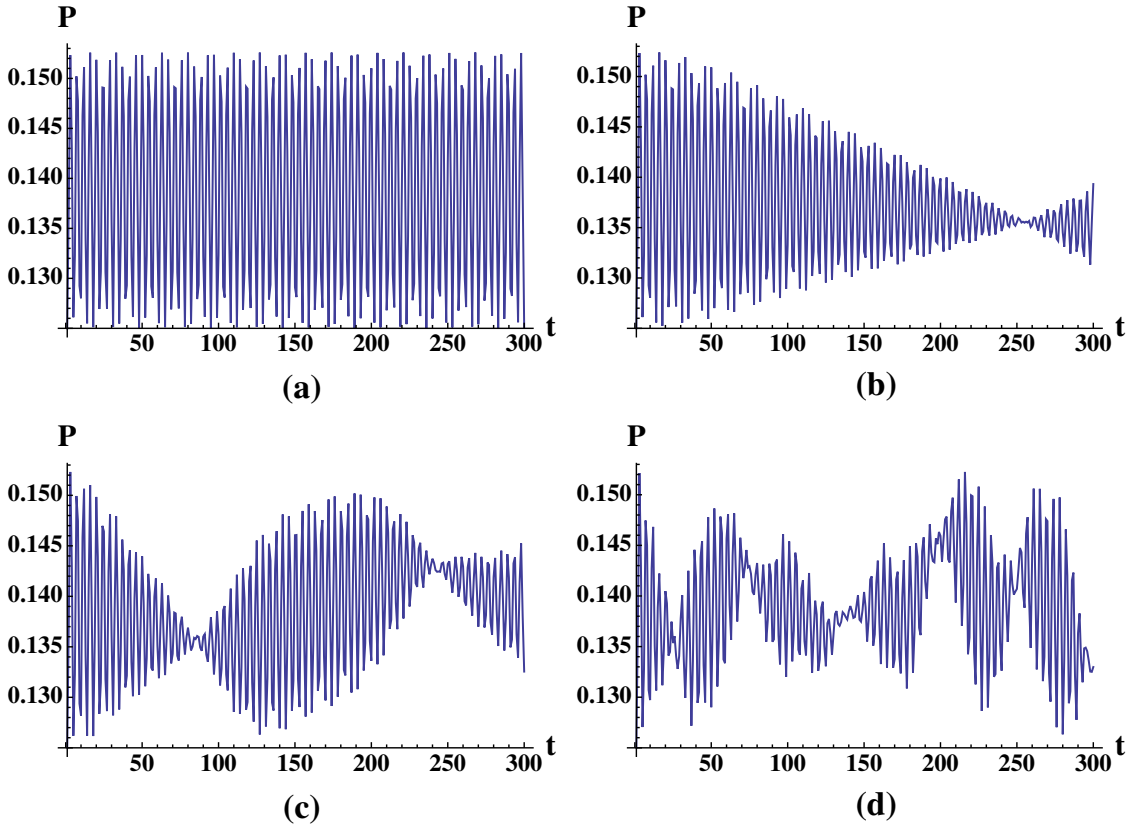


Fig. 11 Marginal probability of a particle at vertex 1 for a two-particle quantum walk on the modified K_8 graph without self-loops under ϕ-Grover interaction, with the initial state being an equal superposition of all states. **a** $\phi = 0.0$, **b** $\phi = 0.01\pi$, **c** $\phi = 0.03\pi$, **d** $\phi = 0.1\pi$

considered. The Q_3 graph has the same dimension as the K_8 graph, so the two are comparable. As can be seen in Fig. 6, the entanglement dynamics of the Q_3 graph differs from that of the K_8 graph, but follow the same trend of increasing complexity as ϕ increases. The entanglement time series appears less regular for this graph than for the K_8 graph, reflecting a reduced degree of symmetry.

Walks on the joined second-generation 3-Cayley tree (3CT2) shown in Fig. 5 were also studied. Again, the complexity of the time series increases with increasing ϕ, and these time series again appear more complex than those for the K_8 graph (Fig. 6). Comparing the Feigenbaum diagrams for the entanglement time series of the K_8, Q_3 and joined 3CT2 graphs (Fig. 7), it can be seen that for all of these graphs the frequency spectra behave similarly as ϕ changes. All have a few prominent frequencies for lower values of ϕ, and these frequencies change on the whole linearly with ϕ. The dynamics become much more complex with increasing ϕ, with more frequencies emerging. The Q_3 and joined 3CT2 graphs have many more frequencies than the K_8 graph, especially for the higher values of ϕ.

As shown in Fig. 8, the entanglement is sensitive to small perturbations in the ϕ parameter for every regular graph studied. This is true for every initial state considered. However, the marginal single-particle probability time series are not so sensitive to small changes in ϕ. For the equal superposition initial state, the probability time series remain uniform, independent of ϕ. For the random initial state, probability time series

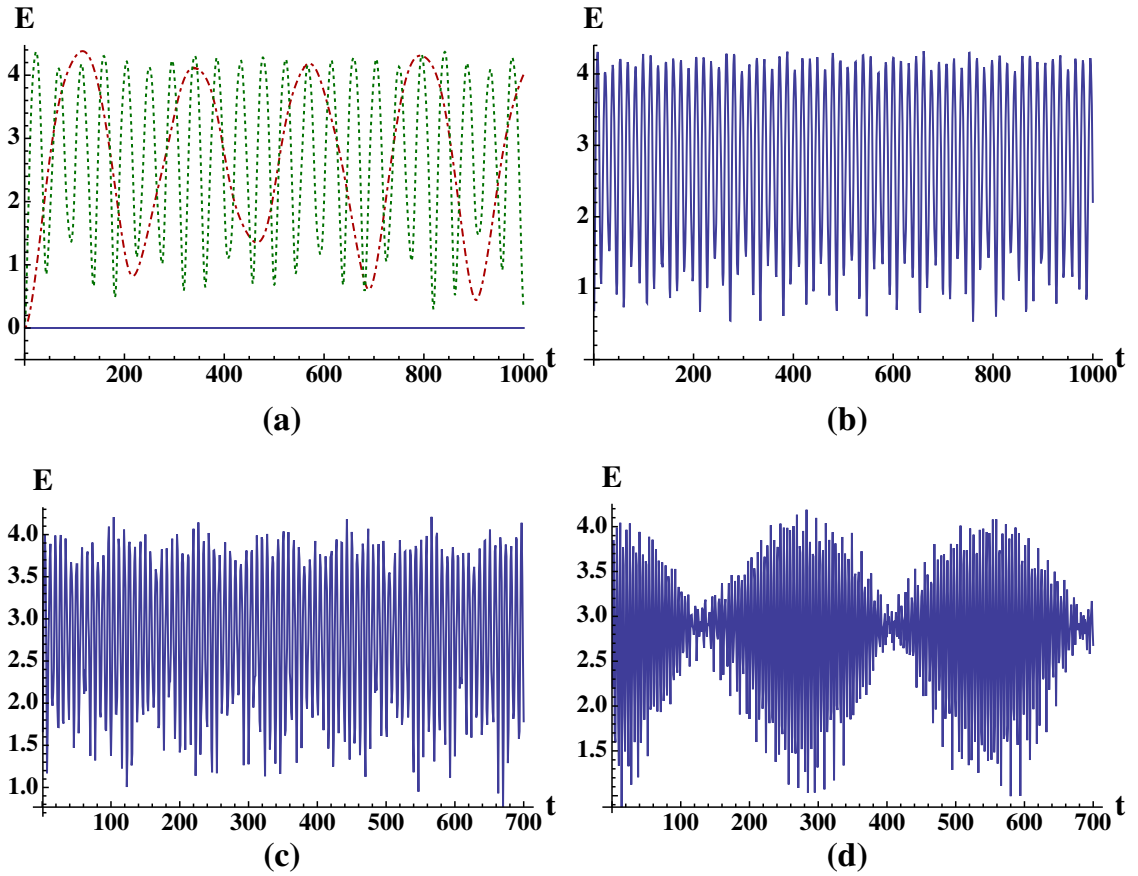


Fig. 12 Entanglement time series for a two-particle quantum walk on the modified K_8 graph without self-loops under ϕ-Grover interaction, with an equal superposition initial state. **a** $\phi = 0.0$ (*blue solid line*), $\phi = 0.02\pi$ (*red dot-dashed line*), $\phi = 0.1\pi$ (*green dotted line*), **b** $\phi = 0.3\pi$, **c** $\phi = 0.6\pi$, **d** $\phi = 0.99\pi$ (Color figure online)

are complex but essentially stable. On the other hand, the entanglement time series are not sensitive to the initial state of the system. The various initial states give similar entanglement time series, appearing to have the same frequencies and general shape. The Feigenbaum diagrams for each of the regular graphs discussed above are almost identical for both the equal superposition and a random initial state, as shown in Fig. 9.

3.2 Non-degree-irregular graphs

We now consider degree-irregular graphs, and for comparison purposes these are all similar to the degree-regular graphs considered above, with several edges removed so that not all vertices are joined to the same number of edges. The irregularity of these graphs introduces more complex dynamics into the system, making them good candidates to study the complexity in the time-evolution of probability distributions and quantum correlations within the system with varying interaction strength ϕ. The degree-irregular graphs studied in this work are the unjoined second-generation 3-Cayley tree, a modified K_8 graph and a modified Q_3 graph, all with between 30 and 40 % of their edges removed (Fig. 10). These graphs still have a high degree of symmetry, as their regular counterparts do.

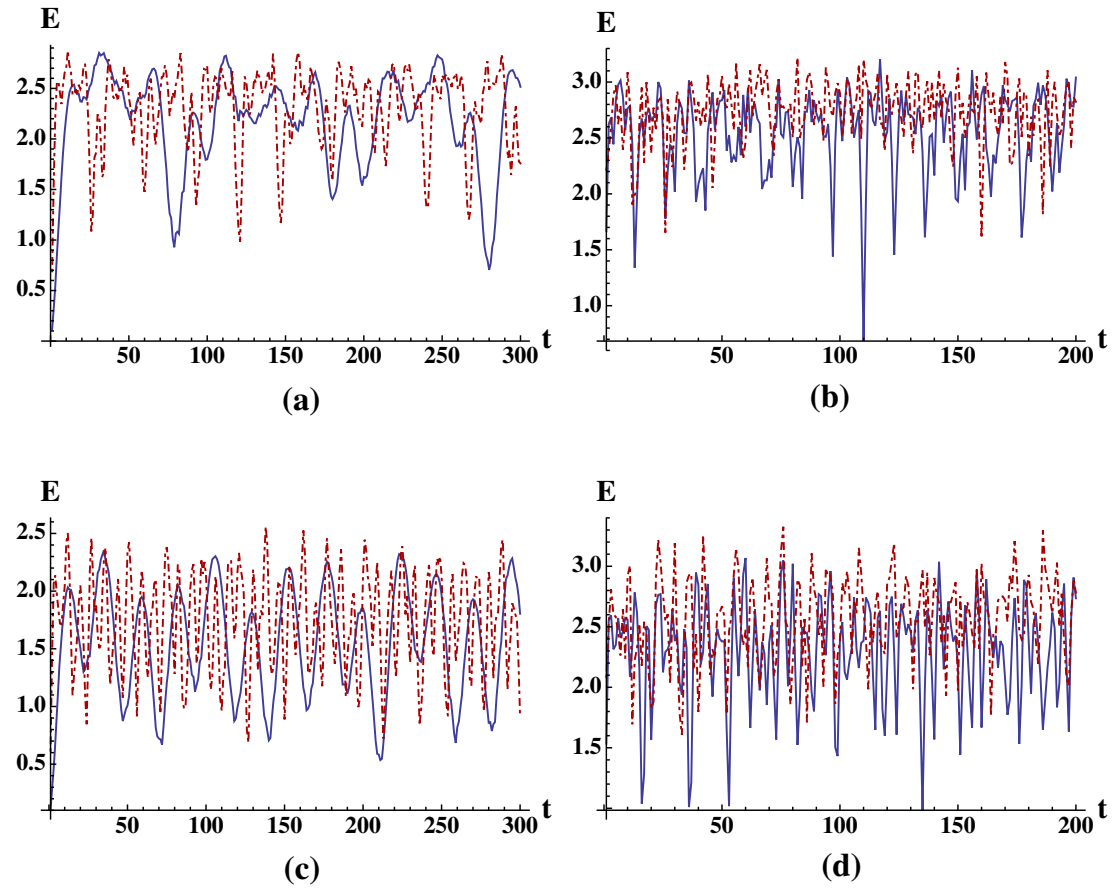


Fig. 13 Entanglement time series for a two-particle quantum walk on the modified Q_3 and unjoined 3CT2 graphs under ϕ-Grover interaction, with an equal superposition initial state. **a** Modified Q_3 graph, $\phi = 0.1\pi$ (*blue solid line*), $\phi = 0.3\pi$ (*red dot-dashed line*), **b** Modified Q_3 graph, $\phi = 0.6\pi$ (*blue solid line*), $\phi = 0.99\pi$ (*red dot-dashed line*), **c** Unjoined 3CT2 graph, $\phi = 0.1\pi$ (*blue solid line*), $\phi = 0.3\pi$ (*red dot-dashed line*), **d** Unjoined 3CT2 graph, $\phi = 0.6\pi$ (*blue solid line*), $\phi = 0.99\pi$ (*red dot-dashed line*) (Color figure online)

These graphs have different probability time series to their regular analogues. The single-particle probability is no longer uniform for the equal superposition equal state, but is periodic for very low values of ϕ, and increasingly complex for larger ϕ. Figure 11 shows these time series for the modified K_8 graph. When the system starts in a random initial state, the time series are no longer very regular for lower values of ϕ.

We now consider the entanglement dynamics of two-particle quantum walks along these irregular graphs. There is no entanglement between the two particles for $\phi = 0$, which corresponds to having no interaction between the two particles. With weak interaction at small values of ϕ, the entanglement time series is periodic with a fixed number of frequencies. As we increase ϕ, we observe a transition to irregular/quasi-periodic dynamics as more frequencies are introduced further into the system. Figure 12 illustrates this for the modified K_8 graph, while some entanglement time series for the other two irregular graphs can be seen in Fig. 13.

When we compare the time series for the irregular graphs with those for the regular graphs (Fig. 14), we see that the modified K_8 graph has similar time series to the original K_8 graph. The time series appear to be similar in shape, with slightly different frequencies. On the other hand, the entanglement time series for a two-particle quantum walk on the modified Q_3 graph differs much more from its regular counterpart, with

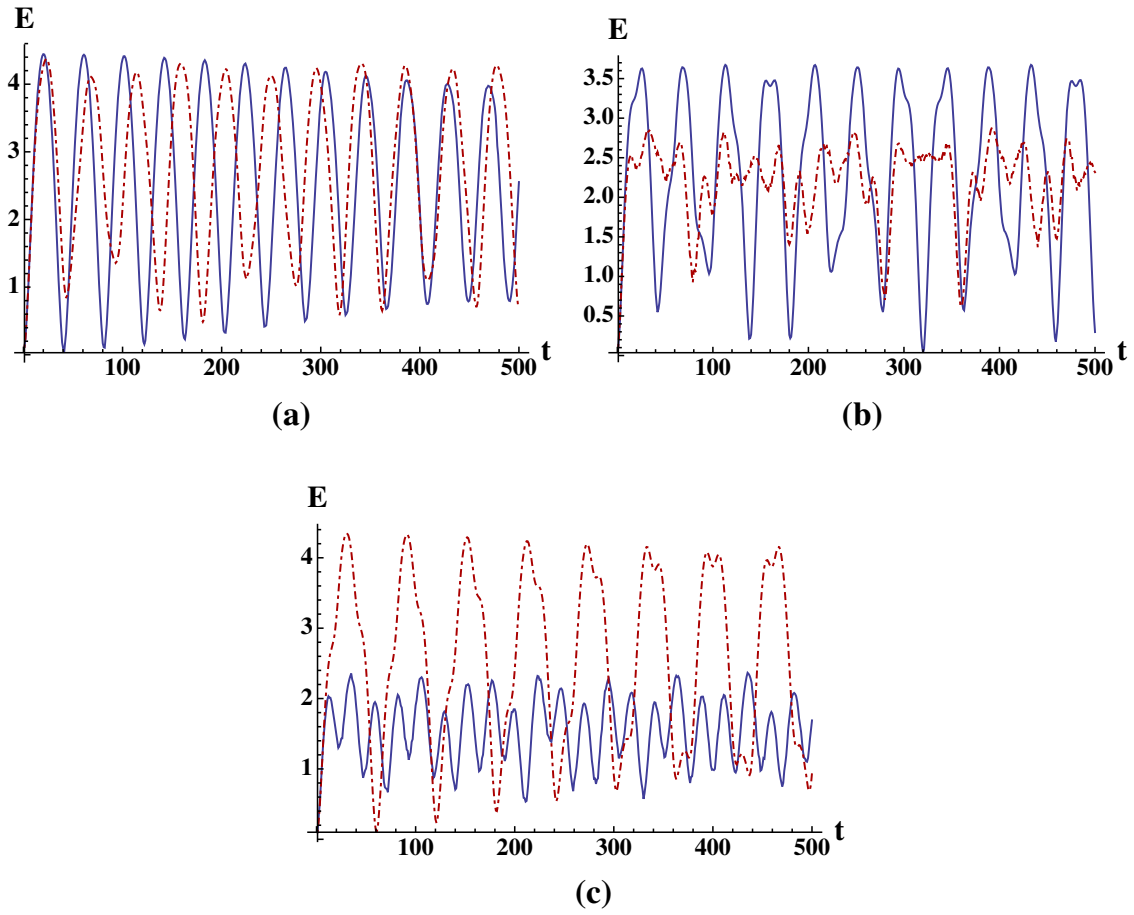


Fig. 14 Comparison of entanglement time series for regular graphs (*blue solid lines*) and their irregular counterparts (*red dot-dashed lines*), with an equal superposition initial state. **a** K_8, **b** Q_3, **c** Cayley tree (Color figure online)

more complex behaviour occurring. This is also true for the unjoined Cayley tree, which has a more complex entanglement time series than the joined Cayley tree.

This is more evident in the Feigenbaum diagrams for these time series, shown in Fig. 15, which depicts the frequencies present in the entanglement time series. As expected, a reduction in complexity is also seen as $\phi \to \pi$. The Feigenbaum diagrams for these graphs show that the entanglement time series for the irregular graphs are more complex than for the regular graphs, with many more frequencies being present, and nonlinear relationships between frequency and ϕ start to emerge (Fig. 15c).

Like the regular graphs studied, the system also exhibits sensitivity to small perturbations in the ϕ parameter for the irregular graphs, as shown in Fig. 16. It can be seen that the single-particle probabilities are less sensitive to these perturbations. Again like the regular graphs studied, the initial state does not make a considerable difference in the frequency spectrum of the entanglement time series, although the shape of the time series does vary slightly, more so for these graphs than for the regular graphs.

4 Conclusions

In conclusion, we have demonstrated that while the probability time series for simple degree-regular and degree-irregular graphs such as the K_8 graph and the second-

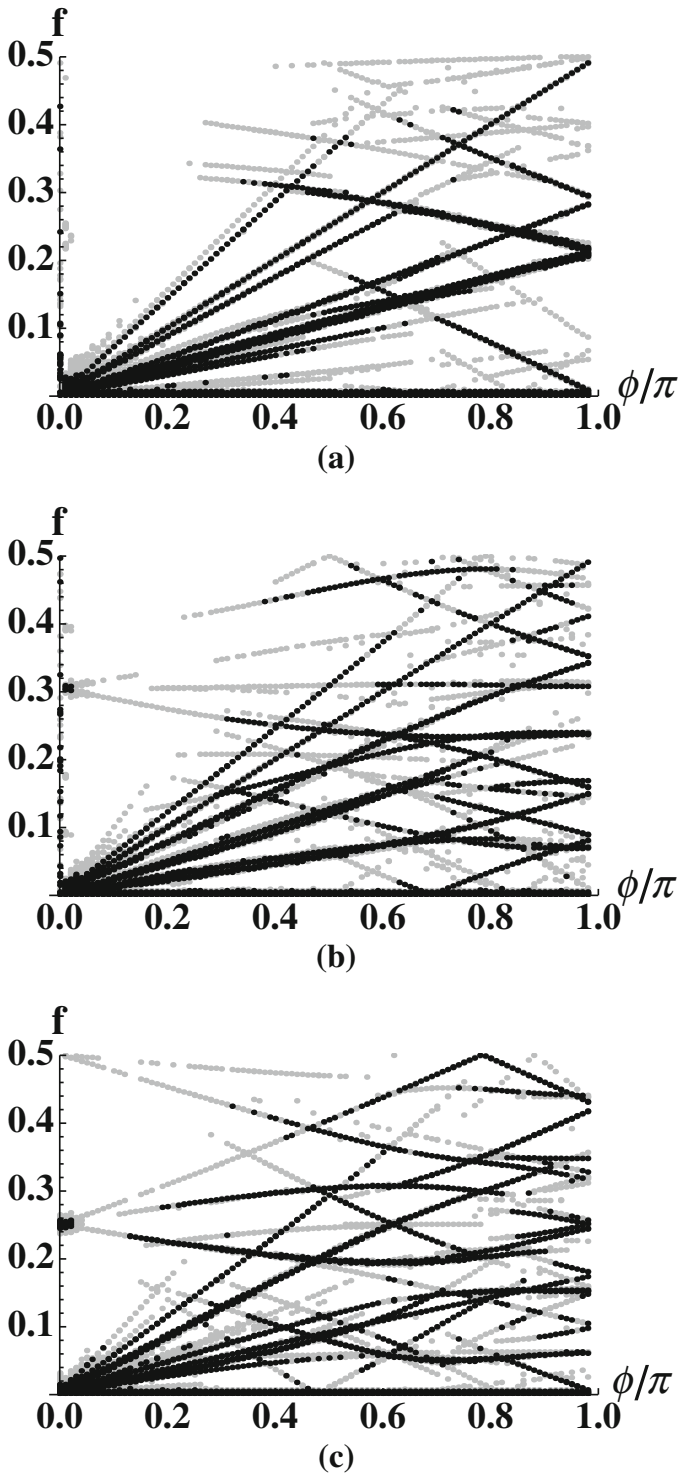
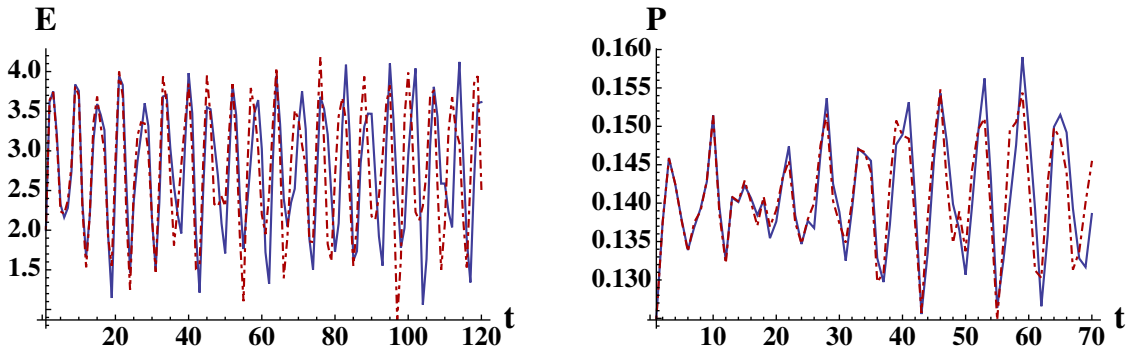
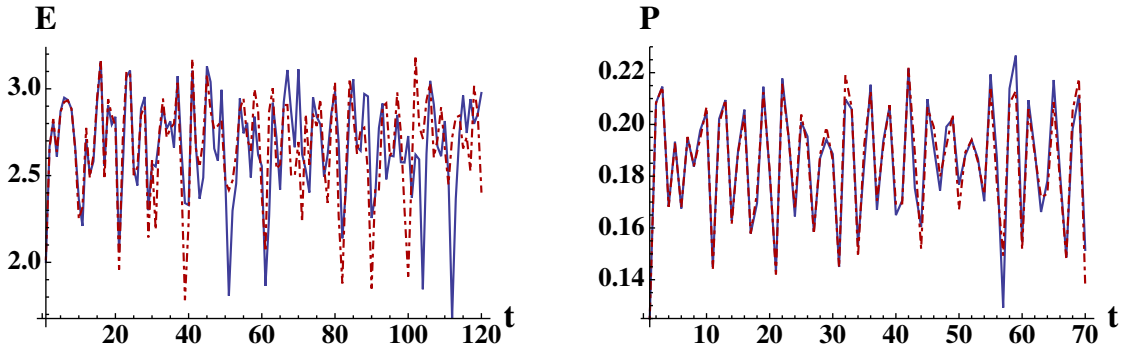


Fig. 15 Feigenbaum diagrams of frequencies present in the entanglement time series of several irregular graphs under ϕ-Grover interaction, with an equal superposition initial state. **a** Modified K_8 graph, **b** modified Q_3 graph, **c** unjoined 3CT2 graph

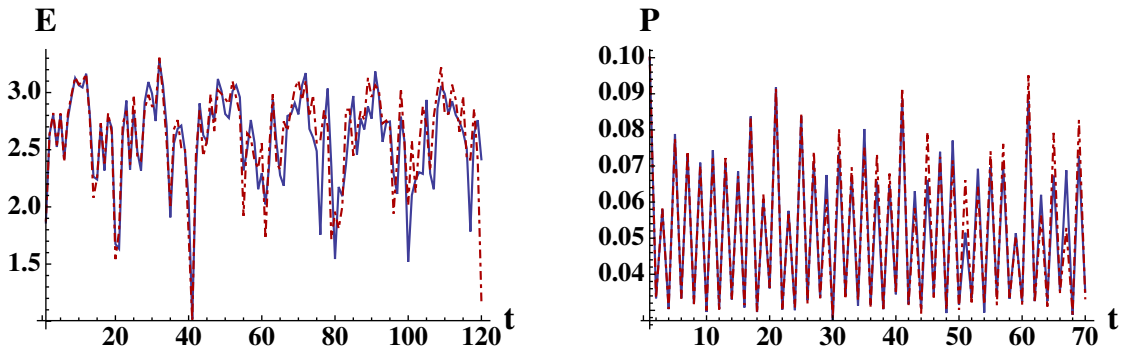
generation 3-Cayley tree are generally stable regarding small changes to interaction strength, complex entanglement dynamics can arise from a locally interacting two-particle quantum walk, and these dynamics are sensitive to small perturbations to interaction strength. Spectral analysis shows that the number of frequencies (and hence the complexity) in the entanglement time series varies as a function of the interaction strength ϕ. The complexity in the corresponding Feigenbaum diagram seems to reflect the degree of symmetry in the underlying graphs, with the degree-irregular graphs



(a) Modified K_8 graph



(b) Modified Q_3 graph



(c) Unjoined 3CT2 graph

Fig. 16 Time series of entanglement and single-particle probability, for $\phi = 0.75\pi$ (*blue, solid*) and $\phi = 0.76\pi$ (*red, dashed*) for a two-particle quantum walk under ϕ-Grover interaction beginning in an equal superposition initial state. **a** Modified K_8 graph, **b** modified Q_3 graph, **c** unjoined 3CT2 graph (Color figure online)

showing much more complex Feigenbaum diagrams than the degree-regular graphs. The sensitivity of the entanglement dynamics to small changes in the two-particle interaction strength could be a useful tool in probing small variations in the system parameters.

Acknowledgments The authors would like to thank Lock Yue Chew and Michael Small for several valuable discussions on the characterisation of nonlinear dynamics.

References

1. Furuya, K., Nemes, M.C., Pellegrino, G.Q.: Quantum dynamical manifestation of chaotic behavior in the process of entanglement. Phys. Rev. Lett. **80**, 5524–5527 (1999)
2. Angelo, R.M., Furuya, K., Nemes, M.C., Pellegrino, G.Q.: Rapid decoherence in integrable systems: a border effect. Phys. Rev. A **60**, 5407–5411 (1999)
3. Angelo, R.M., Furuya, K., Nemes, M.C., Pellegrino, G.Q.: Recoherence in the entanglement dynamics and classical orbits in the N-atom Jaynes–Cummings model. Phys. Rev. A **64**, 043801 (2001)
4. Miller, P.A., Sarkar, S.: Signatures of chaos in the entanglement of two coupled quantum kicked tops. Phys. Rev. E **60**, 1542–1550 (1999)
5. Fujisaki, H., Miyadera, T., Tanaka, A.: Dynamical aspects of quantum entanglement for weakly coupled kicked tops. Phys. Rev. E **67**, 066201 (2003)
6. Bandyopadhyay, J.N., Lakshminarayan, A.: Entanglement production in coupled chaotic systems: case of the kicked tops. Phys. Rev. E **69**, 016201 (2004)
7. Kubotani, H., Adachi, S., Toda, M.: Exact formula of the distribution of Schmidt eigenvalues for dynamical formation of entanglement in quantum chaos. Phys. Rev. Lett. **100**, 240501 (2008)
8. Hou, X.W., Hu, B.: Decoherence, entanglement, and chaos in the Dicke model. Phys. Rev. A **69**, 042110 (2004)
9. Lombardi, M., Matzkin, A.: Dynamical entanglement and chaos: the case of Rydberg molecules. Phys. Rev. A **73**, 062335 (2006)
10. Lombardi, M., Matzkin, A.: Scattering-induced dynamical entanglement and the quantum-classical correspondence. Europhys. Lett. **74**, 771–777 (2006)
11. Kempe, J.: Quantum random walks: an introductory overview. Contemp. Phys. **44**(4), 307–327 (2003)
12. Shenvi, N., Kempe, J., Whaley, K.B.: Quantum random-walk search algorithm. Phys. Rev. A **67**, 052307 (2003)
13. Reitzner, D., Hillery, M., Feldman, E., Buzek, V.: Quantum searches on highly symmetric graphs. Phys. Rev. A **79**, 012323 (2009)
14. Douglas, B.L., Wang, J.B.: A classical approach to the graph isomorphism problem using quantum walks. J. Phys. A **41**, 075303 (2008)
15. Berry, S.D., Wang, J.B.: Quantum-walk-based search and centrality. Phys. Rev. A **82**, 042333 (2010)
16. Smith, J., Mosca, M.: Handbook of natural computing. In: Rozenberg, G., Bck, T., Kok, J.N. (eds.) Algorithms for Quantum Computers. Springer, Berlin (2012)
17. Mahasinghe, A., Wang, J.B., Wijerathna, J.K.: Quantum walk-based search and symmetries in graphs. J. Phys. A **47**, 505301 (2014)
18. Manouchehri, K., Wang, J.B.: Physical Implementation of Quantum Walks. Springer, Berlin (2014)
19. Omar, Y., Paunković, N., Sheridan, L., Bose, S.: Quantum walk on a line with two entangled particles. Phys. Rev. A **74**, 042304 (2006)
20. Štefaňnák, M., Kiss, T., Jex, I., Mohring, B.: The meeting problem in the quantum walk. J. Phys. A **39**(48), 14965–14983 (2006)
21. Pathak, P.K., Agarwal, G.S.: Quantum random walk of two photons in separable and entangled states. Phys. Rev. A **75**, 032351 (2007)
22. Rohde, P.P., Fedrizzi, A., Ralph, T.C.: Entanglement dynamics and quasi-periodicity in discrete random walks. J. Mod. Opt. **59**, 710–720 (2012)
23. Venegas-Andraca, S. E., Bose, S.: Quantum-walk-based generation of entanglement between two walkers. arXiv:0901.3946v1 [quant-ph]
24. Berry, S.D., Wang, J.B.: Two-particle quantum walks: entanglement and graph isomorphism testing. Phys. Rev. A **83**, 042317 (2011)
25. Rohde, P.P., Fedrizzi, A., Ralph, T.C.: Entanglement dynamics and quasi-periodicity in discrete quantum walks. J. Mod. Opt. **59**, 710–720 (2012)
26. Mintert, F., Carvalho, A.R.R., Kus, M., Buchleitner, A.: Measures and dynamics of entangled states. Phys. Rep. **415**, 207–259 (2005)

PHYSICAL REVIEW A **86**, 042338 (2012)

Efficient circuit implementation of quantum walks on non-degree-regular graphs

T. Loke and J. B. Wang
School of Physics, The University of Western Australia, WA 6009, Australia
(Received 30 May 2012; published 31 October 2012)

This paper presents a set of highly efficient quantum circuits for discrete-time quantum walks on non-degree-regular graphs. In particular, we describe a general procedure for constructing highly efficient quantum circuits for quantum walks on star graphs of any degree and Cayley trees with an arbitrary number of layers, which are nonsparse in general. We also show how to modify these circuits to implement a full quantum-walk search algorithm on these graphs, without reference to a "black-box" oracle. This provides a practically implementable method to explore quantum-walk-based algorithms with the aim of eventual real-world applications.

DOI: 10.1103/PhysRevA.86.042338 PACS number(s): 03.67.Ac, 03.65.Ud, 03.67.Bg, 02.10.Ox

I. INTRODUCTION

Quantum walks have shown much potential as a general framework for developing a new generation of quantum algorithms. These algorithms depend on interference between the multiple paths that are simultaneously traversed by the quantum walker, as well as complex interactions and entanglement if more than one quantum walker is involved. Both discrete-time [1,2] and continuous-time [3,4] quantum walks can be considered, both of which have yielded intriguing quantum algorithms [5–12]. An increasingly pressing challenge is to build a physical apparatus that can efficiently implement a prescribed quantum walk, that is, to perform each step of the quantum walk on a specified graph.

Quantum walks can be thought of as the quantum analogues of classical random walks. They are a unitary process and can therefore be naturally implemented in a quantum system by manipulating its internal states according to a specified underlying graph structure (see, for example, references in [13,14]). However, such quantum-state-based implementation of quantum walks cannot achieve exponential efficiency in general. Exponentially efficient quantum circuit implementation of quantum walks has proven to be amenable only for graphs with a high degree of symmetry [15,16] or a very sparse structure [17,18]. All of these graphs can be characterized by a small number of parameters, increasing at most logarithmically with the number of vertices. To date, all symmetric graphs leading to an efficient quantum circuit implementation are degree-regular and also vertex-transitive. To the best of our knowledge, efficient quantum circuit construction of whole families of non-degree-regular graphs has not been considered before.

In this paper, we first describe the construction of efficient circuits for implementing quantum walks on star graphs of arbitrary size n. The construction is efficient in that only $O(\log(n))$ elementary two-qubit gates are required per quantum-walk step. We then utilize this construction to find a circuit implementation for quantum walks on Cayley trees of any dimension. We also modify these circuits to implement the full quantum-walk search algorithm on these graphs.

Here we consider discrete-time quantum walks, which take place in a Hilbert space $\mathcal{H} = \mathcal{H}_P \otimes \mathcal{H}_C$. A single step of the walk is described by the unitary evolution operator $U = SC$, where S and C are the shifting and coin operators, respectively. Consider a general undirected graph $G(V,E)$, with vertex set $V = \{v_1, v_2, v_3, \ldots\}$ and edge set $E = \{(v_i, v_j), (v_k, v_l), \ldots\}$ being unordered pairs connecting the vertices. Each node v_i has d_i edges, corresponding to d_i states, alternatively termed subnode or coin states. The quantum walk acts on an extended node-subnode space, consisting of states $|v_i, a_i\rangle$, where $1 \leqslant a_i \leqslant d_i$. The shifting operator S swaps the subnode states connected by an edge, with its action defined by $S|v_i, a_i\rangle = |v_j, a_j\rangle$, and $S^2 = \mathbf{I}$.

The coin operator comprises a group of unitary operators, or a set of coins, which independently mix the probability amplitudes associated with the group of subnodes of a given node. For the vertex v_i, this corresponds to a unitary $d_i \times d_i$ matrix acting on the coin space of v_i. Berry *et al.* [19] have recently developed a software package called QWVIZ which provides an interactive visualization of the time evolution of quantum walks on arbitrarily complex graphs.

In the following, we present explicit circuit implementations for quantum walks on families of non-degree-regular graphs, which by definition are also non-vertex-transitive. Constructing quantum circuits for non-degree-regular graphs is expected to be more difficult than for degree-regular and vertex-transitive graphs since the coin space of each vertex is no longer of equal size and different vertices have a different neighboring structure. Consequently, there will be redundant states in a circuit implementation of such a graph. Also, since the graph is not vertex-transitive, how we label the node and subnode states plays an important role in achieving the most efficient quantum circuit implementation. Moreover, we note

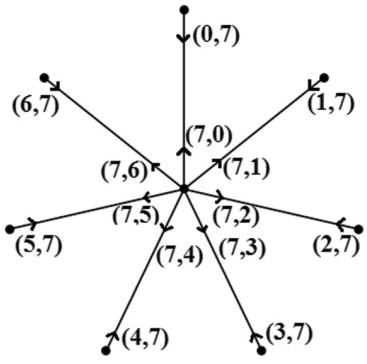


FIG. 1. The S_7 graph with node-subnode states labeled as (v_i, a_i).

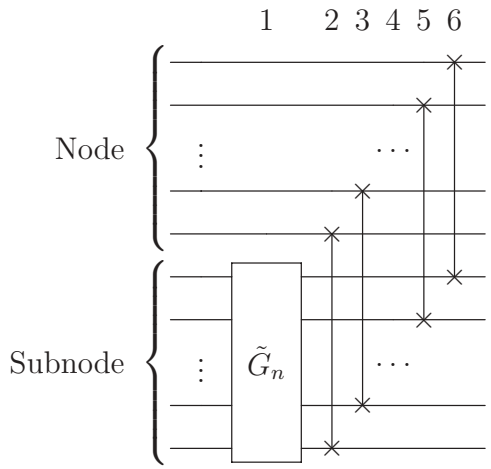


FIG. 2. Efficient circuit implementation of quantum walks on an S_n graph.

that the graphs considered here display centrality, a property analyzed in detail by Berry and Wang [10].

II. STAR GRAPHS

Define a star graph S_n of order n as having 1 vertex of degree n (the central vertex) connected to n vertices of degree 1 (the leaf vertices). An S_n graph has $n+1$ vertices. Figure 1 shows the S_7 graph with the node-subnode states labeled as (v_i,a_i). We take the identity matrix as the coin operator for the leaf vertices. For the central vertex, we use a modified Grover matrix, defined as

$$(\tilde{G}_n)_{i,j} = \begin{cases} \frac{2}{n} - \delta_{i,j}, & i \leqslant n, j \leqslant n, \\ \delta_{i,j}, & \text{otherwise}, \end{cases} \tag{1}$$

which has an extended dimension, rounding up n to the next-highest power of 2. In other words, the $\tilde{G}_n$ coin operator is defined as the Grover operator for the first $n \times n$ elements and the identity operator otherwise.

We parametrize the graph using node and subnode states. Here a node state of 0 to $n-1$ labels the leaf vertices, and the node state n labels the central vertex. Without undue loss of generality, we label the subnode states such that the shifting operator acts as $S|v,s\rangle = |s,v\rangle$. In a circuit implementation, this corresponds to simply swapping the probability amplitudes between the node and subnode states after each quantum-walk step. In total, there are $2n$ distinct states:

$$|v,s\rangle \in \{|0,n\rangle, \ldots, |n-1,n\rangle, \quad |n,n-1\rangle, \ldots, |n,0\rangle\}. \tag{2}$$

For the central node, $v \equiv n$, and for the leaf nodes, $s \equiv n$. Using the S_7 star graph as an example, we have

$$|v,s\rangle \in \{|0,7\rangle, |1,7\rangle, \ldots, |6,7\rangle, |7,6\rangle, \ldots, |7,0\rangle\}, \tag{3}$$

as shown in Fig. 1.

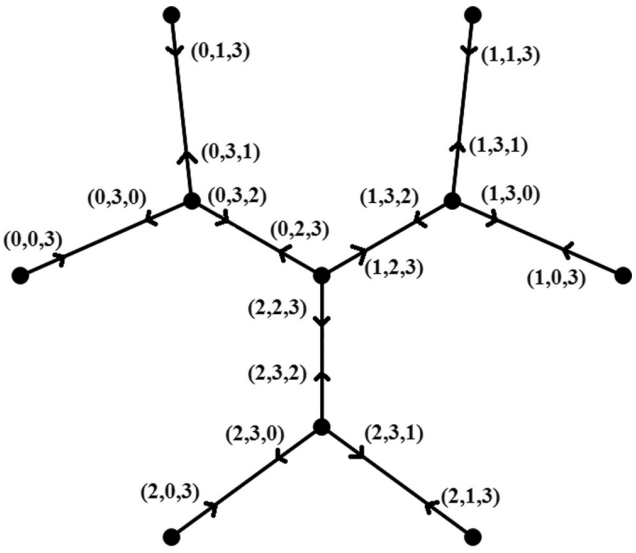
The coin operator C acts as

$$C|v,s\rangle = \begin{cases} |v,s\rangle, & \text{leaf nodes } (0 \leqslant v < n), \\ \sum_{j=0}^{n-1} \left(\frac{2}{n} - \delta_{s,j}\right)|v,j\rangle, & \text{central node } (v = n). \end{cases} \tag{4}$$

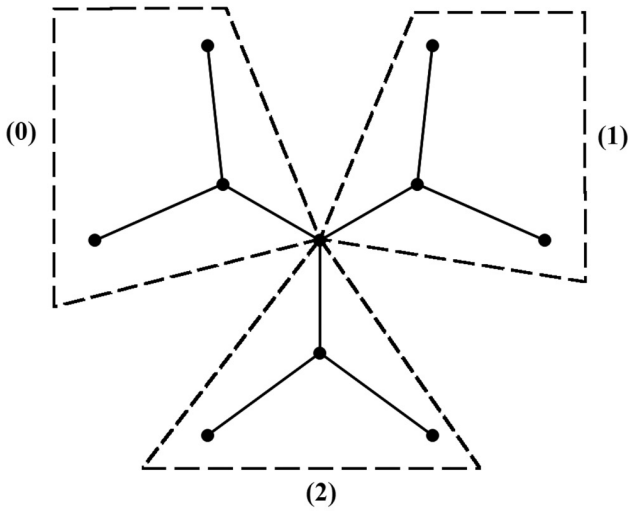
Hence, defining the walk operator as $U = SC$, after one step of the quantum walk, we have

$$U|v,s\rangle = \begin{cases} |s,v\rangle, & \text{leaf nodes } (0 \leqslant v < n), \\ \sum_{j=0}^{n-1} \left(\frac{2}{n} - \delta_{s,j}\right)|j,v\rangle, & \text{central node } (v = n). \end{cases} \tag{5}$$

The circuit implementation for an S_n graph is shown in Fig. 2, where we use $\lceil \log_2(n+1) \rceil$ qubits each for the node and subnode, with $\lceil x \rceil = \min\{n \in \mathbb{Z} | n \geqslant x\}$ being the ceiling function. The $\tilde{G}_n$ coin operator can be implemented



(a)Labeled by (t_1, n_1, s)



(b)Partitioned by (t_1)

FIG. 3. The 3CT-2 graph, partitioned into three S_3 graphs.

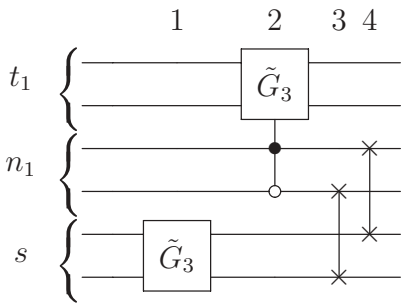


FIG. 4. Efficient circuit implementation of quantum walks on a 3CT-2 graph.

with $O(\log(n))$ two-qubit gates, and each SWAP gate requires three controlled-NOT (CNOT) gates. Hence, the circuit scales logarithmically with the size of the star graph.

III. CAYLEY TREES

The nth generation d-Cayley tree (abbreviated as dCT-n) is a tree of n levels in which all interior vertices have degree $d_i = d$. The outermost layer is called the *surface*, and all vertices on the surface are called *leaves*, with degree $d_i = 1$. The total number of vertices on the dCT-n Cayley tree is $N_v(n,d) = [d(d-1)^n - 2]/(d-2)$, and the corresponding number of distinct states is

$$N_s(n,d) = 2d[(d-1)^n - 1]/(d-2) = 2d[(d-1)^0 + (d-1)^1 + \cdots + (d-1)^{n-1}]. \quad (6)$$

Since $N_s(n,d)$ is always an integer multiple of $2d$, this suggests that a dCT-n graph may be partitioned into multiple S_d graphs, given that the number of states in an S_d graph is $2d$.

Consider, for example, the circuit construction of quantum walks on the 3CT-2 graph shown in Fig. 3. We can treat the 3CT-2 graph as three S_3 graphs with one common vertex. We label each directed edge using three parameters: the tree number t_1, the node n_1, and the subnode s. The t_1 parameter simply accounts for which S_3 graph a particular state lies on. We label each S_3 graph as $t_1 = 0,1,2$, respectively, as shown in Fig. 3(b). We can then use the method described above to treat each of the S_3 graphs separately in the circuit design. To account for the common vertex, which always has the same subnode state s, we mix the t_1 parameter at the common vertex using the coin operator, rather than the s parameter. Here we arbitrarily label the common vertex by $n_1 = 2$. For a 3CT-2 graph, there is a total of $N_v(2,3) = 10$ vertices and $N_s(2,3) = 18$ distinct states. Explicitly, the allowed states are $|t_1,n_1,s\rangle \in \{|t_1,n_1,f(n_1,d)\rangle : t_1 \in D_t, n_1 \in D_n\}$, where $D_t = \{0,1,2\}$, $D_n = \{0,1,2,3\}$, and

$$f(x,d) = \begin{cases} d, & 0 \leqslant x < d, \\ 0,\ldots,d-1, & x = d. \end{cases} \quad (7)$$

Figure 4 shows the resulting circuit implementation of the 3CT-2 graph.

Now, consider constructing the quantum circuit for the 3CT-4 graph. We can view this as adding another layer of S_3 graphs to the 3CT-2 graph, as shown in Fig. 5. To parametrize this graph, we introduce another two parameters:

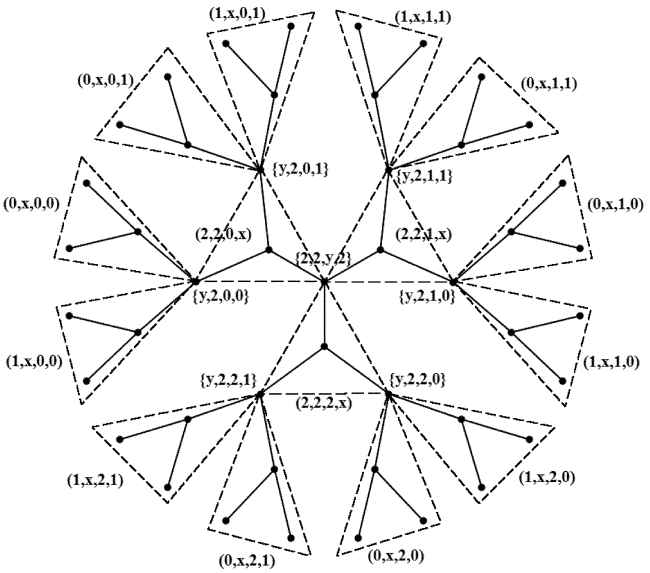


FIG. 5. The 3CT-4 graph, with S_3 graphs labeled by (t_2,n_2,t_1,n_1) and common vertices labeled by $\{t_2,n_2,t_1,n_1\}$, where $x = 0,1,2,3$ and $y = 0,1,2$. We drop the subnode label s here for simplicity, but the same convention as in Fig. 3(a) applies.

t_2 and n_2. We fix $t_2 = n_2 = 2$ for the 3CT-2 subgraph. In this layered construction of the 3CT-4 graph, we vary the t_2 and n_2 parameters when in the external layer (with t_1 and n_1 fixed based on the common vertex) and vary the t_1 and n_1 parameters when in the internal layer (with t_2 and n_2 fixed). For the 3CT-4 graph, there is a total of $N_v(4,3) = 46$ vertices and $N_s(4,3) = 90$ distinct states. Explicitly, the allowed states are $|t_2,n_2,t_1,n_1,s\rangle \in V_1 \cup V_2$, where $V_1 = \{|2,2,t_1,n_1,f(n_1,d)\rangle : t_1 \in D_t, n_1 \in D_n\}$ and $V_2 = \{|t_2,n_2,t_1,n_1,f(n_2,d)\rangle : t_2 \in D_f, n_2 \in D_n, t_1 \in D_t, n_1 \in D_f\}$ and $D_f = \{0,1\}$. The resulting circuit for implementing one quantum-walk step on the 3CT-4 graph is shown in Fig. 6. Note its similarity to Fig. 4.

In general, for a dCT-n graph with $n = 2p$, where $p \in \mathbb{Z}_+$, we label the graph using $n+1$ parameters: $t_1,\ldots,t_p$, $n_1,\ldots,n_p$, and s. The basis for the t_i parameters satisfies $|t_i| \geqslant d$, and the basis for the n_i and s

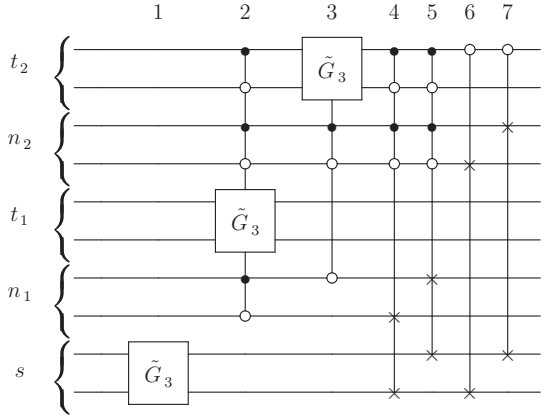


FIG. 6. Efficient circuit implementation of quantum walks on a 3CT-4 graph.

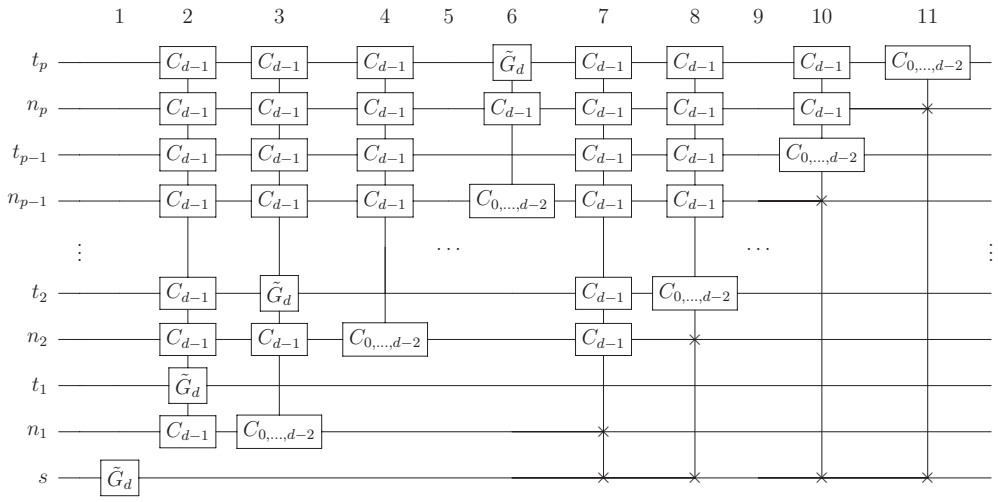


FIG. 7. Efficient circuit implementation of quantum walks on a dCT-n (even n) graph.

parameters satisfies $|n_i| = |s| \geqslant d+1$. The dCT-n graph has $N_v(n,d)$ vertices and $N_s(n,d)$ distinct states. Explicitly, the allowed states are $|\psi\rangle = |t_p,n_p,\ldots,t_1,n_1,s\rangle \in V_1 \cup \cdots \cup V_p$, where $D_f = \{0,\ldots,d-2\}$, $D_t = \{0,\ldots,d-1\}$, $D_n = \{0,\ldots,d\}$, $V_i = \{|t_p,n_p,\ldots,t_1,n_1,f(n_i,d)\rangle : t_j \in T(i,j), n_j \in N(i,j), j \in \{1,\ldots,p\}\}$, and $T(i,j)$ and $N(i,j)$ are defined as

$$T(i,j) = \begin{cases} D_t, & j = 1, \\ D_f, & 1 < j \leqslant i, \\ \{d-1\}, & j > i, \end{cases} \tag{8}$$

$$N(i,j) = \begin{cases} D_f, & 1 \leqslant j < i, \\ D_n, & j = i, \\ \{d-1\}, & j > i. \end{cases} \tag{9}$$

Let the $\tilde{G}_d[x_i]$ operator denote the $\tilde{G}_d$ operator applied to the parameter x_i. We write $\tilde{G}_d[x_i]$ as

$$\tilde{G}_d[x_i]|x_n,\ldots,x_i,\ldots,x_1\rangle = \begin{cases} \sum_{j=0}^{d-1} \left(\frac{2}{d} - \delta_{x_i,j}\right)|x_n,\ldots,j,\ldots,x_1\rangle, & x_i < d, \\ |x_n,\ldots,x_i,\ldots,x_1\rangle, & x_i \geqslant d. \end{cases} \tag{10}$$

Now write the common vertex sets as $V_{C,i} = \{|t_p,n_p,\ldots,t_1,n_1,f(n_i,d)\rangle : t_j \in T_C(i,j), n_j \in N_C(i,j), j \in \{1,\ldots,p\}\}$, where $i = 1,\ldots,p$, and

$$T_C(i,j) = \begin{cases} D_t, & j = 1,i, \\ D_f, & 1 < j < i, \\ \{d-1\}, & j > i, \end{cases} \tag{11}$$

$$N_C(i,j) = \begin{cases} D_f, & 1 \leqslant j < i, \\ \{d-1\}, & j \geqslant i. \end{cases} \tag{12}$$

Then, the coin operator acts as

$$C|\psi\rangle = \begin{cases} \tilde{G}_d[t_i]|\psi\rangle, & |\psi\rangle \in V_{C,i}, i = 1,\ldots,p, \\ \tilde{G}_d[s]|\psi\rangle, & \text{otherwise}, \end{cases} \tag{13}$$

and the shifting operator acts as

$$S|\psi\rangle = \begin{cases} |t_p,n_p,\ldots,t_1,s,n_1\rangle, & |\psi\rangle \in V_1, \\ \vdots & \vdots \\ |t_p,s,\ldots,t_1,n_1,n_p\rangle, & |\psi\rangle \in V_p. \end{cases} \tag{14}$$

We can thus implement the walk operator as $U = SC$. Figure 7 shows the circuit implementation of a quantum-walk step on a dCT-n (even n) graph. Note here the C_i gate on a parameter x indicates that the action of the entire gate is conditional on the state(s) $x = i$. For multiple conditional states (e.g., column 6 in Fig. 7), the gate can be more efficiently implemented, as shown in Fig. 8. Each of the parameters is represented by $\lceil \log_2(d+1) \rceil$ qubits.

Similarly, we can also partition a dCT-n graph with $n = 2p-1$, where $p \in \mathbb{Z}_+$, into multiple S_d graphs. We label the graph using $n+1$ parameters: $t_2,\ldots,t_p$, $n_1,\ldots,n_p$, and s. Explicitly, the allowed states are $|\psi\rangle = |t_p,n_p,\ldots,n_1,s\rangle \in V_1 \cup \cdots \cup V_p$, where $V_i = \{|t_p,n_p,\ldots,n_1,f(n_i,d)\rangle : t_j \in T(i,j), n_k \in N(i,k), j \in \{2,\ldots,p\}, k \in \{1,\ldots,p\}\}$. $T(i,j)$ is defined as earlier, while $N(i,j)$ is defined as

$$N(i,j) = \begin{cases} D_t, & j = 1 \neq i, \\ D_f, & 1 < j < i, \\ D_n, & j = i, \\ \{d-1\}, & j > i. \end{cases} \tag{15}$$

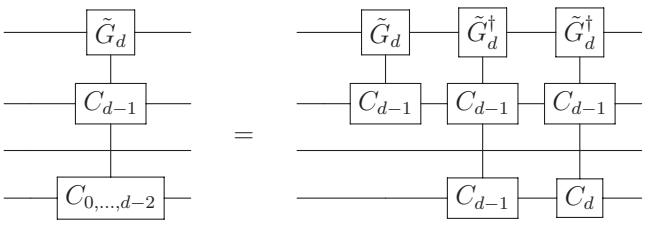


FIG. 8. Efficient implementation of a gate conditional on multiple states.

Define the common vertex sets $V_{C,i} = \{|t_p,n_p,\ldots,n_1, f(n_i,d)\rangle : t_j \in T_C(i,j), n_k \in N_C(i,k), j \in \{2,\ldots,p\}, k \in \{1,\ldots,p\}\}$, where $i = 2,\ldots,p$. $T_C(i,j)$ is defined as earlier, while $N_C(i,j)$ is defined as

$$N_C(i,j) = \begin{cases} D_t, & j = 1, \\ D_f, & 1 < j < i, \\ \{d-1\}, & j \geqslant i. \end{cases} \tag{16}$$

Then, the coin operator acts as

$$C|\psi\rangle = \begin{cases} \tilde{G}_d[t_i]|\psi\rangle, & |\psi\rangle \in V_{C,i}, i = 2,\ldots,p, \\ \tilde{G}_d[s]|\psi\rangle, & \text{otherwise}, \end{cases} \tag{17}$$

and the shifting operator acts as

$$S|\psi\rangle = \begin{cases} |t_p,n_p,\ldots,s,n_1\rangle, & |\psi\rangle \in V_1, \\ \vdots & \vdots \\ |t_p,s,\ldots,n_1,n_p\rangle, & |\psi\rangle \in V_p. \end{cases} \tag{18}$$

We can thus implement the walk operator as $U = SC$. Figure 9 shows the circuit implementation for a dCT-n (odd n) graph. This is equivalent to removing column 2 and the t_1 parameter from the circuit shown in Fig. 7 and including the $d-1$ state in the multiple conditional for each gate in the coin operator. Each of the parameters is represented by $\lceil \log_2(d+1) \rceil$ qubits.

In general, for a dCT-n graph, we require $n+1$ parameters to characterize the graph and $O(n\log(d))$ qubits for the quantum circuit. The $\tilde{G}_d$ operator requires $O(\log(d))$ two-qubit gates to implement, and thus the coin operator requires $O([n\log(d)]^2)$ two-qubit gates. The SWAP gate requires three CNOT gates to implement, and hence the shifting operator requires $O(n^2\log(d))$ two-qubit gates. In total, we require $O([n\log(d)]^2)$ two-qubit gates in the circuit to implement one step of the quantum walk. Rewriting this in terms of the number of vertices $N = N_v(n,d)$, this gives $O(\log(N)^2)$ two-qubit gates.

The circuits presented above can easily be modified to perform a quantum search of a marked vertex. For example, we mark the central vertex of the 3CT-3 graph by defining the perturbed coin operator C' as

$$C'|\psi\rangle = \begin{cases} -|\psi\rangle, & |\psi\rangle \in \{|t_2 = 2, n_2 = 2, n_1 = 3, s\rangle : s = 0,1,2\}, \\ C|\psi\rangle, & \text{otherwise}. \end{cases} \tag{19}$$

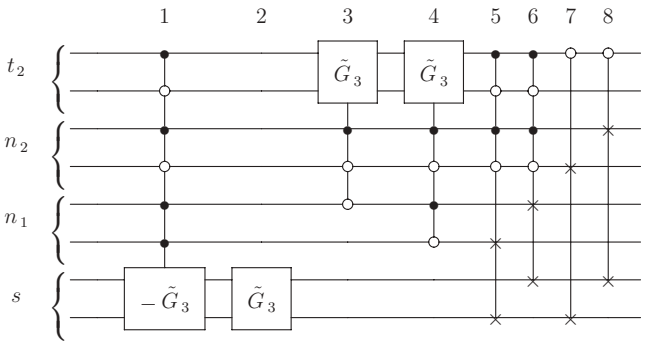


FIG. 10. Circuit implementation of quantum-walk-based search of the marked center vertex of a 3CT-3 graph.

The circuit implementing the quantum search described above is shown in Fig. 10. Column 1 performs the marking operation, and columns 2 to 8 implement the 3CT-3 graph as outlined previously. The result of the quantum search is shown in Fig. 11.

IV. COMPARISON WITH EXISTING METHODS

There is a generic approach to the design of quantum circuits for arbitrarily complex unitary operations via matrix decomposition. Earlier studies applied the standard triangularization or QR-factorization scheme with Givens rotations and Gray codes to map a quantum algorithm to a series of elementary gate operations [20,21]. Subsequently, a more efficient and versatile scheme based on the cosine-sine decomposition (CSD) was proposed [14,22–24]. The recursive

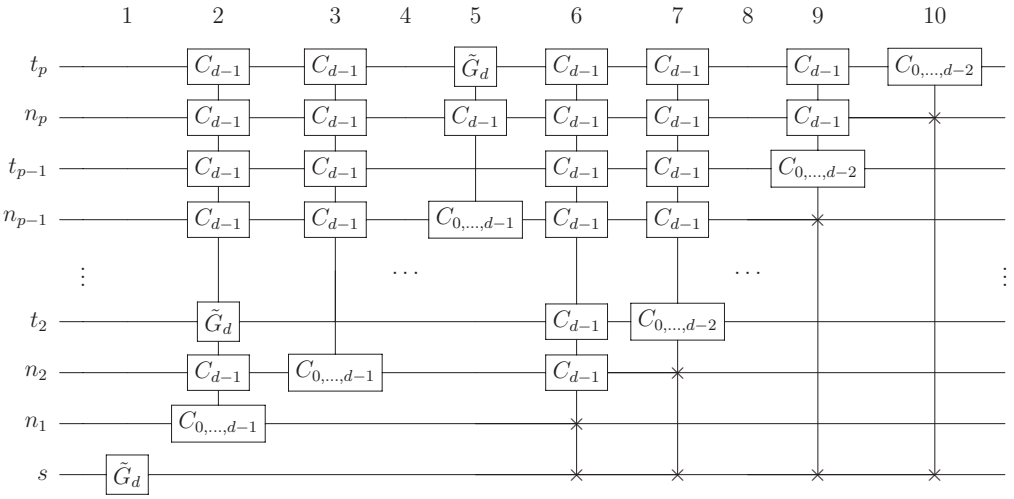


FIG. 9. Efficient circuit implementation of quantum walks on a dCT-n (odd n) graph.

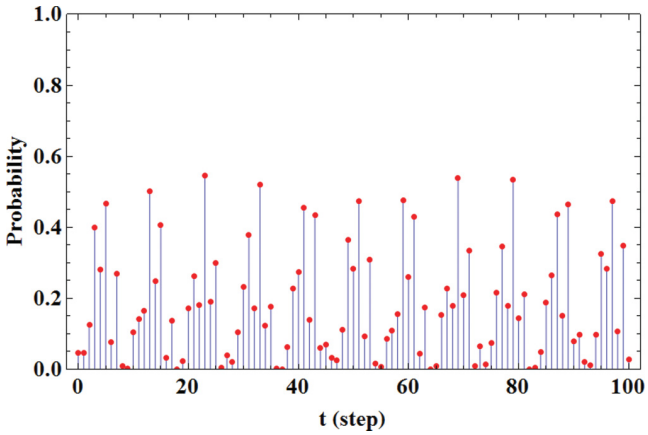


FIG. 11. (Color online) Probability at the marked center vertex of the 3CT-3 graph as a function of time steps. The initial state has equal probability at each vertex, which is also distributed equally among subnode states.

CSD procedure for arbitrary unitary operators is implemented in the QCOMPILER package [25]. Using this package, we decomposed the quantum-walk operator U obtained directly from the adjacency matrix and Grover coins. The resultant circuit implementation for the S_7 graph is shown in Fig. 12 (top), which is compared with the circuit obtained using the methods described above, shown in Fig. 12 (bottom). Clearly, the complexity of our circuit implementation is much lower than that obtained using the general recursive CSD procedure.

For a continuous-time quantum walk on an arbitrary graph, if the associated Hamiltonian is sparse, then such a walk can be efficiently simulated. Formally, the sparse Hamiltonian lemma states that "if H is a row-sparse, row-computable Hamiltonian on n^* qubits and $||H|| \leqslant \text{poly}(n^*)$, then H is simulatable" [26,27]. H is considered to be row sparse iff the number of nonzero entries in each row of H is polynomially bounded in n^*. However, we show below that a dCT-n graph is, in general, not row sparse.

A dCT-n graph has $N_s(n,d) = 2d[(d-1)^n - 1]/(d-2)$ states. For large d, we can write $N_s(n,d) \approx 2d^n$. Let us select n^* as efficiently as possible, that is, $n^* = \lceil \log_2[N_s(n,d)] \rceil$. For large d, this becomes $n^* \approx \lceil \log_2(2d^n) \rceil \approx n\log(d)$. Now, let r_i be the number of nonzero entries for row i of H. For a dCT-n graph, each vertex has a degree of at most d, so we have $r_i \leqslant d$. Hence, we require that $\exists\ \text{poly}(n^*) : d \leqslant \text{poly}(n^*)$ for row sparsity of H. This implies that $d \lesssim \text{poly}[n\log(d)]$.

For sufficiently large d or n, this becomes $d \lesssim [n\log(d)]^m$ for some finite $m > 0$. If we take n to be fixed (and finite), then as d becomes arbitrarily large, the condition for row sparsity becomes $d \lesssim [\log(d)]^m$. Defining $D = \log(d)$, this can be rewritten as $e^D \lesssim D^m$. However, this condition is clearly violated as we take D (and hence d) to be arbitrarily large. Hence, we conclude that the Hamiltonian of the dCT-n graph is, in general, not row sparse, and hence, the sparse Hamiltonian lemma cannot be applied to the dCT-n graph. Additionally, the result of simulating the (continuous-time) evolution operator e^{itH} by a quantum circuit is only approximate [27], whereas our quantum circuit provides exact results for the discrete-time quantum walk considered.

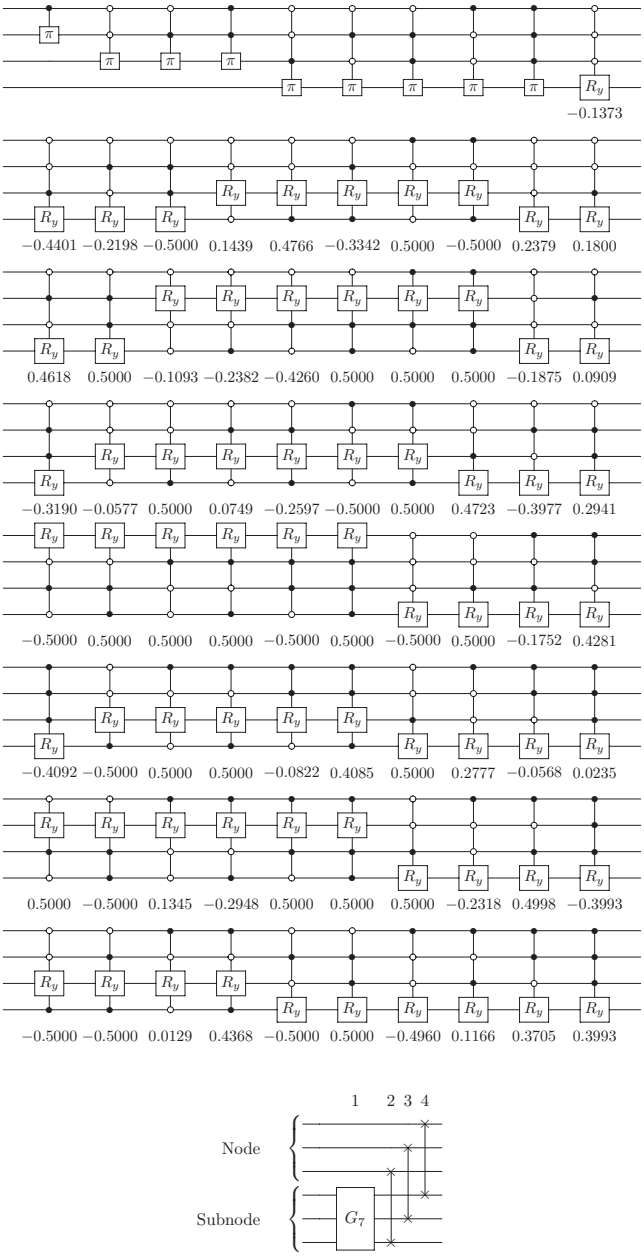


FIG. 12. Circuit implementation of quantum walks on the S_7 graph (top) from QCOMPILER [25] and (bottom) this work.

V. CONCLUSIONS

In conclusion, we have constructed exact and exponentially efficient circuit implementations for two families of non-degree-regular graphs: the star graphs of any degree and the Cayley trees of arbitrary number of layers. Existing work has been limited to circuit implementation of quantum walks on degree-regular and vertex-transitive graphs. However, from an application point of view, the most useful algorithms would more likely arise from quantum walks on graphs with certain nonregularities. For example, we note that the graphs considered here display centrality, a property important in network theory.

Constructing quantum circuits for non-degree-regular graphs is, in general, much more challenging than for degree-regular and vertex-transitive graphs since the coin space of each vertex is no longer of equal size and different vertices have a different neighboring structure. If these graphs are not vertex-transitive, how we label the node and subnode states plays an important role in achieving the most efficient quantum circuit implementation. This work opens up a new way to the construction of quantum circuits to implement quantum walks on graphs with more irregularity and complexity. Such quantum circuit implementations also provide a better estimate of the complexity and resources needed to implement a given graph, as compared to using a black-box oracle.

[1] B. Tregenna, W. Flanagan, R. Maile, and V. Kendon, New J. Phys. **5**, 83 (2003).

[2] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, in *STOC'01: Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing* (ACM, New York, 2001), pp. 50–59.

[3] E. Farhi and S. Gutmann, Phys. Rev. A **58**, 915 (1998).

[4] H. Gerhardt and J. Watrous, in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Lecture Notes in Computer Science Vol. 2764, edited by S. Arora, K. Jansen, J. D. P. Rolim, and A. Sahai (Springer, Berlin Heidelberg, 2003), pp. 290–301.

[5] J. Kempe, Contemp. Phys. **44**, 307 (2003).

[6] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, in *STOC '03: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, 2003), pp. 59–68.

[7] N. Shenvi, J. Kempe, and K. Birgitta Whaley, Phys. Rev. A **67**, 052307 (2003).

[8] A. Ambainis, in *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science* (IEEE Press, Piscataway, NJ, 2004), pp. 22–31.

[9] B. Douglas and J. B. Wang, J. Phys. A **41**, 075303 (2008).

[10] S. D. Berry and J. B. Wang, Phys. Rev. A **82**, 042333 (2010).

[11] J. K. Gamble, M. Friesen, D. Zhou, R. Joynt, and S. N. Coppersmith, Phys. Rev. A **81**, 052313 (2010).

[12] S. D. Berry and J. B. Wang, Phys. Rev. A **83**, 042317 (2011).

[13] K. Manouchehri and J. B. Wang, J. Phys. A **41**, 065304 (2008).

[14] K. Manouchehri and J. B. Wang, Phys. Rev. A **80**, 060304 (2009).

[15] B. L. Douglas and J. B. Wang, Phys. Rev. A **79**, 052335 (2009).

[16] T. Loke and J. B. Wang, Comput. Phys. Commun. **182**, 2285 (2011).

[17] S. P. Jordan and P. Wocjan, Phys. Rev. A **80**, 062301 (2009).

[18] C. Chiang, D. Nagaj, and P. Wocjan, Quantum Inf. Comput. **10**, 420 (2010).

[19] S. Berry, P. Bourke, and J. B. Wang, Comput. Phys. Commun. **182**, 2295 (2011).

[20] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurer, Phys. Rev. A **52**, 3457 (1995).

[21] D. Deutsch, A. Barenco, and A. Ekert, Proc. R. Soc. London, Ser. A **449**, 669 (1995).

[22] M. Mottonen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, Phys. Rev. Lett. **93**, 130502 (2004).

[23] V. Bergholm, J. J. Vartiainen, M. Mottonen, and M. M. Salomaa, Phys. Rev. A **71**, 052330 (2005).

[24] F. S. Khan and M. Perkowski, Theor. Comput. Sci. **367**, 336 (2006).

[25] Y. Chen and J. B. Wang, Computer Physics Communications (in press), arXiv:1208.0194 [quant-ph].

[26] D. Aharonov and A. Ta-Shma, in *STOC '03: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, 2003), pp. 20–29.

[27] A. Childs, Ph.D. thesis, Massachusetts Institute of Technology, 2004.