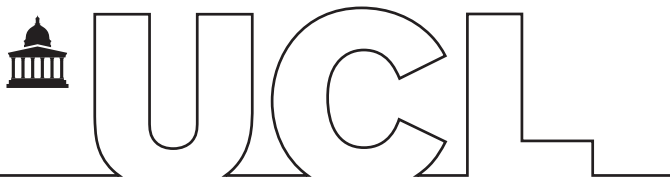

University College London

Department of Computer Science

Quantum Computation, Markov Chains and Combinatorial Optimisation

Author:
Danial Dervovic

Supervisor:
Prof. Simone Severini

Submitted in partial fulfilment for the degree of **Doctor of Philosophy**

March 15, 2020

I, Danial Dervovic, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signature

March 15, 2020

Date

Abstract

This thesis addresses two questions related to the title, *Quantum Computation, Markov Chains and Combinatorial Optimisation.*

The first question involves an algorithmic primitive of quantum computation, *quantum walks* on graphs, and its relation to Markov Chains. Quantum walks have been shown in certain cases to mix faster than their classical counterparts. Lifted Markov chains, consisting of a Markov chain on an extended state space which is projected back down to the original state space, also show considerable speedups in mixing time. We design a lifted Markov chain that in some sense simulates any quantum walk. Concretely, we construct a lifted Markov chain on a connected graph G with n vertices that mixes exactly to the average mixing distribution of a quantum walk on G. Moreover, the mixing time of this chain is the diameter of G. We then consider practical consequences of this result.

In the second part of this thesis we address a classic unsolved problem in combinatorial optimisation, *graph isomorphism*. A theorem of Kozen states that two graphs on n vertices are isomorphic if and only if there is a clique of size n in the weak modular product of the two graphs. Furthermore, a straightforward corollary of this theorem and Lovász's sandwich theorem is if the weak modular product between two graphs is perfect, then checking if the graphs are isomorphic is polynomial in n. We enumerate the necessary and sufficient conditions for the weak modular product of two simple graphs to be perfect. Interesting cases include complete multipartite graphs and disjoint unions of cliques. We find that all perfect weak modular products have factors that fall into classes of graphs for which testing isomorphism is already known to be polynomial in the number of vertices.

Open questions and further research directions are discussed.

Impact Statement

The work in this thesis is manifestly of a theoretical nature, and as such its foreseeable impact is largely academic. We answer two different questions lying in similar domains, with largely differing techniques.

We make a contribution towards a rigorous comparison between quantum walks and classical random walks. This lets one understand mixing in quantum walks in terms of classical computational resources. At the time of writing, we are in the Noisy Intermediate Scale regime of quantum computing (NISQ), where quantum machines are now becoming available for the public to use, with commensurate coverage in both the scientific and mainstream press. It is important to understand the capabilities and limitations of quantum computers, so as to not be misleading about the relationship between classical and quantum computational resources. The work described here makes this relationship more clear in our chosen context of dynamics on discrete structures. Quantum walks algorithms continue to enjoy speedups in relation to their classical counterparts, this work allows us to see why this cannot be the case for algorithms based on mixing.

The techniques used in this thesis for finding perfect and non-perfect weak modular products can be applied to understanding of the weak modular product more generally, which has not been studied in great depth due to its less than favourable theoretical properties. This work also constitutes one of the first applications of the strong perfect graph theorem that isn't ostensibly linked to perfect graphs, via its application to graph isomorphism. Indeed, the question connecting perfect weak modular product graphs and isomorphism could well have been posed decades ago, but the strong perfect graph theorem and computational methods have allowed it to be answered in practice. This combination of techniques may well prove to be fruitful for solving other problems in graph theory.

Acknowledgements

A PhD is a considerable undertaking, and for me would certainly have been impossible without the help of many great people, a subset of whom I will thank here. Firstly, I thank my supervisor Simone. He has gifted me with a friendly introduction to the scientific world and its effective practise, and has always been on hand for advice, general problem solving or just an entertaining anecdote! I am most grateful to him for teaching me to read and write mathematics and showing me the magic of graph theory. Next I must thank the wider UCLQ community, those in Computer Science in particular, for a fertile research environment where there is always someone to have an interesting discussion with about a new paper or development in the field. I have learnt an awful lot in the CS Quantum weekly group meetings, for which I must thank Toby Cubitt especially. I am grateful for my office mates over the years in room 3.14 for cultivating such a convivial atmosphere. My colleagues from Cohort 2 of the CDT in Delivering Quantum Technologies are lifelong friends and have been invaluable for the success of this PhD. Needless to say: thanks to Lopa for getting us through the tough MRes year and to the organisers of the CDT for creating such a great program. For the writing of thesis itself, I must thank Simon Apers profusely for casting his keen eye over the manuscript, and his helpful comments. The examiners Sougato Bose and Niel de Beaudrap have provided many insightful comments on the thesis that have greatly improved its clarity. Last but not least, I must thank my close family for their love and support, especially my mother and grandmother — and Clarissa, for making each day better than the last.

Contents

List of Figures

List of Tables

List of Symbols

$\mathbb{N}$	The set of natural numbers $\{1, 2, \ldots\}$
$\mathbb{Z}$	The set of integers $\{0, \pm 1, \pm 2, \ldots\}$
$\mathbb{Z}_+$ ($\mathbb{Z}_-$)	The set of nonnegative (resp. nonpositive) integers
$\mathbb{R}$	The set/field of real numbers
$\mathbb{C}$	The set/field of complex numbers
z^*	Complex conjugate of $z \in \mathbb{C}$
$\lvert z \rvert$	Modulus of $z \in \mathbb{C}$
$[k]$	The set $\{1, 2, \ldots, k\}$ for $k \in \mathbb{N}$
δ_i^j	Kronecker delta
$\mathbb{F}^d$	The vector space of d-dimensional column vectors with elements taken from the field $\mathbb{F}$
$(\mathbb{F}^d)^*$	The vector space of d-dimensional row vectors with elements taken from the field $\mathbb{F}$
$\lVert x \rVert_p$	ℓ_p-norm of the vector $x \in \mathbb{F}^d$
$\mathcal{S}(\mathbb{F}^d)$	The unit sphere in $\mathbb{F}^d$
$\mathrm{L}(\mathbb{F}^d)$	Set of linear operators on $\mathbb{F}^d$
$\mathrm{Herm}(\mathbb{F}^d)$	Set of hermitian operators on $\mathbb{F}^d$
$\mathrm{Pos}(\mathbb{F}^d)$	Set of positive semidefinite operators on $\mathbb{F}^d$
$\mathrm{U}(\mathbb{F}^d)$	Set of unitary operators on $\mathbb{F}^d$
$\mathrm{Stoch}(\mathbb{R}^d)$	Set of $d \times d$ row-stochastic matrices
$\mathrm{DStoch}(\mathbb{R}^d)$	Set of $d \times d$ doubly-stochastic matrices
$\lVert A \rVert$	Operator norm of the matrix $A \in \mathrm{L}(\mathbb{F}^d)$
$\mathrm{Tr}(A)$	Trace of the matrix $A \in \mathrm{L}(\mathbb{F}^d)$
I_d	The $d \times d$ identity matrix
X^T	Transpose of the matrix $X \in \mathrm{L}(\mathbb{F}^d)$
$X^\dagger$	Hermitian conjugate of the matrix $X \in \mathrm{L}(\mathbb{F}^d)$
$X[i, j]$	Element (i, j) of the matrix $X \in \mathrm{L}(\mathbb{F}^d)$
e_i	The i^{th} standard basis vector for $\mathbb{F}^d$

$\mathbf{1}_d$	All-ones vector in d dimensions
$\|\psi\rangle$	Quantum state "ket" vector
$\langle\psi\|$	Hermitian conjugate of the state vector $\|\psi\rangle$, "bra" vector
$\mathcal{M}_\epsilon$	ϵ-mixing time of a Markov chain
$\mathcal{M}_\epsilon^c$	ϵ-mixing time to the marginal of a c-lifted Markov chain
$\mathcal{M}_\epsilon^q$	ϵ-quantum average mixing time of a quantum walk
$V(G)$	Vertex set of the graph G
$E(G)$	Edge (resp. arc) set of the graph (resp. digraph) G
$A(G)$	Adjacency matrix of the graph G
$D(G)$	Diameter of the graph G
$G[U]$	Subgraph of G induced on the vertices $U \subseteq V(G)$
$\overline{G}$	Complement of the graph G
$G \cup H$	Union of graphs G and H
$G \uplus H$	Disjoint union of graphs G and H
$N_G(X)$	Neighbourhood of a subset of vertices $X \subseteq V(G)$ in the graph G
$G \otimes H$	Direct (or tensor) product of graphs G and H
$G \nabla H$	Weak modular product of graphs G and H
$\alpha(G)$	Independence number of the graph G
$\omega(G)$	Clique number of the graph G
$\chi(G)$	Chromatic number of the graph G
$\overline{\chi}(G)$	Clique cover number of the graph G
$\Delta(\Omega)$	Set of probability distributions over a sample space Ω
$\Pr(\omega)$	Probability of an event $\omega \in \Omega$
$\mathbf{1}_{\{\omega\}}$	Indicator variable for the event $\omega \in \Omega$

Chapter 1

Introduction

Random walks are a very general and powerful lens through which to view many naturally occurring phenomena, and also comprise a potent algorithmic tool for use in numerous applications. Quantum walks, having been discovered later, are now enjoying an analogous trajectory, finding themselves used to describe quantum dynamics acting on discrete structures alongside providing algorithmic utility when deployed on a quantum computer. Both classical and quantum random walks can be used to infer properties of the graphs they act on. One of the most basic questions one can ask about two different graphs is that of isomorphism: are their topologies the same? This question, being simple to formulate, has nonetheless resisted being fully understood by computational complexity theorists, in contrast to the overwhelming majority of other well-known combinatorial optimisation problems.

In this thesis we shall investigate in detail the interplay between quantum and classical random walks. We will also formulate a new angle of attack for the question of determining isomorphism of two graphs, providing more insight into this problem. The results are primarily mathematical, that is, they are given as theorems and proofs. There are a number of numerical experiments; indeed, many of the proofs in Chapter 4 are computer-assisted. We attempt to make clear the relationship between the mathematical models being used and their correspondence with real-world phenomena.

Some remarks on the structure of the thesis: for the remainder of this chapter we introduce the research questions the thesis seeks to answer, summarise our contributions and introduce the necessary notation for the remainder of the text. In Chapter 2, we review the literature surrounding the thesis topics which is necessary

to understand the main results in context. Chapters 3 and 4 present the main results of the thesis. In Chapter 5 we conclude with a critical analysis of the results and future research directions. To Appendix A we relegate proofs deemed to intrude excessively on the main text.

1.1 Research questions

This thesis focuses on answering two main questions.

Research Question 1 (Lifted Markov chains and quantum walks) *Which computational resources are required for a classical random walk to replicate the mixing dynamics of a quantum walk?*

Research Question 2 (Graph products and isomorphism) *Can we use easy instances of* NP*-hard problems to make progress on the* NP*-intermediate problem* GraphIsomorphism*, which has thus far eluded a polynomial-time algorithm?*

We address Research Questions 1 and 2 in Chapters 3 and 4 respectively.

For the first question, we are interested in designing a classical random walk that can in some sense simulate a quantum walk, that is, if a given quantum walk mixes to a distribution π, is there always a classical random walk that mixes to π? If there is, how long does it take to mix? How much computation time and memory is required to derive the walk parameters?

The second question asks if there is a "roundabout" route for solving GraphIsomorphism: we know that GraphIsomorphism $\in$ NP. Can we show that it lies in the "easy" subset of NP by showing that it maps to the polynomial-time solvable instances of a particular NP-hard problem? If not, what do we learn about GraphIsomorphism?

1.2 Thesis contribution

Our response to Research Question 1 is the following: For a graph G on n vertices

- We provide a classical stochastic process (called a lifted Markov chain) that mixes to any target distribution π over the vertices of G that has all positive elements, using the minimum possible number of timesteps. This builds on prior work, which we extend to a rigorous proof of correctness and more algorithmic focus, namely full accounting of the required computational resources.

- More precisely, we show that this construction requires $O(n^2 D(G))$ memory and $O(n^4 D(G))$ time to compute, where $D(G)$ is the diameter of G and G has n vertices.

- We show that for any quantum walk on an arbitrary connected graph, the average mixing distribution has all positive elements and so can be sampled from by this construction.

Our contribution to Research Question 2 takes the form:

- Following Kozen [Koz78], we formulate GraphIsomorphism as an instance of an NP-hard problem: finding the clique number of a product graph derived from the candidate graphs. We observe that for perfect graphs, finding the clique number is polynomial-time.

- Combining the previous two observations, we have that GraphIsomorphism is efficiently solvable for graphs with a perfect product. This defines an approach for GraphIsomorphism in certain restricted cases. We enumerate all perfect and non-perfect instances of this product graph to show when this proposed approach is efficient.

- We make comparisons with other GraphIsomorphism algorithms in the literature for the instances with a perfect product.

1.3 Preliminaries and notation

Much of the notation (for quantum information theory in particular) we use is taken from [Wat18]. We write the set of natural numbers $\mathbb{N} = \{1, 2, \ldots\}$. The set of nonnegative (resp. nonpositive) in integers is $\mathbb{Z}_+$ (resp. $\mathbb{Z}_-$). The set $[k] := \{1, 2, \ldots, k\}$ for $k \in \mathbb{N}$. The sets of real and complex numbers are denoted by $\mathbb{R}$ and $\mathbb{C}$ respectively, and are extended to the fields when equipped with the usual operations of addition and multiplication. We shall use the symbol $\mathbb{F}$ in place of $\mathbb{R}$ and $\mathbb{C}$ to avoid repeating definitions. We denote by $\mathbb{F}^d$ the vector space over $\mathbb{F}$ with $d \in \mathbb{N}$ elements, equipped with the inner product $\langle \cdot, \cdot \rangle$, acting as

$$\langle x, y \rangle = \sum_{i=1}^{d} (x[i])^* y[i],$$

where $x, y \in \mathbb{F}^d$, z^* the complex conjugate of $z \in \mathbb{C}$ and $x[i]$ is the i^{th} element of x.

We shall exclusively use finite-dimensional vector spaces in this thesis. Following Watrous [Wat18] we shall refer to $\mathbb{C}^d$ (resp. $\mathbb{R}^d$) as a complex (resp. real) Euclidean space, thus emphasising the difference from the more general notion of Hilbert space often employed in the quantum community[1]. Often, such a space is denoted by $\mathcal{H}$ as appropriate. We will represent a vector $x \in \mathbb{F}^d$ as a column vector. The i^{th} standard basis vector of $\mathbb{F}^d$ is denoted by e_i. The function δ_i^j is the Kronecker delta, whence $\delta_i^j = 1$ if and only if $i = j$ and zero otherwise. The vector $\mathbf{1}_d$ is the all-ones (column) vector with d elements.

The conjugate transpose of $x \in \mathbb{F}^d$, $x^\dagger$, is a row vector[2] that transforms according to the rules of matrix multiplication, for instance $x^\dagger y = \langle x, y \rangle$. We shall denote the set of d-dimensional row vectors over $\mathbb{F}$ by $(\mathbb{F}^d)^*$.

The ℓ_p norm of a vector $x \in \mathbb{F}^d$ is written as

$$\|x\|_p = \left(\sum_{i=1}^{d} |x[i]| \right)^{1/p},$$

where $|z|$ is the modulus of $z \in \mathbb{C}$. The ℓ_2 norm is commonly referred to as the *Euclidean norm* and $\|x\|_2^2 = \langle x, x \rangle$ for $x \in \mathbb{F}^d$. The *unit sphere* in $\mathbb{F}^d$, $\mathcal{S}(\mathbb{F}^d)$, is the set of unit vectors $\mathcal{S}(\mathbb{F}^d) := \left\{ u \in \mathbb{F}^d \,\middle|\, \|u\|_2 = 1 \right\}$. The *span* of a set of vectors $S = \{v_1, \ldots, v_k\}$ is defined as

$$\operatorname{span}(S) = \left\{ \sum_{i=1}^{k} \xi_i v_i \,\middle|\, v_i \in S,\ \xi_i \in \mathbb{F} \right\}.$$

The set $\mathrm{L}(\mathbb{F}^d)$ is the set of linear operators on $\mathbb{F}^d$ (i.e. $d \times d$ matrices with elements from $\mathbb{F}$). We denote the application of an operator $A \in \mathrm{L}(\mathbb{F}^d)$ to a vector $x \in \mathbb{F}^d$ by Ax. We denote the transpose of A by A^T. The $d \times d$ identity matrix is denoted by I_d, or simply I if the dimension is clear from context. The conjugate transpose of a matrix A is denoted by $A^\dagger$. A matrix A is *hermitian* if $A^\dagger = A$. We denote the set of $d \times d$ hermitian matrices by $\mathrm{Herm}(\mathbb{F}^d)$. Observe that $\mathrm{Herm}(\mathbb{R}^d)$ is

[1] A Hilbert space is a vector space with an inner product, not limited to finite dimensions. Moreover, it is required to be *complete*, a technical condition already satisfied by the spaces $\mathbb{F}^d$. To avoid trouble we will use Watrous' nomenclature of complex Euclidean spaces.

[2] Formally, $x^\dagger$ is the linear functional that maps $x \in \mathbb{F}^d$ to 1 and is an element of the *algebraic dual space* of $\mathbb{F}^d$, usually denoted by $(\mathbb{F}^d)^*$. This more abstract interpretation of $x^\dagger$ is not needed for this thesis.

the set of *symmetric* matrices, that is, $A = A^{\mathsf{T}}$ for any $A \in \mathrm{Herm}(\mathbb{R}^d)$. The set $\mathrm{U}(\mathbb{F}^d)$ is the set of unitary operators on $\mathbb{F}^d$, that is, $U \in \mathrm{U}(\mathbb{F}^d)$ if and only if $U \in \mathrm{L}(\mathbb{F}^d)$ and $U^\dagger U = I_d$. Observe that $\mathrm{U}(\mathbb{R}^d)$ is the set of *orthogonal* matrices. The set $\mathrm{Pos}(\mathbb{F}^d)$ is the set of positive semidefinite operators on $\mathbb{F}^d$, that is, the set

$$\mathrm{Pos}(\mathbb{F}^d) = \left\{ X \in \mathrm{Herm}(\mathbb{F}^d) \,\middle|\, \langle y, Xy \rangle \geq 0 \text{ for all } y \in \mathbb{F}^d \right\},$$

where the field $\mathbb{F}$ is usually clear from context. We write $A \succeq B$ for $A, B \in \mathrm{L}(\mathbb{F}^d)$ if $A - B \in \mathrm{Pos}(\mathbb{F}^d)$. The trace of an operator $A \in \mathrm{L}(\mathbb{F}^d)$, $\mathrm{Tr}(A)$, is the sum of its diagonal elements. The set $\mathrm{Stoch}(\mathbb{R}^d)$ is the set of row-stochastic matrices, the matrices whose rows sum to unity and whose elements are all nonnegative. The set $\mathrm{DStoch}(\mathbb{R}^d)$ is the set of doubly-stochastic matrices, the row-stochastic matrices whose columns also sum to unity. The *operator norm* of a matrix $X \in \mathrm{L}(\mathbb{F}^d)$, $\|X\|$, is given by

$$\|X\| := \sup \left\{ \frac{\|Ax\|_2}{\|x\|_2} \,\middle|\, x \in \mathbb{F}^d,\ x \neq 0 \right\}.$$

Where applicable, we write the space of linear operators from $\mathbb{F}^n$ to $\mathbb{F}^m$ as $\mathrm{L}(\mathbb{F}^n, \mathbb{F}^m)$ and do the same, *mutatis mutandis*, for other sets of operators. Concretely, an element of $\mathrm{L}(\mathbb{F}^n, \mathbb{F}^m)$ is an $m \times n$ matrix. The *column space* of a matrix is the span of its columns. The *tensor product*, or *Kronecker product*, of the matrices $A \in \mathrm{L}(\mathbb{F}^m, \mathbb{F}^n)$, $B \in \mathrm{L}(\mathbb{F}^p, \mathbb{F}^q)$, denoted by $A \otimes B$ is the $mp \times nq$ block matrix

$$A \otimes B := \begin{bmatrix} A[1,1] \cdot B & \cdots & A[1,n] \cdot B \\ \vdots & \ddots & \vdots \\ A[m,1] \cdot B & \cdots & A[m,n] \cdot B \end{bmatrix}.$$

A *graph* $G = (V(G), E(G))$ consists of a set $V(G)$ of n vertices and a set of edges $E(G) \subseteq \{\{x, x'\} : x, x' \in V(G),\ x \neq x'\}$. A *directed graph* G, or *digraph* for short, has directed edges or *arcs*, that is, $E(G) \subseteq V(G) \times V(G)$. Note that we will allow self-loops of the form (x, x) for $x \in V(G)$ in the case of directed graphs. We write $x \sim y$ to denote that vertices x and y are neighbours, that is, they are connected by an edge. We denote by $\overline{G}$ the *complement* of G, whence $V(\overline{G}) = V(G)$ and $x \sim x'$ in $\overline{G}$ if and only if $x \not\sim x'$ in G and $x \neq x'$. The *union* of graphs G and H is the graph $G \cup H$ with vertex set $V(G \cup H) = V(G) \cup V(H)$ and edge set

$E(G \cup H) = E(G) \cup E(H)$. The *disjoint union* of graphs G and H is the graph $G \uplus H$ with vertex set $V(G \uplus H) = V(G) \uplus V(H)$ and edge set $E(G \uplus H) = E(G) \uplus E(H)$. The graph $G \uplus H$ is to be interpreted as the graphs G and H drawn next to one another. For a graph G, we denote the disjoint union of k copies of G by kG.

A graph G' is a *subgraph* of another graph G, $G' \subseteq G$, if and only if $V(G') \subseteq V(G)$, and

$$E(G') \subseteq E(G) \text{ and for all } \{x,x'\} \in E(G'), \quad x,x' \in V(G'). \tag{1.1}$$

Suppose we have a subset of the vertices $U \subseteq V(G)$. An *induced subgraph* of G, $G[U]$, is the graph with vertex set $V(G[U]) = U$ and edge set

$$\{\{x,x'\} \,|\, x,x' \in U,\ \{x,x'\} \in E(G)\}. \tag{1.2}$$

We say $G[U]$ is *induced* by $U \subseteq V(G)$.

A graph is called *k-regular* if every vertex has k neighbours, where $k \in [|V(G)-1|]$. We call G a *symmetric* directed graph if $(x,y) \in E(G) \Leftrightarrow (y,x) \in E(G)$ for all $x,y \in V(G)$ and say that x and y are *adjacent*. We denote by $x \sim y$ the adjacency of two vertices x, y. The *neighbourhood* of a vertex, $N(x)$, is the set of all its neighbours. Moreover, the neighbourhood $N_G(X)$ of a subset of vertices $X \subseteq V(G)$ is the union of the neighbourhoods of the vertices in X, minus elements of X themselves, that is,

$$N_G(X) = \{y \,|\, y \in N(x),\ x \in X,\ y \notin X\}.$$

We omit the subscript when the relevant graph is clear from context. A *path* between vertices x and y is sequence of vertices beginning with x and finishing with y such that each vertex in the path is distinct and successive vertices are adjacent. A *walk* between vertices x and y is a path that can repeat vertices. A *cycle* is a path that starts and ends at the same vertex. A *circuit* is a walk that starts and ends at the same vertex. A graph is *acyclic* if it contains no cycle as a subgraph.

A graph is *connected* if there is a path between any two distinct vertices. A *connected component* of a graph G is a subgraph that is connected, and is maximal

with respect to this property; no additional edges or vertices from G can be included in the subgraph without breaking its property of being connected. A digraph is *strongly connected* if there is a directed path between every ordered pair of vertices.

A *tree* is an undirected graph in which any two vertices are connected by exactly one path, or equivalently a graph that is connected and acyclic. A *spanning tree* T of an undirected graph G is a subgraph that is a tree including all of the vertices of G, with minimum possible number of edges.

The *distance*, $d(u,v)$, between the nodes u and v in G is the shortest length path between them. The *diameter* of G, $D(G)$, is the greatest distance between any pair of vertices in G, or rather

$$D(G) := \max_{u,v \in V(G)} d(u,v). \tag{1.3}$$

A *weak homomorphism* $\varphi : G \to H$ is a map $\varphi : V(G) \to V(H)$ for which $\{u,v\} \in E(G)$ implies either $\{\varphi(u),\varphi(v)\} \in E(H)$ or $\varphi(u) = \varphi(v)$. Recall that a homomorphism is a map from G to H satisfying only the first condition.

Let Ω be a sample space. The probability of an event $\omega \in \Omega$ is denoted by $\Pr(\omega)$. The conditional probability of an event $\omega \in \Omega$ given $\nu \in \Omega$ is denoted by $\Pr(\omega \,|\, \nu)$. A probability distribution over a finite sample space Ω is represented by a row vector $p \in \mathbb{R}^{|\Omega|*}$. We denote the set of all probability distributions over Ω by $\Delta(\Omega)$. We take a random variable[3] X over Ω with distribution $p \in \Delta(\Omega)$ simply to mean a variable whose value is randomly distributed according to p.

For the functions $f, g : \mathbb{R} \to \mathbb{R}$, we write $f(x) = O(g(x))$ if and only if $f(x)$ is at most a constant multiple of $g(x)$ as $x \to \infty$. Similarly, we write $f(x) = \Omega(g(x))$ if $g(x) = O(f(x))$. Intuitively, $f(x) = O(g(x))$ means that g asymptotically *upper bounds* f and $f(x) = \Omega(g(x))$ means that g asymptotically *lower bounds* f. We write $f(x) = \Theta(g(x))$ if and only if $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$.

We will require some basic notions from computational complexity theory as a reference, a complete introduction is given in [AB09]. The computational complexity class P is the set of all decision (yes/no) problems that can be solved by a deterministic Turing machine using a polynomial amount of computation time. The class NP is the set of decision problems, for which a solution is verifiable in polynomial

[3] The modern measure-theoretic definition of a random variable will not be needed in this thesis.

time. This set of problems are in practise computationally hard to solve, that is, superpolynomial in the input size. It is widely believed [Aar16], and there is much evidence to suggest that, $\mathsf{P} \neq \mathsf{NP}$. If a computational problem is NP-hard, it is at least as difficult to solve as any problem in NP, modulo polynomial-time reductions. If a problem is NP-complete, it is both NP-hard and contained in NP.

Chapter 2

Background

In this section we present the necessary precursory material for the thesis contributions in Chapters 3 and 4, namely a review of the surrounding literature.

2.1 Quantum computation

In this section, we introduce *gate-model quantum computation*, the most popular model for quantum computing theory. For a complete introduction to quantum computing we refer the reader to [NC10]. We note that the content of thesis sits squarely within the *theory* of quantum computation, ignoring experimental or implementation considerations. There are currently many avenues being explored for implementing gate-model quantum computers, each with their own strengths and weaknesses. At the time of writing, machines based on superconducting qubits [DWM04] are considered to have made the most progress, although there are many other approaches, including but not limited to: trapped ions [CZ95], linear optics [KLM01] and donor qubits in silicon [Kan98]. A good entry point for this research at the time of writing can be found at [NAoSM19]. There are many other models of quantum computation that have received attention in the literature that are polynomially equivalent to the gate model, such as adiabatic quantum computation [AvDK$^+$08] and measurement based quantum computation [RBB03].

2.1.1 Quantum Mechanics and the Gate Model

The fundamental unit of information is the bit, which takes the values 0 and 1. In the circuit model of classical computing, the input is a classical bit string which through the application of a circuit is transformed to an output bit string. The circuit consists of a finite number of classical gates picked from a universal gate set

such as $\{\text{NAND}\}$.

In quantum computing the fundamental unit of information is a qubit, which is represented [1] by a two-dimensional unit (column) vector from $\mathcal{S}(\mathbb{C}^2)$. A qubit state $|\psi\rangle$ is given by $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, such that $\alpha_0, \alpha_1 \in \mathbb{C}$ and $|\alpha_0|^2 + |\alpha_1^2| = 1$, where $|0\rangle$ and $|1\rangle$ are the basis vectors of $\mathbb{C}^2$.

At this point we have introduced the *Dirac bra-ket* notation for quantum states, which we shall use only when a vector from $\mathbb{C}^d$ is explicitly representing a quantum state. Ordinary vectors will have no such designation. Note that otherwise we treat a vector $|\psi\rangle \in \mathbb{C}^d$ exactly the same as a vector $\psi \in \mathbb{C}^d$, the notation merely serves as a reminder of the intended interpretation of the vector at hand. We shall denote by $\langle\psi|$ the conjugate-transpose of the state vector $|\psi\rangle$.

Multiple qubits are combined using the tensor product, that is, for two qubits separately in states $|\phi\rangle, |\psi\rangle$, their joint state is given by $|\phi\rangle \otimes |\psi\rangle$. Thus, an n-qubit quantum state can be expressed as a vector in $\mathcal{S}(\mathbb{C}^{2^n})$

$$|\psi\rangle = \sum_{i_1,\ldots,i_n} \alpha_{i_1\ldots i_n} |i_1 \ldots i_n\rangle, \tag{2.1}$$

where $i_k \in \{0,1\}$, $\sum_{i_1\ldots i_n} |\alpha_{i_1\ldots i_n}|^2 = 1$ and $|i_1 \ldots i_n\rangle \equiv |i_1\rangle \otimes \cdots \otimes |i_n\rangle$. We call the basis $\{|i_1 \ldots i_n\rangle \,|\, i_k \in \{0,1\}\}$ the *computational basis*, as each basis vector is described by a string of n bits.

There are two types of operation we can perform on a d-dimensional quantum state: *unitary operators* and *measurements*. A unitary operator $U \in \mathrm{U}(\mathbb{C}^d)$ has the property that $UU^\dagger = U^\dagger U = I_d$, *i.e.* its inverse is given by the hermitian conjugate. Furthermore, this implies that the operator is norm-preserving, that is, unitary operators map quantum states to quantum states. A unitary operator on n qubits can be expressed as matrix of dimension $2^n \times 2^n$. Moreover, we have that unitary operators are closed under composition. Often, especially when describing physical systems, unitary dynamics are generated by a *Hamiltonian* $H \in \mathrm{Herm}(\mathbb{C}^d)$, that is, $U = \mathrm{e}^{\mathrm{i}Ht}$, where $t \in [0,\infty]$ represents time passed and matrix exponentiation is defined as a power series. A measurement is described by a collection of operators

[1] Strictly speaking, a qubit state $|\psi\rangle$ is the equivalence class of vectors $\left\{ |\psi'\rangle \,\middle|\, |\psi'\rangle = \mathrm{e}^{\mathrm{i}\theta} |\psi\rangle,\ \theta \in [0, 2\pi) \right\}$, formally called a *ray* in the complex projective space $\mathbb{C}P^1$. This more formal description won't be needed in this thesis, as will become clear shortly.

$\{M_m\}$, where $M_m \in \mathrm{Herm}(\mathbb{C}^{2^n})$ and the index m indicates a given measurement outcome. The operators M_m satisfy the *completeness equation*, $\sum_m M_m^\dagger M_m = I_{2^n}$. For a quantum state $|\psi\rangle$, the probability of measuring outcome m is given by

$$\Pr(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle \tag{2.2}$$

and the resulting quantum state is

$$\frac{M_m |\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}}. \tag{2.3}$$

The completeness equation encodes the fact that measurement probabilities over all outcomes sum to unity. A *computational basis measurement*, $\{M_x\}$ for $x \in \{0,1\}^n$ consists of the operators $M_x = |x\rangle\langle x|$, the projectors onto the computational basis states. Notice from Eq. 2.2 that states differing by a *global phase*, that is, states that are equal up to overall multiplication by a complex phase factor $\mathrm{e}^{\mathrm{i}\theta}$ for $\theta \in [0, 2\pi)$, are physically indistinguishable since they have the same measurement outcome probabilities.

In the circuit model of quantum computation, we are given an input $x \in \{0,1\}^n$ which is a classical bit string. The first step is to prepare an m-qubit quantum input state $|\psi\rangle$, where $m = O(\mathrm{poly}(n))$. A unitary operator U is then applied to $|\psi\rangle$, and finally the output state is measured in the computational basis. This returns a classical bit string $y \in \{0,1\}^m$ with probability $|\langle y| U |\psi\rangle|^2$.

This completes the description of the model of quantum computation used in this thesis, the gate model, which is widely adopted by the quantum algorithms community. The model is assumed to have no limits on the number of qubits used and it is assumed that every unitary $U \in \mathrm{U}(\mathbb{C}^{2^n})$ is realisable. It is not obvious that the gate model as described here is physically reasonable. It falls outside of the scope of this thesis to provide such a justification, but nonetheless I briefly describe the ongoing research in this area and provide some references for the unconvinced reader.

2.1.2 Practical Considerations

In practice, a quantum computer will consist of a small set of one and two-qubit unitary operators, which we call gates, that can be applied to the qubits in the

machine. A set of quantum gates is said to be *universal* if any unitary operator can be approximated 'well-enough' using only gates from the set. More precisely, a set of of one and two-qubit gates S is universal if any unitary operator U can be decomposed into the sequence, or *circuit*, $U_L U_{L-1} \dots U_1$, such that $||U - U_L \dots U_1|| \leq \varepsilon$ for any $\varepsilon > 0$ and some $L \in \mathbb{N}$, where the $U_k \in S$. We call L the *depth* of the circuit. There are many such universal gate sets, such as $\{$TOFFOLI, HADAMARD$\}$, or $\{U, \text{CNOT} \mid U \in \mathrm{U}(\mathbb{C}^2)\}$. A picture of a quantum circuit can be found in Figure 2.4a. Thus, any arbitrary unitary operator U can be implemented given a universal set of gates. But can this be done efficiently?

The Solovay-Kitaev theorem (see [NC10, Appendix 3]) guarantees that exponential approximation accuracy can be achieved with a polylogarithmic overhead on the number of gates used in a computation. This allows us to assume that we can use any unitary in a quantum computation, without significantly increasing the runtime complexity of the computation. A quantum algorithm is then specified as a series of unitaries to be applied to an initial state, with a measurement scheme for extracting a result from the computation.

Quantum computers are microscopic physical objects, and are thus subject to myriad sources of noise, regardless of the physical implementation of the qubits, gates, measurements and architecture. This naturally poses a risk to destroy the results of any computation. Preskill [Pre13] has shown sufficient conditions on the noise impinging a given computation for scalable quantum computation to be possible, contingent on assumptions widely considered to be physically reasonable. The theory of fault-tolerant quantum computation is well-developed and is still an active field of research. A good, recent review can be found at [CTV17]. Broadly speaking, at present there are schemes that allow one to carry out arbitrary quantum computations, but these schemes require significant practical overhead in being carried out. Reducing this overhead is a core challenge for the field.

2.1.3 Relation to classical computation

There are many quantum algorithms that achieve theoretical superiority over their classical counterparts, usually in terms of runtime but sometimes also in memory usage. The two most famous examples are Shor's algorithm [Sho97] and Grover's algorithm [Gro97]. The former is a polynomial time algorithm for factoring integers,

providing an exponential speedup over the best known classical algorithm and the latter an algorithm for unstructured search on N elements, with quadratic speedup over the classical $\Theta(N)$ bound. A good overview of quantum algorithms can be found at [MCvD$^+$16]. We discuss quantum algorithms based on *quantum walks* in more detail in Section 2.4.

One can also compare classical and quantum computers from a complexity-theoretic point of view. The class BQP is defined as the class of decision problems solvable by a quantum computer in polynomial time, with an error probability of at most $1/3$ for all instances. It is known that $\mathsf{P} \subseteq \mathsf{BQP}$ and $\mathsf{BQP} \subseteq \mathsf{PSPACE}$ and widely conjectured that $\mathsf{P} \subset \mathsf{BQP}$, meaning there are likely many problems solvable in polynomial time on a quantum computer for which a classical computer requires superpolynomial time. In this direction, Bravyi, Gosset and König recently showed the first unconditional separation between classical and quantum computation [BGK18]. The separation shown is logarithmic, which is still consistent with $\mathsf{P} = \mathsf{BQP}$. It is also widely conjectured that $\mathsf{NP} \not\subseteq \mathsf{BQP}$ and $\mathsf{BQP} \not\subseteq \mathsf{NP}$. Thus, a quantum computer cannot solve NP-complete problems in polynomial time, although one does get a generic quadratic speedup over brute force search via Grover's algorithm, taking the search space as instances to the problem at hand. A survey of quantum complexity theory can be found at [Wat08].

2.2 Markov chains

Markov first studied the stochastic processes that were eventually given his name in 1906. In this section we shall define a Markov chain, and discuss its mixing properties, along with more modern developments.

Consider a directed graph $G = (V(G), E(G))$ on $n \in \mathbb{N}$ vertices with vertex set $V(G) = [n]$ and arc set $E(G) \subseteq V(G) \times V(G)$. We can define a *discrete-time Markov chain* M_G on the vertices of G as follows: Let $X(t)$ be a random variable, where $X(t) \in V(G)$, for all $t \in \mathbb{Z}_+$. The Markov chain M_G is the sequence of states $(X(0), X(1), \ldots)$, that additionally satisfies the following properties: *i.* The initial state $X(0)$ is selected according to some fixed probability distribution $p^{(0)} \in \Delta(V(G))$. *ii.* The probability of observing state $X(t)$ is independent of all

previous states, apart from its immediate predecessor, $X(t-1)$, that is

$$\Pr(X(t)\,|\,X(t-1),X(t-2),\ldots,X(0)) = \Pr(X(t)\,|\,X(t-1))$$

for all $t \in \mathbb{Z}_+$. *iii.* The *transition probability* between states $i,j \in V(G)$, $\Pr(X(t+1)=i\,|\,X(t)=j) > 0$ if and only if $(i,j) \in E(G)$ and is zero otherwise.

From the above definition, each arc (i,j) in G has an associated transition probability from vertex i to vertex j. The arc probabilities must be nonnegative and the sum of the probabilities leaving a given vertex must equal one. These transition probabilities are listed in the matrix $P \in \mathbb{R}^{n\times n}$, where

$$P[i,j] = \Pr(X(t+1)=j\,|\,X(t)=i)\,,\ i,j \in [n]. \tag{2.4}$$

This matrix must satisfy

$$P[i,j] \geq 0, \qquad P\mathbf{1} = \mathbf{1}, \qquad P[i,j] = 0 \Longleftarrow (i,j) \notin E(G). \tag{2.5}$$

The conditions in Eq. (2.5) state that P must be a row-stochastic matrix with support only on elements corresponding to arcs in the the graph G.

At a time $t \in \mathbb{Z}_+$, the distribution over states will be given by the row vector

$$p^{(t)} = p^{(0)}P^t. \tag{2.6}$$

In other words, $X(t)$ is a random variable distributed according to $p^{(t)} = p^{(0)}P^t$.

We can justify Eq.(2.6) formally by the following claim (we defer the proof to the appendix (Claim A.1)), which states that the only linear maps taking probability distributions to probability distributions are stochastic.

Claim 2.1 *Let Ω be a finite sample space and suppose $P \in \mathrm{L}(\mathbb{R}^{|\Omega|})$ is a linear map. Then P maps every distribution $\pi \in \Delta(\Omega)$ to another distribution $\pi' \in \Delta(\Omega)$, that is, $\pi' = \pi P$, only if $P \in \mathrm{Stoch}(\mathbb{R}^{|\Omega|})$. Moreover, let $\pi'' = \pi P'$ for $P' \in \mathrm{Stoch}(\mathbb{R}^{|\Omega|})$ and $\pi \in \Delta(\Omega)$. Then, $\pi'' \in \Delta(\Omega)$.*

We see that a Markov chain M_G on a directed graph G can be completely characterised by the transition matrix P and the initial distribution over states $p^{(0)}$,

so we shall use the shorthand $M_G = (P, p^{(0)})$.

Definition 2.1. Let G be a digraph. Moreover, let $P \in \text{Stoch}(\mathbb{R}^{|V(G)|})$ and $p^{(0)} \in \Delta(V(G))$. We say the tuple $M_G := (P, p^{(0)})$ is a *Markov chain* over G.

Note that we will use the terms 'Markov chain' and 'random walk' interchangeably. A *simple* random walk is a Markov chain whose transition probabilities from any vertex are the same to all of its neighbours. Often a Markov chain starts at a particular vertex j, in which case $p^{(0)} = e_j$.

2.2.1 Mixing

Suppose we have a Markov chain $M_G = (P, p^{(0)})$ over a finite directed graph G and a probability distribution on the states of M_G, π such that $\pi P = \pi$. Then we call π a *stationary distribution* of M_G. Indeed, π exists and is unique if M_G is *irreducible* and *aperiodic* [LPW09]. Moreover, an irreducible, aperiodic Markov chain always converges to the stationary distribution, that is, $\lim_{t\to\infty} p^{(0)} P^t = \pi$; a result known as the *convergence theorem* in the literature. Moreover, all of the elements of π in this case are strictly positive. Irreducibility of M_G is equivalent to saying that the graph G is strongly connected. The chain M_G is aperiodic if there exists some time T_0 such that for all $t \geq T_0$ and all vertices $i, j \in V(G)$, $P^t[i,j] > 0$. A Markov chain that is irreducible and aperiodic is called *ergodic*. We say that an irreducible Markov chain is *reversible* if for all $i, j \in \Omega$ the transition matrix satisfies $\pi[i]P[i,j] = \pi[j]P[j,i]$.

We now define the *ϵ-mixing time*, $\mathcal{M}_\epsilon$, of M_G, for $\epsilon > 0$,

$$\mathcal{M}_\epsilon = \max_{p^{(0)} \in \mathcal{P}} \min_{T \in \mathbb{Z}_+} \left\{ T \,\middle|\, \forall t \geq T,\ \left\| p^{(0)} P^t - \pi \right\|_{TV} \leq \epsilon \right\}, \tag{2.7}$$

where $\|\cdot\|_{TV}$ is the total variation distance[2] between two distributions π and κ and $\mathcal{P} \subseteq \Delta(V(G))$ is the allowed domain of initial starting states. Intuitively, the mixing time is the number of steps it takes for an arbitrary starting state to be ϵ-close in total variation distance to the stationary distribution in the worst case. Typically, $\mathcal{P}$ is the set of distributions with all probability mass on one and only one state, i.e. $\mathcal{P} = \{e_i \,|\, i \in V(G)\}$. We can do this without loss of generality, since it can be shown that $\max_{x \in V(G)} \|e_x P^t - \pi\|_{TV} = \sup_{\mu \in \Delta(V(G))} \|\mu P^t - \pi\|_{TV}$ [LPW09, Exercise 4.1].

[2] The total variation distance of probability distributions $\pi, \kappa \in \Delta(\Omega)$ over finite Ω is $\|\kappa - \pi\|_{TV} = \frac{1}{2} \sum_{x \in \Omega} |\pi[x] - \kappa[x]|$

We say that the chain M_G has *mixed* at a time T if $\left\|p^{(0)}P^T - \pi\right\|_{TV} \leq \epsilon$; from submultiplicativity of the ℓ_1-norm the chain will be mixed for all $t \geq T$. By convention, we shall often take $\epsilon = 1/4$. Indeed, from [LPW09, Eq. (4.36)] we can get a bound for arbitrary $0 < \epsilon < 1/4$,

$$\mathcal{M}_\epsilon \leq \left\lceil \log_2 \frac{1}{\epsilon} \right\rceil \mathcal{M}_{1/4}. \tag{2.8}$$

The mixing time is strongly related to a topological property of the Markov chain called the *conductance*. We must first define the conductance of a Markov chain $(P,p^{(0)})$ on G. For a subset $X \subseteq V(G)$ let $\pi(X) = \sum_{i \in X} \pi[i]$, where π is the stationary distribution under P. The conductance $\Phi(P)$ of P is defined as

$$\Phi(P) = \min_{X \subset V(G); \pi(X) \leq \frac{1}{2}} \frac{\sum_{i \in X, j \notin X} P[i,j]\pi[i]}{\pi(X)}. \tag{2.9}$$

Often, the numerator of Eq. (2.9) is referred to as the *flow* through X and the denominator as the *capacity* of X. The conductance gives a measure of how hard it is to leave a small subset of vertices, minimised over the graph (where by small we mean fewer than half of the vertices). Given only a graph G and a target stationary distribution π, the *conductance* Φ *of* G *towards* π is the maximum of $\Phi(P)$ over all row-stochastic P that satisfy the locality constraints of G and whose unique stationary distribution is π.

Sinclair provided the following relationship between the conductance and mixing time [Sin93, Eq. (2.13)]

$$\frac{1-2\Phi(P)}{2\Phi(P)} \log \frac{1}{\epsilon} \leq \mathcal{M}_\epsilon \leq \frac{2}{\Phi(P)^2} \left(\log \frac{1}{\epsilon} + \log \left(\frac{1}{\min_i \pi[i]} \right) \right). \tag{2.10}$$

Observation 2.1. The mixing time of a Markov chain is bounded between $\Omega(1/\Phi)$ and $O(1/\Phi^2)$ (for fixed $\min_i \pi[i]$).

2.2.2 Markov chain Monte Carlo (MCMC)

Sampling from a desired probability distribution over a given state space is an important computational task, used in many diverse fields. Markov chain methods have proven to be widely successful in this domain being used for applications such as approximating the permanent of a matrix [Sin93], machine learning [AdFDJ03]

and analysing the performance of distributed systems [MK82]. In practise, many approaches suffer from a lack of provable upper bounds on the time it takes to draw samples. One such class of methods is Markov Chain Monte Carlo (MCMC), where a specific random walk is conducted on the state space of interest for some set number of timesteps, then the position of the walker is measured. Often the user of an algorithm in the MCMC framework is unsure if the chain has mixed, that is, is sampling the walker's position equivalent to sampling the desired distribution? More precisely, is the distribution over the vertices close in total variation distance to the stationary distribution of the Markov chain? In most cases, the answer to this question is unknown, the practitioner empirically determines a favourable time to run the chain for, without any theoretical guarantee of closeness to the desired distribution [KF09]. For a survey of the mathematics of MCMC, see [Dia08].

One of the first and most celebrated algorithms in MCMC is the *Metropolis algorithm*, which we briefly describe now. The Metropolis algorithm is a way of taking a Markov chain and modifying the transition probabilities such that it converges to any desired stationary distribution π. This modification is called the *Metropolis chain*, which is discussed at length in [LPW09, Chapter 3]. Suppose we have a Markov chain $(\Psi, p^{(0)})$ and we wish to impose that it mixes to the designed stationary distribution π. Can we modify the transition matrix Ψ to accommodate this? Indeed we can; the resulting chain is the Metropolis chain of Ψ, with transition matrix elements

$$P[x,y] = \begin{cases} \Psi[x,y] \cdot \min\left\{ \frac{\pi[y]}{\pi[x]} \frac{\Psi[y,x]}{\Psi[x,y]}, 1 \right\}, & \text{if } y \neq x; \\ 1 - \sum_{z:z\neq x} \left[\Psi[x,z] \cdot \min\left\{ \frac{\pi[z]}{\pi[x]} \frac{\Psi[z,x]}{\Psi[x,z]}, 1 \right\} \right], & \text{if } y = x. \end{cases} \tag{2.11}$$

One can easily verify that the chain $(P, p^{(0)})$ has stationary distribution π. Moreover, the chain will mix to π since it is aperiodic and irreducible by construction, although we have no guarantees on the mixing time, apart from the loose upper bound $O\left(\frac{1}{1-\lambda_2}\right)$, where λ_2 is the second largest eigenvalue of the transition matrix P. In practise the topology of the underlying graph greatly affects the mixing time so must be chosen with due care.

2.2.3 Fastest mixing Markov chain

It is well known that the mixing time of a reversible, ergodic Markov chain is upper bounded by the reciprocal of the spectral gap of the transition matrix [LPW09, Theorem 12.3] This is a monotone function of the *second largest eigenvalue modulus* (SLEM) of the transition matrix, P:

$$\mu(P) = \max_{i=2,\dots,n} |\lambda_i(P)| = \max\{\lambda_2(P), -\lambda_n(P)\},$$

where the eigenvalues of P satisfy $1 = \lambda_1(P) \geq \lambda_2(P) \geq \cdots \geq \lambda_n(P) \geq -1$.

Thus, minimising the SLEM minimises an upper bound on the mixing time. Boyd *et. al.* [BDX04] use this insight to define an optimisation problem the *Fastest Mixing Markov Chain Problem* (FMMC), whose solution is a symmetric Markov chain that mixes to the uniform distribution faster than a simple random walk.

$$\begin{aligned} \underset{P}{\text{minimize}} \quad & \mu(P) \\ \text{subject to} \quad & P \geq 0, \quad P\mathbf{1} = \mathbf{1}, \quad P = P^{\mathsf{T}}, \\ & P[i,j] = 0, \ (i,j) \notin E(G) \end{aligned} \tag{2.12}$$

Observe that the constraint $P = P^{\mathsf{T}}$ means that the Markov chain is symmetric and thus will necessarily mix to a uniform distribution over the vertices of the underlying graph. The problem (2.12) can be cast as a semidefinite program and so is solvable efficiently.

2.3 Lifted Markov chains

Lifted Markov chains are a technique for speeding up the mixing time of certain Markov chains. The basic idea is to add additional states to each graph that act as a local memory. One then runs this lifted chain for a specified number of steps, then collapses back to the original graph. Sampling vertices via the above procedure can allow one to draw samples from particular distributions faster than from a normal Markov chain.

Lifting was first introduced by Diaconis, Holmes and Neal [DHN00] for the particular example of the cycle graph. Chen, Lovász and Pak [CLP99] formalised lifting for any graph and provided bounds on their mixing. Apers, Sarlette and Ticozzi [ATS17] broadened the scope of lifted Markov chains by showing that under

different design scenarios, different mixing bounds apply to liftings. Liftings have been applied to sampling from Ising models on complete and path graphs [Vuc16], and considered in the context of continuous Markov chains by Ramanan and Smith [RS16], with associated mixing bounds given. Lifted Markov chains and have even been extended by Jung and colleagues [JSS10] to a construction called a *pseudo-lifting*, and applied to consensus algorithms. The lifting technique has even been applied to gradient descent optimisation of a certain restricted class of objective functions defined on graphs [FB17].

2.3.1 Lifting the cycle graph

We shall first consider the first lifted Markov chain as an example, on the cycle graph. Here we take the n-cycle, C_n, to be the graph with vertex set $V(C_n) = \{0, \dots, n-1\}$ and arc set $\{(i, i \pm 1 \bmod n) \,|\, i \in \{0, \dots, n-1\}\}$. Consider the Markov chain M_{C_n} on C_n, that has an arbitrary starting state in $V(C_n)$ and transition probabilities of $1/2$ on each arc. It is well known that the mixing time of this Markov chain is quadratic in n for odd n, i.e. $\mathcal{M}_\epsilon = \Theta(n^2 \log(1/\epsilon))$ and is undefined for even n. For the cycle these transition probabilities are optimal for mixing to the uniform distribution over all vertices.

We can consider a lift of this chain first considered by Diaconis, Holmes and Neal, the *Diaconis lift* [DHN00]. We augment each vertex $i \in \{0, \dots, n-1\}$ with the pair of vertices $(\pm 1, i)$ and define the *lifted cycle* C_n^c like so:

$$\begin{aligned} V(C_n^c) &= \{(s,k) \,|\, k \in \{0, \dots, n-1\},\ s \in \pm 1\} \quad \text{and} \\ E(C_n^c) &= \{((s', k \pm 1 \bmod n), (s,k)) \,|\, k \in \{0, \dots, n-1\},\ s, s' \in \pm 1\}. \end{aligned} \tag{2.13}$$

The transition probabilities of the chain are as follows:

$$P^c[i,j] = \begin{cases} 1 - 1/n, & i = (s,k),\ j = (s, k + s \bmod n); \\ 1/n, & i = (s,k),\ j = (-s, k + s \bmod n); \\ 0, & \text{otherwise}, \end{cases} \tag{2.14}$$

where $s \in \pm 1,\ k \in \{0, \dots, n-1\}$. Figure 2.1 shows the allowed transitions and associ-

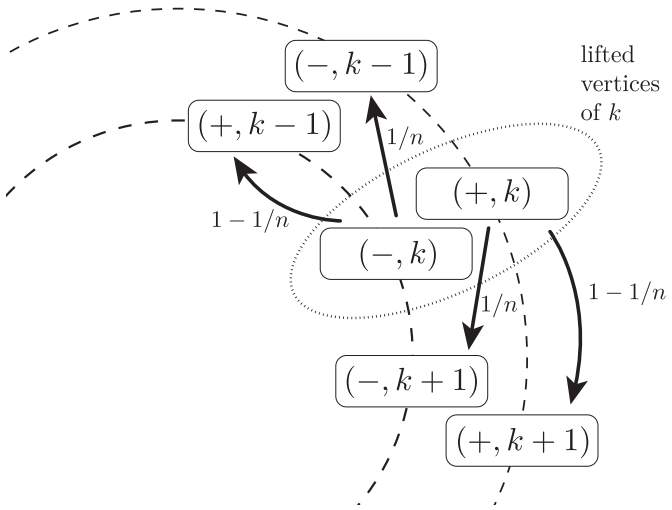

Figure 2.1: Illustration of the lifted Markov chain on C_n^c. Each vertex k is lifted to the pair of vertices $(-,k)$, $(+,k)$. The arrows indicate the outgoing transition probabilities from the lifted vertices of k. Observe that there is high probability to maintain the 'sign' of the vertices as the walk progresses.

ated probabilities. This chain has been shown to have mixing time to the marginal[3] $\Theta(n)$ (for fixed ϵ), displaying a quadratic speedup over the non-lifted chain [DHN00]. This choice of transition probabilities imposes some kind of 'inertia' on the walk, in that if the walker takes a step (anti-)clockwise around the cycle, it is far more likely to take the next step (anti-)clockwise around the cycle. This inertia allows the walker to traverse the graph faster, and therefore mix faster.

2.3.2 Lifting design scenarios and mixing

The more contemporary view of lifted Markov chains initiated by Apers, Sarlette and Ticozzi [ATS17] contextualises lifted Markov chains under a variety of design scenarios. These scenarios are most efficiently presented in tabular form, which we give in Table 2.1. The scenarios require certain constraints be imposed on the lifted Markov chain with respect to the collapsed chain. Bounds on the attainable mixing times for the various design scenarios are shown in Table 2.2.

It should be noted that the paper by Chen, Lovász and Pak [CLP99] takes place in the (slmre) context, that is, the bounds on mixing to the marginal are proven for the case where the lifted chain is irreducible, marginally mixes from any starting distribution, itself mixes and has matching ergodic flows.

[3]By mixing time to the marginal, we mean the time for the induced distribution on the collapsed chain to have converged. A more rigorous definition will be given in Chapter 3.

Scenario label	Description
(s)	Lifted chain must marginally mix from any starting distribution.
(S)	Can choose the (linear) initialisation map for a lifting.
(i)	Invariance of target marginal, that is, once the marginal distribution has reached the target, the distribution stays the same.
(I)	Collapsed chain can reach the target distribution, but then deviate from this distribution.
(m)	Both lifted and collapsed chains mix.
(M)	Mixing to the marginal only.
(r)	Lifted Markov chain must be irreducible.
(R)	Lifted Markov chain is reducible.
(e)	Ergodic flows of the lifted chain must match the collapsed chain. The ergodic flow $Q[x,y]$ for $x,y \in V(G)$ is defined as $Q[x,y] := \pi[x]P[x,y]$, where π is the stationary distribution.
(e_δ)	Ergodic flows match to within accuracy δ.
(E)	No constraints on ergodic flows of lifted chain.

Table 2.1: Table of lifting design scenarios investigated by Apers, Sarlette and Ticozzi in [ATS17]. A five letter string, one from each section of the table, denotes a scenario. For example, the conductance bounds proved by Chen, Lovász and Pak [CLP99] apply to the scenario slmre. A capital letter corresponds to a less restrictive design scenario. Omitting letters in the scenario string implies the union of the upper and lower case scenarios, i.e. the scenario (SIre) $\equiv$ (SImre) $\cup$ (SIMre).

Scenarios	Bounds on mixing time
(si)	no advantage
(s)	$\geq 1/(4\Phi)$
(se)	$\geq 1/(4\Phi(P))$
(i)	$\geq 1/(8\Phi)$
(ie)	$\geq 1/(8\Phi)$
(SI) $\setminus$ (SIre)	$\leq D(G)+1$
(SIre_δ)	$\leq D(G)+1$

Table 2.2: Bounds on the mixing time for different lifting scenarios proven by [ATS17]. Note that Φ and $\Phi(P)$ are distinct, standing for the graph conductance and Markov chain conductance respectively. The parameter $D(G)$ is the graph diameter. Scenarios are given as in Table 2.1.

2.4 Quantum walks

Quantum walks are the quantum analogue of Markov chains. The basic idea is to replace the iteration of a stochastic matrix on a probability vector with the iteration of a unitary matrix on a unit vector in complex Euclidean space. In doing so, the physical model changes from a walker moving stochastically between the vertices on a graph to a "quantum" walker moving in superposition over the vertices. Indeed, the quantum walk phenomenon has even been used to describe certain biological systems [MRLAG08, ECR$^+$07].

The term *quantum random walk* was first coined by Aharonov, Davidovich, and Zagury [ADZ93], and was initially conceived with a view to quantum optics applications. Coined quantum walks began with Meyer [Mey96a, Mey96b], who investigated them in the context of quantum cellular automata. He showed that a one-dimensional quantum cellular automaton which is translationally invariant and acts only locally[4] displays only trivial behaviour in its time evolution. He also showed that adding an internal spin degree of freedom could be included to make the evolution nontrivial, and the resulting dynamics could be interpreted a discretisation of the Dirac equation. This cellular automata picture encouraged research into quantum walks on a line. Aharonov et al. [AAKV01] framed this evolution as a quantum walk, renaming the internal state a coin, in reference to a coin flip in classical random walks. The quantum walk is then defined by a coin flip followed by a shift or hop to adjacent vertices. The paper by Aharonov et al. [AAKV01] constituted the first more algorithmic focus on quantum walks; from this work the field of quantum walks grew into its modern form.

There are many different quantum walks models that have been proposed, analysed and used for applications. We shall describe a number of the most relevant models for our purposes. A more comprehensive survey can be found at [VA12] and a more combinatorial view on various quantum walk models can be found at [GZ17].

2.4.1 Zoology of quantum walks models

Quantum walks broadly fall into two classes, discrete and continuous-time. In this thesis we will focus mainly on discrete-time walks as they are more amenable to sim-

[4]The properties of translational invariance and locality are natural assumptions for a reasonable model of a cellular automaton.

ulation on a gate-model quantum computer, although one can use generic Hamiltonian simulation techniques to simulate a continuous walk. For some time it was not known whether there was any separation in terms of computational capabilities between continuous and discrete walks. This question was settled by Ambainis, Kempe and Rivosh [AKR04], who showed that a set of N items arranged on a grid can be searched on an $N \times N$ grid in time $O(\sqrt{N}\log N)$. Childs and Goldstone [CG04] established earlier that a continuous time walk requires time $\Omega(N)$ to carry out the same task.

We note at this stage that we will not consider *open quantum walks* [SP19] in this thesis, which concern quantum walks subject to noise, of which discrete-time and continuous-time models have been studied. In what follows, we constrain ourselves to unitary quantum walks.

2.4.2 Continuous Quantum Walks

We begin by defining a complex Euclidean space, $\mathcal{H}_{V(G)}$, spanned by basis states $|v\rangle : v \in V(G)$ corresponding to the vertices of some graph, G. Concretely,

$$\mathcal{H}_{V(G)} := \operatorname{span}\left(|v\rangle \,|\, v \in V(G)\right).$$

We also have some initial state $|\Psi_0\rangle \in \mathcal{H}_V$. A *Continuous-Time Quantum Walk* [FG98] over time t is given by $\exp(\mathrm{i}H(G)t)\,|\Psi_0\rangle$, where $H(G)$ is given elementwise by

$$H(G)[i,j] = \begin{cases} -\gamma, & v_i \neq v_j \text{ and } \{v_i, v_j\} \in E; \\ 0, & v_i \neq v_j \text{ and } \{v_i, v_j\} \notin E; \\ d_i\gamma, & v_i = v_j; \end{cases} \tag{2.15}$$

with γ being the transition probability per unit time. A graph G admits *perfect state transfer* [Bos03] between vertices v and u at time τ if there exists some $\alpha \in \mathbb{C}, |\alpha| = 1$ such that $\exp(\mathrm{i}H(G)\tau)\,|v\rangle = \alpha\,|u\rangle$. This phenomenon of perfect state transfer does not occur in non-trivial classical Markov chains[5] and constitutes an application of quantum interference. In [CFG01], a graph is given in which an exponential

[5] We can see this since submultiplicativity of the ℓ_1-norm implies that the distribution of an ergodic Markov chain monotonically converges to the stationary distribution. By Perron-Frobenius theory (see main theorem and subsequent remark of [Mac00]) this distribution always has all positive elements and so cannot be a Euclidean basis vector.

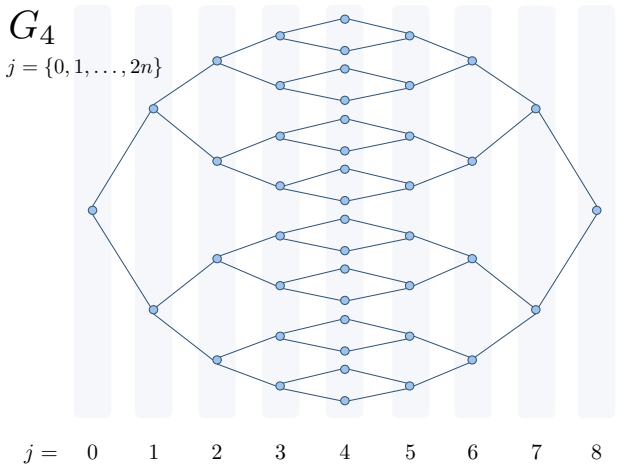

Figure 2.2: The graph G_4: Two binary trees of depth $n = 4$ glued together at their leaves, with column indices, $j = \{0, 1, \ldots, 2n\}$.

separation is observed between classical and quantum random walks, in the time to traverse a particular graph. This graph, G_n, consists of two binary trees of depth n glued together at their leaves. An example, G_4, is shown in Fig. 2.2. To be more precise, the probability of travelling from the root of the first tree to the root of the second in time polynomial in n is shown to be exponentially faster in the quantum case. To see this, we group the vertices of G_n into columns indexed by $j \in \{0, 1, \ldots, 2n\}$. Column 0 has just the left root, column 1 contains the adjacent two vertices and so on. Note that column n contains the 2^n vertices in the middle and column $2n$ has just the rightmost root. Analysing the classical case requires us only to keep track of the probabilities of being in a given column. When in the left tree ($0 \leq j \leq n$), the probability of moving to the right (column $j \to j+1$) is twice the probability of moving to the left (column $j \to j-1$). This means that the walk quickly moves into the middle of the graph. However, in the right hand tree the situation is reversed, whereby moving to the right has half the probability of moving to the left, thus making the time to destination exponential in n. More precisely, reaching column $2n$ from column 0 in n steps has probability less than 2^{-n}, meaning that traversing the graph in a time polynomial in n has a probability exponentially small in n.

In the quantum case, the symmetries of the Hamiltonian keep the state's evolution restricted to the $2n+1$-dimensional subspace spanned by the states $|\text{col } j\rangle$,

the uniform superposition over all vertices in column j, that is,

$$|\text{col } j\rangle = \frac{1}{\sqrt{N_j}} \sum_{v \in \text{column } j} |v\rangle, \quad \text{where} \quad N_j = \begin{cases} 2^j, & 0 \leq j \leq n; \\ 2^{2n-j}, & n \leq j \leq 2n. \end{cases} \tag{2.16}$$

This evolution is then equivalent to a quantum walk on the path graph, for which the transition time from root to root is polynomial in n.

2.4.3 Discrete-time quantum walks (DTQW)

We recall the fundamental idea of a quantum walk. The basic idea is to associate a basis vector of a particular complex Euclidean space to a vertex in a digraph and then repeatedly enact some unitary dynamics on the complex Euclidean space that respects the structure of the digraph. In the continuous time case, as seen in Section 2.4.2, it is straightforward to devise a way of doing this for an arbitrary graph. When we move from continuous to discrete-time we encounter a fundamental issue. There are some digraphs that do not support discrete unitary dynamics when we assign a basis vector to each vertex [Sev05], so for quantum walks to be applicable to any digraph we must introduce an auxiliary space.

We are thus forced to consider discrete-time dynamics on an extended state space (relative to $\mathcal{H}_{V(G)}$ defined in Section 2.4.2). This was first considered by Aharonov et. al. [AAKV01] in the coined model, and later in Szegedy's model [Sze04].

2.4.3.1 Coined walks

The intuition for a coined walk is as follows: the quantum walker at a particular vertex flips a unitary 'coin' that determines which edge to travel down, then proceeds in superposition along the edges to the corresponding vertices. This process repeats until the end of the walk. We now describe this formalism in more detail. The following construction can be generalised to non-regular digraphs (see for instance, the construction in [Ken06]), but for simplicity we restrict to the d-regular case.

Suppose we have a d-regular digraph G. Define a complex Euclidean space associated to the vertices of G, $\mathcal{H}_{V(G)} = \text{span}\,(|v\rangle \,|\, v \in V(G))$. Also define a complex Euclidean space associated to the coin $\mathcal{H}_C = \text{span}\,(|k\rangle \,|\, k \in [d])$. Our quantum walk acts on the complex Euclidean space $\mathcal{H}_C \otimes \mathcal{H}_{V(G)}$.

We need two unitary operators to define a coined quantum walk, the *coin operator* and the *shift operator*. We introduce the coin first: the coin $C \in \mathrm{U}(\mathcal{H}_C)$ is a unitary operator on $\mathcal{H}_C$. A common coin operator is the *Hadamard coin*, H_d, given by

$$H_d = \frac{1}{\sqrt{d}} \sum_{j\in[d]} \sum_{k\in[d]} \omega^{(j-1)(k-1)} |j\rangle\langle k|, \tag{2.17}$$

where $\omega := \mathrm{e}^{\frac{2\pi \mathrm{i}}{d}}$. We call a coined quantum walk utilising the Hadamard coin a *Hadamard walk*.

We need one more piece to define a coined quantum walk, the shift operator S, for which we use the description of Godsil [GZ17]. First, for each vertex u we must specify a linear order on its neighbours

$$f_u : \{1, 2, \ldots, d\} \to \{v : (u, v) \in E(G)\}. \tag{2.18}$$

The vertex $f_u(j)$ will be referred to as the j^{th} neighbour of u and the arc $(u, f_u(j))$ the j^{th} arc of u. For each vertex u, the shift operator S maps its j^{th} arc to the j^{th} arc of $f_u(j)$, i.e. $S(|j\rangle \otimes |u\rangle) = |j\rangle \otimes |f_u(j)\rangle$.

We can now construct one step of a coined quantum walk, described by the unitary operator $U = S \cdot (C \otimes I_{\mathcal{H}_{V(G)}})$. An initial state of the walk is some unit vector $|\psi(0)\rangle \in \mathcal{H}_C \otimes \mathcal{H}_{V(G)}$, typically a basis state $|k, v\rangle$ for some $k \in [d]$, $v \in V(G)$, where we abbreviate $|k\rangle \otimes |v\rangle$ as $|k, v\rangle$. The state after t timesteps is $|\psi_t\rangle = U^t |\psi(0)\rangle$. Thus we can totally characterise a quantum walk by the tuple $(U, |\psi(0)\rangle)$.

Following Aharonov *et. al.* [AAKV01][6], we denote by $Q_t(v|\psi(0))$ the probability of measuring the vertex v at time t of the quantum walk, contingent on the initial state being $|\psi(0)\rangle$. More concretely,

$$Q_t(v|\psi(0)) = \sum_{k\in[d]} \left| \langle k, v| U^t |\psi(0)\rangle \right|^2. \tag{2.19}$$

We denote by $Q_t(\cdot|\psi(0))$ the induced probability distribution over the vertices.

In fact, we can define a *general quantum walk* also, as in [AAKV01]. In this case we relax the requirement of the exact form that U can take, merely that U must respect the structure of the digraph. More precisely, for any k, v, the vector

[6] They use $P_t(v|\psi(0))$, which we change to avoid notational clashes.

$U|k,v\rangle$ only has support on basis states $|k',v'\rangle$ with $v' \in N(v) \cup \{v\}$, where $N(v)$ is the neighbourhood of v.

2.4.3.2 Szegedy walks

Szegedy's model of quantum walks differs in approach from coined walks in a number of ways, but nevertheless comes under the umbrella of a general walk, introduced at the end of Section 2.4.3.1. The Szegedy walk has been embraced by the quantum algorithms community as a subroutine underlying many algorithms, to be discussed in Section 2.4.3.3. The Szegedy walk is based on the *bipartite walk* model.

The bipartite walk model is somewhat different to the coined model, and has following advantages:

- It directly quantises a classical Markov chain.
- This scheme naturally encompasses non-regular graphs.
- The eigenvalues and eigenvectors of the walk operator are fully characterised in relation to the singular value decomposition of P.

First we introduce a classical bipartite walk. Every non-bipartite walk can be made bipartite by 'duplicating', which is equivalent to a walk on the bipartite double cover of the original graph (see Figure 2.3). Let X and Y be two finite sets and P, Q be matrices describing probabilistic maps from X to Y and Y to X respectively. Concretely[7], $P \in \mathrm{Stoch}(\mathbb{R}^{|Y|}, \mathbb{R}^{|X|})$ and $Q \in \mathrm{Stoch}(\mathbb{R}^{|X|}, \mathbb{R}^{|Y|})$. Since P and Q are stochastic, we have $\sum_{y \in Y} P[x,y] = 1$ for every $x \in X$ and $\sum_{x \in X} Q[y,x] = 1$ for every $y \in Y$, and all $P[x,y]$, $Q[y,x]$ are nonnegative. If we have a single $P : X \to X$ then we can build a bipartite walk by imposing that $P[x,y] = Q[y,x]$ for every $x, y \in X$, i.e. by setting $Q = P^{\mathsf{T}}$. Indeed this is how Szegedy proceeds in quantising a classical transition matrix P.

We quantise the walk (P,Q) by defining the two operators on the complex Euclidean space

$$\mathcal{H} = \mathrm{span}\{|x\rangle|y\rangle : x \in X,\ y \in Y\}.$$

Define the states

$$|\phi_x\rangle = \sum_{y \in Y} \sqrt{P[x,y]}|x\rangle|y\rangle$$

[7] Recall that probability distributions are represented as row vectors and so linear operators act from the right.

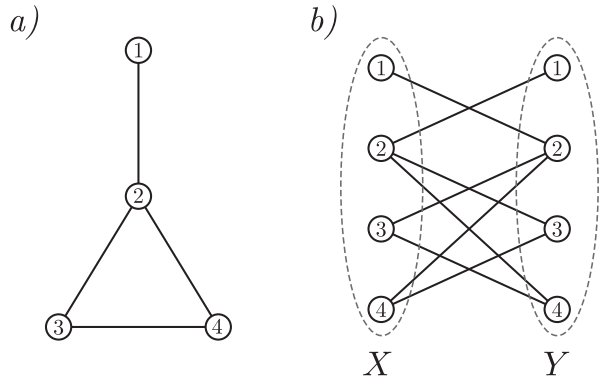

Figure 2.3: *a)* The paw graph, and *b)* its bipartite double cover. The bipartite double cover of a graph G is the graph $G \otimes K_2$, where $\otimes$ is the graph direct product.

for each $x \in X$ and

$$|\psi_y\rangle = \sum_{x \in X} \sqrt{Q[y,x]}|x\rangle|y\rangle$$

for each $y \in Y$.

Let $A = [\phi_x]_{x \in X}$ be the matrix composed of column vectors $|\phi_x\rangle$ and $B = [\psi_y]_{y \in Y}$ be the matrix composed of column vectors $|\psi_y\rangle$. Our walk operator, W, will be the product of

$$\mathrm{ref}_1 = 2AA^\dagger - I, \quad \mathrm{ref}_2 = 2BB^\dagger - I.$$

So we have

$$W = \mathrm{ref}_1 \mathrm{ref}_2.$$

Let $\mathcal{C}(A)$ be the column space[8] of A and let $\mathcal{C}(B)$ be the column space of B. Observe that $A^\dagger A = I_X$, therefore $(2AA^\dagger - I)A = A$. Also, for any $|\phi\rangle \in \mathcal{C}(A)^\perp$ we have $(2AA^\dagger - I)|\phi\rangle = -|\phi\rangle$. Recall that for a subspace V, $V^\perp$ is the subspace orthogonal to V. Thus, ref_1 and ref_2 are reflections about $\mathcal{C}(A)$ and $\mathcal{C}(B)$ respectively. The matrix $W := W_{PQ}$ is the *quantisation* of a bipartite walk (P,Q).

Szegedy [Sze04] used his construction to strengthen certain quantum search algorithm results. Namely

- For every symmetric, ergodic Markov chain with transition matrix P the quantum hitting time (appropriately defined) is at most a square root of the classical one. This holds for an arbitrary target set $M \subseteq X$. This definition of hitting means a Szegedy walk can detect if a subset of vertices contains a marked

[8] The column space of a matrix is the span of its columns.

element quadratically faster than a classical algorithm.

- For every symmetric, ergodic Markov chain with transition matrix that is also *state-transitive* (e.g. it comes from a vertex-transitive graph[9]) and for a single marked element z, when running the quantised version of P as above the element z is observed with probability at least $|X|/h$, where h is the average hitting time of the classical version of the chain. This corresponds to a speedup in returning a marked vertex.

2.4.3.3 DTQW-based algorithms

Both the coined model and Szegedy's model lend themselves to new algorithms based on searching. Exhaustively listing all of these algorithms lies outside the scope of this thesis; for clarity we briefly list several notable examples.

In the coined model:

- Element distinctness: Let $M, N \in \mathbb{N}$ such that $N < M$. Given $x_1, \ldots, x_N \in [M]$, does there exist $i, j \in [N]$, $i \neq j$ such that $x_i = x_j$? Ambainis provides an algorithm in [Amb04] solving element distinctness in $O(N^{2/3})$ time, (compared to $O(N)$ queries classically) by using a quantum walk on a Johnson graph[10].

- Searching on a grid: Find a marked element on a $\sqrt{N} \times \sqrt{N}$ grid. The authors of [AKR04] use coined walks to solve this problem in $O(\sqrt{N} \log N)$ time, as compared with $O(N)$ time for a classical algorithm and using naïve quantum search.

Szegedy's model, due to its favourable spectral properties has seen more algorithmic applications, with extensions to overcome its limitations. Magniez, Nayak, Roland, and Santha extended this model [MNRS11] so that the range of Markov chains to which Szegedy's search results apply grew from state-transitive chains to symmetric, ergodic chains for the searching problem. Later Krovi, Magniez, Ozols and Roland further developed this model, giving stronger upper bounds on the algorithm runtime [KMOR10]. Ambainis, Gilyén, Jeffery and Kokainis extend their method to work for multiple marked vertices [AGJK19]. Recent algorithms

[9] A graph is vertex-transitive if its automorphism group acts transitively upon its vertices.

[10] Johnson graphs are a family of undirected graphs defined from systems of sets. The vertices of the Johnson graph $J(n, k)$ are comprised of the k-element subsets of an n-element set; two vertices are adjacent if and only if the intersection of the two vertices (subsets) contains $(k-1)$ elements.

of a different flavour include walks-based Hamiltonian simulation [BN16] and fast-forwarding a reversible Markov chain [AS18], that is, producing a quantum state encoding the evolution of a Markov chain in time quadratically smaller than the time to run the chain.

2.4.3.4 Mixing of Quantum Walks

Much of the discussion of the preceding sections concentrates on hitting times in quantum walks; we shall now discuss mixing, which is more relevant for this thesis. Let us consider a quantum walk on the m-regular digraph G. For a quantum walk, unitarity prevents the state itself from converging, by the following argument: the ℓ_2-norm distance between consecutive states in a quantum walk is constant, as the walk operator is unitary. Thus the limit $\lim_{t\to\infty} U^t |\psi(0)\rangle$ does not exist in general, as for convergence we demand that the distance monotonically decreases with an increasing number of timesteps. Perhaps more naturally, we can consider the convergence of the induced probability distribution over the nodes[11], $Q_t(\cdot|\psi(0))$. We can see that this distribution does not converge either, using the following argument from [AAKV01]. As the quantum walk operator U is unitary, it has eigenvalues of the form $\mathrm{e}^{\mathrm{i}\theta}$. For any $\epsilon > 0$, there exists some finite t for which $\left|1-\mathrm{e}^{\mathrm{i}\theta t}\right| \leq \epsilon$ for all eigenvalues θ. Thus, $U^t|\psi(0)\rangle$ can be made arbitrarily close in ℓ_2-distance to $|\psi(0)\rangle$ for infinitely many times t. Unless $U|\psi(0)\rangle = |\psi(0)\rangle$, the walk is *periodic* and $Q_t(\cdot|\psi(0))$ does not converge.

However, we can talk about the well-defined notion of *average mixing*. Consider the Cesàro average of $Q_t(v|\psi(0))$ over the first T timesteps,

$$\overline{Q}_T(v|\psi(0)) = \frac{1}{T}\sum_{t=0}^{T-1} Q_t(v|\psi(0)).$$

The limit $\lim_{T\to\infty} \overline{Q}_T(v|\psi(0))$ exists for any v and $|\psi(0)\rangle$, see Proposition A.1. We denote by $\pi^q_{\psi(0)}$ the distribution $\lim_{T\to\infty} \overline{Q}_T(\cdot|\psi(0))$, in analogy with classical case and refer to it as the *quantum average mixing distribution* of the quantum walk. One can easily sample from the distribution $\overline{Q}_T(\cdot|\psi(0))$ using the following procedure. Choose a time $t \in \{0,\ldots,T-1\}$ uniformly at random, run the quantum walk for t timesteps, then measure which node the walker is at. The vertex will be distributed

[11] As defined in Section 2.4.3.1

according to $\overline{Q}_T(\cdot\,|\psi(0))$.

We are now in a position to define the *ϵ-quantum average mixing time*,

$$\mathcal{M}^q_\epsilon = \max_{|\psi(0)\rangle\in\Psi} \min_{T\in\mathbb{Z}_+} \left\{ T \,\middle|\, \forall t \geq T,\ \left\| \overline{Q}_T(\cdot\,|\psi(0)) - \pi^q_{\psi(0)} \right\|_{TV} \leq \epsilon \right\}, \tag{2.20}$$

where Ψ is the allowed set of starting states, typically the basis vectors $\{|k,v\rangle \,|\, k \in [m],\ v \in V(G)\}$.

Examples of quantum mixing. The mixing of a coined quantum walk on the n-cycle, C_n, was rigorously analysed in [AAKV01]. More concretely they perform the Hadamard walk on a complex Euclidean space isomorphic to $\mathbb{C}^2 \otimes \mathbb{C}^n$. The basis states for the coin space are $\{|L\rangle, |R\rangle\}$, standing for 'left' and 'right'. The coin operator is $C = H_2 = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$ and the shift operator S acts as

$$\begin{aligned} S\,|L,i\rangle &= |L, i-1 \bmod n\rangle\,; \\ S\,|R,i\rangle &= |R, i+1 \bmod n\rangle\,. \end{aligned} \tag{2.21}$$

It is shown for this walk that the mixing time $\mathcal{M}^q_\epsilon = O(n\log(n)\frac{1}{\epsilon^3})$, demonstrating quadratic speedup in mixing as compared with the classical walk on the cycle (for fixed precision). Interestingly, this speedup is seen in the lifted Markov chain also. We also note that the inverse polynomial dependence on ϵ can be made inverse polylogarithmic using an amplification scheme detailed in [AAKV01].

We briefly list further progress on analysing mixing for particular graphs.

- Ahmadi, Belk, Tamon and Wendler consider mixing of continuous walks on circulant graph [ABTW03].
- Fedichkin, Solenov and Tamon study mixing of continuous-time walks on the cycle [FST06].
- Marquezino, Portugal and Donangelo investigate mixing of discrete-time walks on the hypercube [MPAD08].
- Marquezino, Portugal and Abal study mixing on a 2D lattice embedded on a torus [MPA10].
- Kieferová and Nagaj examine mixing on necklace graphs [KN12].

- Chakraborty, Luh and Roland consider mixing of continuous walks on Erdős-Rényi random graphs [CLR19].

Upper and lower bounds. The bounds on mixing time that have been derived in the literature usually only hold for particular family of graphs. There has also been research into general-purpose (but necessarily looser) bounds.

Aharonov and colleagues [AAKV01] prove a general lower bound on the quantum mixing time $\mathcal{M}_\epsilon^q = \Omega(1/\Phi)$, where Φ is the graph conductance, for graphs of bounded degree. They also provide an upper bound. Godsil and Zhan [GZ17] provide a more general bound than this upper bound. For completeness, we provide the proof of the following proposition in Appendix A, Corollary A.1.

Proposition 2.1 (Godsil, Zhan [GZ17, Corollary 9.1]) *Suppose the quantum walk matrix U has spectral decomposition $U = \sum_r \mathrm{e}^{\mathrm{i}\theta_r} F_r$, where F_r are the spectral idempotents and $\mathrm{e}^{\mathrm{i}\theta_r}$ the eigenvalues. Then, the quantum average mixing time satisfies*

$$\mathcal{M}_\epsilon^q \leq \frac{2|E|}{\epsilon} \sum_{r \neq s} \frac{1}{|\mathrm{e}^{\mathrm{i}\theta_r} - \mathrm{e}^{\mathrm{i}\theta_s}|}$$

2.4.3.5 Implementation on a Quantum Computer

Since a quantum walk consists of a unitary operator U repeatedly applied to an initial state $|\psi(0)\rangle$, it is readily amenable to simulation on a quantum computer (recall the discussion on implementing arbitrary unitary operators in Section 2.1.2). Indeed, assigning a computational basis state to each basis state of the walk space $\mathcal{H}$ allows one to simulate a quantum walk on an exponentially large graph. For an efficient simulation, one needs to efficiently prepare the initial state $|\psi(0)\rangle$. Usually the initial state of interest corresponds to starting on a particular vertex; the typical encoding of quantum walk states would make this a computational basis state and so easy to prepare.

2.4.3.6 Universality of quantum walks

Going in the other direction, quantum walks have been shown to be universal for quantum computation in the discrete [LCE+10] and continuous-time [Chi09] regimes. More precisely, suppose we are given an input quantum state $|\psi_{\text{in}}\rangle$ and a circuit $\mathcal{C}$ of depth L, such that the computation is in state $|\psi_{\text{out}}\rangle$ having applied $\mathcal{C}$. One can construct a graph and associated quantum walk such that the state of the

quantum walk mimics that of the computation, namely, the initial state of the walk is (appropriately encoded) $|\psi_{\text{in}}\rangle$ and the output state is $|\psi_{\text{out}}\rangle$, and the walk only requires a number of steps that is a modestly growing function of L.

We briefly describe how this is done in the discrete time case, with the continuous time case being similar. This universality was originally proved in the continuous-time case by Childs [Chi09], with a similar construction given for coined quantum walks by Lovett et al [LCE+10], which we describe below.

The general idea is to first represent the given quantum circuit $\mathcal{C}$ using a particular universal set of gates, namely $\{T, \text{HADAMARD}, \text{CNOT}\}$ in a manner similar to the discussion in Section 2.1.2. These gates are represented by the matrices

$$T = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\pi/4} \end{bmatrix}, \qquad \text{HADAMARD} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

An illustration of this construction is given in Figure 2.4. This circuit representation is then "expanded" into a representation in which each computational basis state is represented by a wire, and so circuit elements in the usual quantum circuit representation are expanded in this picture. Specific sets of vertices and edges for each circuit element are used. For each degree-d vertex, a d-dimensional *Grover coin* $G^{(d)}$ is used as the coin, where

$$G^{(d)} := \begin{bmatrix} \frac{2}{d} & \cdots & \frac{2}{d} \\ \vdots & \ddots & \vdots \\ \frac{2}{d} & \cdots & \frac{2}{d} \end{bmatrix} - I_d.$$

The combination of the four-dimensional Grover coin with a shift enacts perfect state transfer, allowing one to draw the "wires" in a given circuit. The other circuit elements are constructed using similar gadgets. A coined walk is carried out for a number of steps sufficient to "complete the computation". Measurement at the output arcs (appropriately normalised) then gives the same output statistics as the original computation to be simulated. This is made more concrete in Figure 2.4.

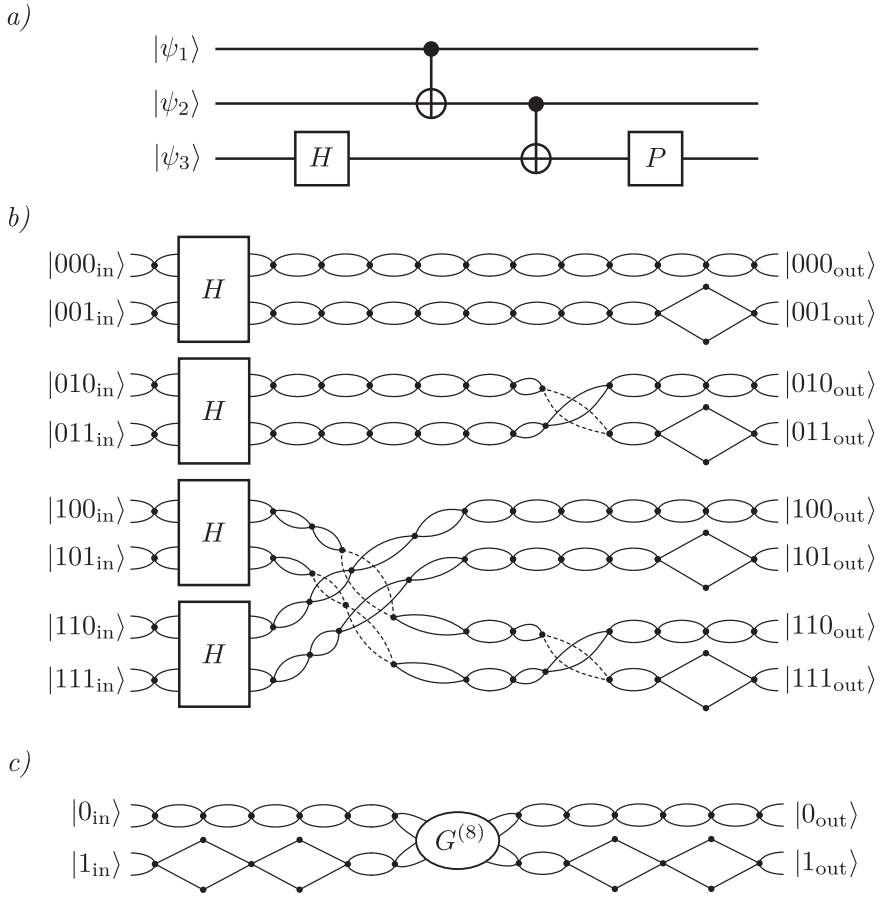

Figure 2.4: Illustration of the universality construction for coined quantum walks. In panel *(a)* we see a circuit to be emulated by the quantum walk. A HADAMARD operation is performed on qubit 3 followed by two CNOT gates. Finally, a PHASE gate is applied on qubit 3. In panel *(b)* we see the equivalent quantum walk graph. The edges are directed from left to right. For each basis state the input state amplitude is split equally between the equivalent input arcs. The boxes marked "H" represent the gadget for the HADAMARD, which is shown in panel *c)*. For a vertex of degree d the Grover coin $G^{(d)}$ is used. After running the quantum walk for the full length of the computation (i.e. enough steps to traverse the graph from left to right), measuring gives the same output statistics as the circuit output.

2.4.4 Quantum walks with memory

Recalling the discussion in Section 2.3, lifted Markov chain can in some sense be thought of as a random walk with memory, for instance, in the Diaconis lift of Figure 2.1, the current position of the walker encodes the location of the previous state at any given timestep with the current state. One might think of extending this principle to quantum walks. Indeed there is a small literature on this topic, with notable references [FAJ04, McG10]. The evolution of these walks on the path graph is studied and compared with classical random walks and coined quantum walks.

2.5 Combinatorial optimisation

Combinatorial Optimisation [KV12] is the search for an optimum object in a finite collection of objects. Combinatorial optimisation problems see wide and numerous applications in industry [PH02, PK03, KPRS09], as well as continuing substantial theoretical interest [GJ79, AB09].

To solve such problems mathematically we typically describe the finite collection in some concise representation (usually a graph) and the number of objects is exponential in the size of the representation (say, Hamiltonian cycles[12]).

Curiously, most interesting combinatorial optimisation problems are defined on digraphs, and usually fit into two classes: their decision version is in P, or is NP-complete. We give two examples:

Problem 2.1. MinimumSpanningTree

Instance: A graph G and edge weights $c : E(G) \to \mathbb{R}$.

Task: Find a spanning tree T in G whose weight $c(T) = \sum_{e \in E(T)} c(e)$ is minimum, or decide that G is not connected.

Problem 2.2. MaxCut

Instance: A graph G and (nonnegative) edge weights $c : E(G) \to \mathbb{R}_+$.

Task: Find a cut S in G with maximum total weight $c(S) := \sum_{e \in S} c(e)$.

Recall that a *spanning tree* is an acyclic subgraph containing every vertex of $V(G)$, and a *cut* $S \subseteq E(G)$ is a collection of edges that if removed from the graph,

[12] A Hamiltonian cycle is a cycle that visits every vertex in a given graph.

partition the vertices into two subgraphs, disconnected from one another. The first problem is solvable in polynomial time whereas the second problem is NP-hard.

There is also the notion of solving a combinatorial optimisation problem *approximately*. In this case one doesn't return an optimal solution, but a solution that has a value that is within a guaranteed distance to the optimal value. One can even solve a combinatorial optimisation *heuristically*, with no success guarantees, but empirically demonstrated success.

2.5.1 Semidefinite approximations

Many interesting combinatorial optimisation problems are NP-hard, and so there is no efficient exact solution. Nonetheless, efficient approximation algorithms exist, with some theoretical guarantee on closeness to the optimal value. One successful scheme for approximating solutions to combinatorial optimisation is semidefinite programming (SDP), wherein one forms an SDP whose optimal solution corresponds to a (not generally optimal) solution of the original problem. The SDP is known as the *SDP relaxation* of the original problem. SDPs are well known to be solvable in polynomial time using interior point methods [BV09]

The first result launching this scheme of research was presented by Goemans and Williamson [GW95]; they provided a polynomial-time solution to MaxCut with a cut that is guaranteed to have weight > 0.878 times the optimal solution. Indeed, there are SDP relaxations that can be applied to general combinatorial optimisation problems, such as the *Lasserre Hierarchy* [Las02], with trade-offs between accuracy and computational complexity. A good survey can be found at [Lau03].

A semidefinite program takes the following form. Let $X \in \mathrm{Pos}(\mathbb{R}^n)$ be a positive semidefinite matrix. To specify a linear function of X, $C : \mathrm{Pos}(\mathbb{R}^n) \to \mathbb{R}$, we can write

$$C(X) = C \bullet X := \sum_{i=1}^{n} \sum_{j=1}^{n} C[i,j]X[i,j] = \mathrm{Tr}\Big(C^{\mathsf{T}} X\Big) \tag{2.22}$$

The SDP then takes the form

$$\begin{aligned} \text{minimise} \quad & C \bullet X, \\ \text{subject to} \quad & A_i \bullet X = b[i],\ i \in [m], \\ & X \succeq 0, \end{aligned} \tag{2.23}$$

where C and the A_i are symmetric $n \times n$ matrices and $b[i]$ is the i^{th} component of the vector $b \in \mathbb{R}^m$. We are thus interested in minimising a linear function over the cone of positive semidefinite matrices, subject to affine constraints.

There is also the associated Langrangian *dual* program, given by

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{m} b[i]y[i] = b^{\mathsf{T}} y \\ \text{subject to} \quad & \sum_{i=1}^{m} A_i y[i] \preceq C \\ & y \in \mathbb{R}^m \end{aligned} \tag{2.24}$$

In general, *weak duality* holds, that is, $C \bullet X \geq b^{\mathsf{T}} y$ for all $X \succeq 0$, $y \in \mathbb{R}^m$. Under certain circumstances, equality holds for the optimal (X, y) and this is known as *strong duality*.

2.5.2 Markov chain approaches to Combinatorial Optimisation

Markov chains form the theoretical foundation for the metaheuristic approach to combinatorial optimisation [BR03, BLS13], which see the most widespread use in industrial applications.

As an illustration, we shall briefly show how to use the Metropolis algorithm described in Section 2.2.2 to solve combinatorial optimisation problems. We describe the method outlined in [LPW09, Example 3.2]. Let f be a real-valued function defined on the vertex set of a graph. The goal is to compute

$$f^\star := \max_{x \in \Omega} f(x).$$

Note that many combinatorial optimisation problems can be cast in to this form, where $|\Omega|$ is exponentially large. Consider for instance the problem of finding a Hamiltonian cycle in a graph G: one forms a new graph in which each vertex x is a permutation of the vertices in G, an edge is drawn between permutations x and y when they are the same up to a pairwise swap of two vertices. Defining

$$f(x) = \begin{cases} 1, & x \text{ is a Hamiltonian cycle in } G \\ 0, & \text{otherwise,} \end{cases}$$

and solving for $f^\star$ decides if there is a Hamiltonian cycle in G. Similar constructions can be devised for many other combinatorial optimisation problems.

Problems of this type can be solved using the Metropolis algorithm (see Section 2.2.2), in the following manner. Fix some $\lambda \geq 1$ and define

$$\pi_\lambda(x) = \frac{\lambda^{f(x)}}{Z(\lambda)},$$

where $Z(\lambda) = \sum_{x\in\Omega} \lambda^{f(x)}$ is the *partition function*. Then, run the Metropolis algorithm with π_λ as the target distribution. Note that we don't need to compute $Z(\lambda)$ which may well consist of exponentially many terms, due to the design of the Metropolis chain transition probabilities.

We then define the set of solutions $\Omega^\star := \{x \in \Omega \,|\, f(x) = f^\star\}$. Computing the limit

$$\lim_{\lambda\to\infty} \pi_\lambda(x) = \lim_{\lambda\to\infty} \frac{\lambda^{f(x)}/\lambda^{f^\star}}{|\Omega^\star| + \sum_{x\in\Omega\setminus\Omega^\star} \lambda^{f(x)}/\lambda^{f^\star}} = \frac{\mathbf{1}_{\{x\in\Omega^\star\}}}{|\Omega^\star|}.$$

Concretely, this means as $\lambda \to \infty$, the Metropolis chain will converge to a stationary distribution, uniformly distributed over the global maxima of f. Importantly, the speed of this convergence is not known in general (apart from a loose upper bound inverse in the spectral gap), and for the case of NP-hard problems, must be necessarily superpolynomial unless $\mathsf{P} = \mathsf{NP}$.

2.5.3 Quantum Approaches to Combinatorial Optimisation

There has been much research interest in solving combinatorial optimisation problems on a quantum computer, due to asymptotic improvements in algorithm runtime found for many computational problems (see Section 2.1.3). For completeness, we describe some of the approaches below.

2.5.3.1 Quantum Metropolis

There has been much research effort into producing a quantum algorithm with proven speedup over the Metropolis-Hastings algorithm for sampling from probability distributions. Since this line of research is not the focus of this thesis, I merely mention briefly an important result, linking quantum algorithms to MCMC (Section 2.2.2). See also [Mon15] for more general results in this area.

Typically, the focus is sampling from *thermal distributions*, that is, probabilities $p_i = \exp(-\beta E_i)/Z$, where β is a real-valued parameter known as the inverse

temperature, E_i is the energy of the i^{th} state and $Z := \sum_i \exp(-\beta E_i)$, the *partition function*. The end result of this line of research is the algorithm of Yung and Aspuru-Guzik [YAG12] for sampling from a thermal distribution. The algorithm has a runtime of $O\left(\frac{1}{\sqrt{1-\lambda_2}}\right)$, where $1-\lambda_2$ is the spectral gap of the associated transition matrix (in this case, the classical Metropolis chain, see Eq. 2.11).

2.5.3.2 Quantum SDP algorithms

Here we describe the recent algorithm by Brandão and Svore for solving SDPs [BS18]. The algorithm has worst-case running time

$$O\left(n^{\frac{1}{2}} m^{\frac{1}{2}} \text{poly}(\log(n), \log(m), R, 1/\delta)\right),$$

with n and s respectively the dimension and row-sparsity[13] of the input matrices, m the number of constraints, δ the accuracy of the solution and R an upper bound on the trace of the optimal solution. This gives an unconditional square-root speedup in both m and n over the best currently known classical methods, while incurring large polynomial overhead in R and δ.

The authors also show that the algorithm is nearly optimal (in n and m) by proving a quantum lower bound of $\Omega(\sqrt{n}+\sqrt{m})$ for solving SDPs with constant s, R and δ. This is in contrast to the classical lower bound of $\Omega(n+m)$.

For a different input model, which we shall go into later, the algorithm offers an *exponential* speedup over classical methods, with runtime $O\left(\text{poly}(\log(n), \log(m), r, R, 1/\delta) m^{\frac{1}{2}}\right)$, where r is an upper bound on the rank of any input matrix.

The algorithm in [BS18] solves SDPs of the form:

$$\begin{aligned} &\max\ \text{Tr}(CX), \\ &\forall j \in [m]\ :\ \text{Tr}(A_j X) \leq b[j], \\ &X \succeq 0, \end{aligned} \tag{2.25}$$

$$\begin{aligned} &\min\ b \cdot y, \\ &\sum_{j=1}^{m} A_j y[j] \succeq C, \\ &y \geq 0, \end{aligned} \tag{2.26}$$

where the matrices $\{A_1, \ldots, A_m, C\}$ are $n \times n$ and Hermitian and $b \in \mathbb{R}^m$. Eqs. 2.25 and 2.26 represent the *primal* and *dual* problems respectively.

[13] The row-sparsity is the nonzero elements in a row, maximised over all the rows.

Without loss of generality we can assume that

$$\|A_i\| \leq 1, \ \forall i \in [m], \quad \text{and} \quad \|C\| \leq 1, \tag{2.27}$$

with $\|\cdot\|$ being the operator norm, by normalising the elements of b and optimal solution appropriately.

Since the algorithm runtime depends intimately on the input and output models, we shall describe these in a little more detail. The matrices $\{A_1, \ldots, A_m, C\}$ and the vector $b \in \mathbb{R}^m$ are the input to the algorithm. For b, we assume there is an oracle $\mathcal{P}_B$ which acts as

$$|j, z\rangle \overset{\mathcal{P}_B}{\mapsto} |j, z \oplus b[j]\rangle\,, \tag{2.28}$$

where $j \in [m+1]$ and $b[j]$ is given as a truncated bit string representation.

For the $\{A_1, \ldots, A_m, C := A_{m+1}\}$, there are two input models.

Input Model 1: We have an oracle $\mathcal{P}_A$ that given the indices $j \in [m+1]$, $k \in [n]$ and $l \in [s]$ computes the bit string representation of the l^{th} non-zero element of the k^{th} row of A_j, i.e.

$$|j, k, l, z\rangle \overset{\mathcal{P}_A}{\mapsto} |j, k, l, z \oplus A_j[k, f_{jk}(l)\rangle], \tag{2.29}$$

with $f_{jk} : [r] \to [N]$ giving the l^{th} non-zero entry of row k in matrix A_j.

Input Model 2: Assume there is an oracle $\mathcal{P}_{A'}$ which prepares copies of the eigenstates of the input matrices and the corresponding eigenvalues. More concretely, for $i \in \{0, 1, \ldots, n\}$, let

$$A_i = \sum_{l=1}^{r_i} \kappa_l^i \left|\eta_l^i\right\rangle\!\left\langle\eta_l^i\right|, \tag{2.30}$$

be the spectral decomposition of A_i, with $r_i \leq r$. Then we assume $\mathcal{P}_{A'}$, given i, l as input, approximates (to accuracy $\nu > 0$) of a copy of quantum state $|\eta_l^i\rangle$ and the corresponding eigenvalue κ_l^i.

Output: We must note that writing down the full solution (either X or y) will eliminate any speedup as we need $\Theta(n)$ time for this task. Taking this into consideration, the algorithm's output provides:

- An estimate of the optimal objective value.
- An estimate of $\|y\|_1$ and/or $\mathrm{Tr}(X)$.
- *Samples* from the distribution $p := y/\|y\|_1$ and/or from the quantum state $\rho := X/\mathrm{Tr}(X)$.

Van Apeldoorn and colleagues have improved the runtime of this algorithm with significantly reduced polynomial dependence on R, r and δ [vGGdW17, vAG18], under similar input and output models. Recently, van Apeldoorn and Gilyén provided an application of these methods to computing Nash equilibria of two-player zero-sum games in sublinear time [vAG19].

There has yet to be a demonstrated unconditional asymptotic speedup of these methods for a combinatorial optimisation problem, which remains a tantalising open question.

2.5.3.3 Quantum Approximate Optimisation Algorithm (QAOA)

The Quantum Approximate Optimisation Algorithm (QAOA) [FGG14] of Farhi, Goldstone and Gutmann is a general method for solving combinatorial optimisation problems approximately. The approach goes roughly as follows: define unitaries $U(\beta)$ and $U(\gamma)$ parametrised on real values $\beta, \gamma \in [0, 2\pi]$, that will depend on the optimisation problem at hand. Then, for an appropriately initialised input state, $|s\rangle$ (a uniform superposition over all basis states), apply the evolution

$$U(\beta_p)U(\gamma_p)\cdots U(\beta_1)U(\gamma_1)\,|s\rangle := |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle\,, \tag{2.31}$$

where we define the length p vectors $\boldsymbol{\gamma} := [\gamma_1 \cdots \gamma_p]^\mathsf{T}$ and $\boldsymbol{\beta} := [\beta_1 \cdots \beta_p]^\mathsf{T}$. We then measure in the computational basis and evaluate the cost function operator C. The expected value of C is given by $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}|\, C\, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$. Defining M_p as the maximum F_p over all angles, the authors of [FGG14] show that

$$\lim_{p\to\infty} M_p = \max_z C(z) \tag{2.32}$$

where the z are basis states representing the objects being optimised over.

Finding the optimal $\boldsymbol{\gamma}, \boldsymbol{\beta}$ for a given problem efficiently is non-trivial. The authors demonstrate an efficient scheme for the case when p is fixed. Furthermore,

they analyse the performance of the algorithm for MaxCut on 2-regular and 3-regular graphs at fixed p.

In the work [HWO+17], the authors take the general scheme of QAOA and apply it to 8 specific problems, providing the operators $U(\beta)$ and $U(\gamma)$ for each problem.

The problem of choosing the parameters $\boldsymbol{\gamma}, \boldsymbol{\beta}$ efficiently in the general case remains open. This method lies within the class of heuristics, seen to be applicable for NISQ machines [OMA+17].

2.5.3.4 Exact Methods

Ambainis and colleagues [ABI+18] develop quantum techniques for dynamic programming, a classical technique that has long been used in the solution of combinatorial optimisation problems. They demonstrate speedups for the travelling salesman problem and minimum set cover problems.

Even more recently, Montanaro has derived a quantum algorithm with near quadratic improvement for branch-and-bound algorithms [Mon19], which are classically used to exactly solve many combinatorial optimisation problems.

2.5.3.5 Quantum Annealing for Combinatorial Optimisation Problems

While so far discussion has focused only on gate-model quantum algorithms (with which this PhD is concerned), one must not forget that for combinatorial optimisation problems there is a large community of researchers applying quantum annealing as a heuristic for solving these problems. Indeed this area was the first (and arguably at the time of writing remains the only) to represent quantum methods being practically used. The company D-Wave produces commercial quantum annealers such as its 2X machine [D2X].

Quantum annealing is the finite temperature analogue of adiabatic quantum computation [FGGS00], which works as follows: A quantum system is prepared in the *known* ground state of some Hamiltonian, H_B. Then, the Hamiltonian is interpolated between H_B and H_P, where the ground state of H_P encodes the solution to some problem of interest. Provided the interpolation is slow enough, the system remains in its ground state throughout the evolution and ends up in the solution ground state. The hope is that sampling the state at the end of this evolution

returns a state encoding a close to optimal solution. The papers [KXB+16, KYR+17] demonstrate D-Wave's recent progress in this area in detail.

2.5.4 The graph isomorphism problem

A combinatorial optimisation problem that provides motivation for a large part of this thesis is the graph isomorphism problem.

Problem 2.3. GraphIsomorphism

Instance: A graph G and a graph H.

Task: Find bijections $\Phi_V : V(G) \to V(H)$ and $\Phi_E : E(G) \to E(H)$ such that $\Phi_E(\{u,v\}) = \{\Phi_V(u), \Phi_V(v)\}$ for all $\{u,v\} \in E(G)$, or decide that G and H are not isomorphic.

This problem essentially asks if there is a way to permute the vertex labels of G and H such that their edge sets are the same.

Much of the interest in GraphIsomorphism comes from its unknown complexity status. Indeed, GraphIsomorphism $\in$ NP and is not known to be NP-hard. However, it is unlikely that GraphIsomorphism is NP-hard as this would imply collapse of the polynomial hierarchy [GMW91]. On the other hand, there is no general polynomial-time algorithm for GraphIsomorphism, even after decades of intense research. Interestingly, the SubgraphIsomorphism problem is NP-complete [GJ79, GT48]. SubgraphIsomorphism seeks to find an isomorphism between a subgraph of G and the graph H. It can however be solved in polynomial-time if G is a forest and H is a tree, but remains NP-complete if G is a tree and H is a forest. GraphIsomorphism is the SubgraphIsomorphism problem specialised to the case $|V(G)| = |V(H)|$ and $|E(G)| = |E(H)|$.

The current state of the art for solving GraphIsomorphism with proven runtime complexity is the recent quasipolynomial time algorithm of Babai [Bab15], whereas the most practically used scheme is the `nauty` program of McKay and Piperno [MP14]. Also, there are many classes of graph for which GraphIsomorphism admits a proven polynomial-time algorithm, for instance digraphs with a forbidden minor [Pon91, Gro10]. This deep result includes planar graphs and graphs of bounded genus. Conversely, there are certain classes of graphs for which solving graph isomorphism is as difficult as in the general case, in which case the class is

described as GraphIsomorphism-complete. For instance, deciding if two bipartite graphs are isomorphic is GraphIsomorphism-complete [UTN05]. Many such classes can be found in [ZKT85].

Many of the recent advances in GraphIsomorphism, including Babai's recent breakthrough [Bab15] and the `nauty/traces` programs of McKay and Piperno [MP14] use a group theoretic approach.

A classical approach to graph isomorphism is known as *colour refinement*. The 1-dimensional version goes as follows: the vertices of the graphs are labelled/coloured by their degrees. Then, for each iteration every vertex label is extended by the multiset[14] of the labels of its neighbours. The labels are then replaced by their positions in the lexicographic order of all the occurring labels. The algorithm stops when the (multi-)set of vertex labels stabilises. The multisets of labels of G and H are compared to determine if they are isomorphic. A stable colouring can be found in at most n refinement steps. This can be computed in time $O((n+m)\log n)$, where m is the number of edges [CC82] (note that this method fails for all regular graphs by construction). This method was extended by Weisfeiler and Lehman [WL68] to the so-called *k-dimensional WL method*, which considers k-tuples of vertices. For a long time it was thought that this method solved GraphIsomorphism but had not been proven correct. However, Cai, Fürer and Immerman [CFI92] constructed a family of graphs for which the k-WL method fails to distinguish non-isomorphic graphs.

2.5.4.1 Quantum approaches to graph isomorphism

As discussed earlier, GraphIsomorphism is NP-intermediate. Another famous problem that is NP-intermediate is Factoring: the problem of determining the prime factors of an integer. Indeed, since Shor [Sho97] showed that Factoring $\in$ BQP there have been many attempts to find efficient quantum algorithms for GraphIsomorphism, which we briefly describe below. A good (but incomplete) survey is given in [Bac10]. This work has so far not been successful in delivering an efficient solution to GraphIsomorphism but has stimulated further theoretical and practical developments. The approaches fall into three broad categories.

Hidden subgroup problem. Recall the *hidden subgroup problem*, as solved by

[14] A multiset is merely a set with repeated elements allowed.

Shor's algorithm [Sho97]: you are given query access to a function f from a group G to a set S such that f is constant and distinct on an left cosets of an unknown subgroup H. Find H by querying f. Quantum computers can solve the hidden subgroup problem in time $\operatorname{poly}\log|G|$ when the group G is a finite Abelian group. Indeed, if one can achieve the same for the symmetric group (which is non-Abelian), this yields a polynomial-time algorithm for GraphIsomorphism. Attempts to this end have been documented in [CvD10, EH99].

Index Erasure. The approach based on a procedure called index erasure was first described in print by Aharonov and Ta-Shma [ATS03]. The approach rests on being able to efficiently prepare the (unnormalised) quantum state

$$|\alpha_G\rangle = \sum_{\sigma\in\mathbb{S}_n} |\sigma(G)\rangle, \tag{2.33}$$

where G is a graph on n vertices and $\mathbb{S}_n$ is the symmetric group over n elements. The state $|\alpha_G\rangle$ represents a superposition over all labellings of the graph G for a suitably defined encoding. For two isomorphic graphs, the two states will be the same and for non-isomorphic graphs the states will be orthogonal. A simple circuit (the so-called 'swap test') can distinguish these two scenarios. One might assume that since the uniform distribution over all labellings of a graph is easy to sample from that it is easy to prepare $|\alpha_G\rangle$. However, this line of argument leads to efficient preparation of the state $|\beta_G\rangle = \sum_{\sigma\in\mathbb{S}_n} |\sigma\rangle \otimes |\sigma(G)\rangle$. The task of recovering $|\alpha_G\rangle$ from $|\beta_G\rangle$, that is, "erasing" the index $|\sigma\rangle$ is called *index erasure*, in the oracle setting. Lindzey and Rosmanis have recently proved tight *exponential* lower bounds on this task [LR19], thereby providing strong evidence against this line of attack for finding an efficient quantum algorithm.

Quantum invariants. The final approach considered is a more heuristic one, where graph invariants are designed according to a quantum physical principle. The effectiveness of these methods is hard to prove and has yet to be conclusively demonstrated as effective for GraphIsomorphism in a practical way. Papers such as [Rud02, EHSW06, MRS+17] present various proposals. Indeed, many of these methods can be shown to be equivalent to the k-WL

method for certain k [AIP08, BP09].

Recently, Atserias and colleagues [AMR$^+$19] define the notion of *quantum isomorphism*, inspired by a two-player, non-local game such that classical players can win with the game with certainty if and only if the graphs G and H are isomorphic. Quantum isomorphism derives from perfect quantum strategies for this game. The authors provide examples of quantum isomorphic graphs that are not isomorphic.

Chapter 3

Lifted Markov chains and quantum walks

In this chapter, we set out to answer Research Question 1, which is restated below for convenience.

Research Question 1 (Lifted Markov chains and quantum walks) *Which computational resources are required for a classical random walk to replicate the mixing dynamics of a quantum walk?*

The quantum average mixing distribution allows one to study the long-term behaviour of quantum systems defined on graphs. One may also be interested in sampling from this distribution as an algorithmic primitive, for use in Markov chain Monte Carlo for instance. Particularly now in the so-called Noisy Intermediate Scale [Pre18] (NISQ) era of quantum computation it is important to understand which distributions quantum computers allow us to sample from efficiently, as this is seen to be one of the first applications of quantum computers. By the same token it is also important to know which of these distributions cannot be sampled from efficiently using classical computation. Indeed, it is widely thought (at the time of writing) quantum computational supremacy [HM17] will first be demonstrated via sampling [1].

As discussed in Section 2.4, the quantum average mixing time has been studied in various levels of generality. Lifted Markov chains, briefly addressed in Section 2.3, were devised at roughly the same time as quantum walks and were introduced as a device to speed up the mixing of classical Markov chains, much in the same vein

[1] Not from the average mixing distribution of a quantum walk however.

as quantum walks. As such, it is curious that up until the time of conducting this work there had been no formal comparison made in the literature. We were spurred on by the work of Apers, Sarlette and Ticozzi [ATS17] on lower bounds on mixing times for liftings to carry out this comparison. Indeed, at the same time the work in this chapter was conducted the same authors were concurrently carrying out a similar comparison themselves [AST18], which we discuss further in Section 3.4.

We will briefly summarise our result, then rigorously define lifting in Section 3.1 as a continuation of the discussion in Section 2.3, followed by the full proof of our results in Section 3.3.

We construct a lifted Markov chain that mixes to the quantum average mixing distribution defined in Section 2.4.3.4. We prove that the lifted chain mixes exactly to this distribution in time equal to the diameter of the graph upon which the quantum walk takes place. Moreover, we show that computing the lifting takes time $O(n^8)$, where n is the number of vertices in the graph. Intuitively, this means that a lifted chain can be constructed that simulates the mixing of a quantum walk in a shorter time it takes to carry out the walk. However, using this lifting only confers an advantage over the native quantum walk if the quantum walk takes $T = \Omega(n^8)$ timesteps, taking into account computation of the transition probabilities. More precisely, our lifted chain mixes to the average mixing distribution of a quantum walk of choice; the full result is given in Theorem 3.1 and Corollary 3.1. The average mixing distribution after T timesteps corresponds to sampling uniformly at random a time $t \in \{0,1,\ldots,T-1\}$, running the quantum walk for t timesteps, then measuring the position of the walker. This procedure is used as the basis for a definition of quantum mixing time instead of simply running for T timesteps then measuring, as the latter process does not converge in the limit of infinite T.

The proof of Theorem 3.1 proceeds in the following manner: we begin with a quantum walk on the graph G over T timesteps. We then use a lifting defined by Apers, Ticozzi and Sarlette in [ATS17] which we call the d-lifting, that allows diameter-time mixing to any probability distribution over the vertices of G with full support, taking the quantum average mixing distribution as the target distribution. We further prove that the runtime of computing this lifting is polynomial in n, and that the quantum average mixing distribution has full support for any quantum walk

on a connected m-regular graph.

3.1 Lifting: a rigorous definition

A graph G^c is a *lift*, or *lifting*, of G if there exists a weak homomorphism $c : G^c \to G$. Following Apers, Ticozzi and Sarlette [ATS17], we denote by $c^{-1} : V(G) \to 2^{V(G^c)}$ the map that takes as input the vertex $k \in V(G)$ and outputs the set of nodes $j \in V(G^c)$ for which $c(j) = k$. The homomorphism c induces a linear map from $V(G^c)$ into $V(G)$, which we can represent using the matrix C with elements

$$C[i,j] = \begin{cases} 1, & \text{if } c(j) = i; \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

where $i \in V(G)$, $j \in V(G^c)$. We can now define a lifted Markov chain.

Definition 3.1. *(c-lifted Markov chain)* Let G be a finite, directed graph and let $M_G = (P, p^{(0)})$ be a Markov chain on G. Furthermore, the graph G^c is a lift of G via the mapping $c : G^c \to G$. A *c-lifted Markov chain* for M_G, M_G^c, is the Markov chain $(P^c, p^{c(0)})$ on G^c whose initial distribution $p^{c(0)}$ satisfies $p^{(0)} = p^{c(0)} \cdot C^\mathsf{T}$.

We note as a direct consequence of Definition 3.1 that the transition matrix P^c satisfies $P^c[u,v] = 0 \Longleftarrow \{c(u), c(v)\} \notin E(G) \wedge (c(u) \neq c(v))$. The lifted Markov chain M_G^c proceeds in the usual way, by repeated application of P^c. The probability distribution over $V(G)$ is given at time t by the marginal $p^{(t)} = (p^c)^{(t)} C^\mathsf{T}$. We shall call M_G the *collapsed chain* with respect to the lifted chain M_G^c.

The definition of a c-lifting gives some freedom for the form of P^c and $p^{c(0)}$, even for a fixed homomorphism c. Usually, we will specify the graph G^c, transition matrix P^c and initial distribution $p^{c(0)}$ and refer to this specific configuration as *the* c-lifting.

Suppose we have a lifted Markov chain M_G^c lifted from M_G, with the lifted graph G^c related to G via the homomorphism c. We define the *ϵ-mixing time of the marginal*, $\mathcal{M}_\epsilon^c$, of M_G^c, for $\epsilon > 0$ as

$$\mathcal{M}_\epsilon^c = \max_{p^{c(0)} \in \mathcal{P}^c} \min_{T \in \mathbb{Z}_+} \left\{ T \,\middle|\, \forall t \geq T,\ \left\| p^{c(0)} \cdot (P^c)^t \cdot C^\mathsf{T} - \pi \right\|_{TV} \leq \epsilon \right\}, \tag{3.2}$$

where π is the stationary distribution of M_G, $\mathcal{P}^c$ is the set of allowed starting

distributions of M_G^c and C is the linear map induced by the homomorphism c. Note that $\mathcal{M}_\epsilon^c \leq \mathcal{M}_\epsilon$ for all ϵ. This comes from the following: we do not set $\mathcal{P}^c$ as all basis states in the lifted state space, $\Delta(V(G^c))$, analogously to the definition of $\mathcal{M}_\epsilon$ (indeed, if this were the case we would have equality for all ϵ). Instead, we are allowed to choose a mapping from the initial state on the collapsed Markov chain to an initial state on the lifted chain. The set $\mathcal{P}^c$ is then the image of $\mathcal{P}$ under this mapping. The map is chosen so as to prune the 'bad' starting states from $\mathcal{P}^c$ and give a faster mixing time of the marginal, yielding the inequality. An important result of [ATS17] is that for a lifted Markov chain to give any speedup over its coarse-grained chain *for an arbitrary graph*, we must be allowed to choose this initialisation mapping (see Table 2.2, Scenario S).

We shall say for a lifted chain M_G^c that the *marginal has mixed* at a time T when $\left\| p^{c(0)} \cdot (P^c)^t \cdot C^\mathsf{T} - \pi \right\|_{TV} \leq \epsilon$. Furthermore, a lifted chain may have a marginal that has mixed without itself mixing, that is, $p^{c(0)} \cdot (P^c)^t \cdot C^\mathsf{T}$ will converge to π but $p^{c(0)}(P^c)^t$ won't necessarily converge to its stationary distribution, π^c; indeed π^c doesn't even have to exist [ATS17].

In [CLP99], the authors show that for any Markov chain M_G, every lifted Markov chain satisfies

$$\mathcal{M}_{1/4}^c \geq \frac{1}{2\Phi} \tag{3.3}$$

with an upper bound of $O\Big(\log\Big(\frac{1}{\min_i \pi_i}\Big)\frac{1}{\Phi}\Big)$ in the case of *reversible chains*, Markov chains where the flow through any cut $X \subseteq V(G)$ is the same in both directions. For the bounds above, the proofs show existence of these optimal liftings, but do not provide an efficient (that is, polynomial-time) procedure to construct the lifting. The optimal lifting in [CLP99] relies on the solution of an NP-hard problem. Note also that these bounds hold for the case when $\mathcal{P}^c$ is taken as the set of distributions with all probability mass on any basis state in the lifted space. We have already seen in Section 2.3 that these bounds can be beaten; in Section 3.2 we will see how.

3.2 A lifting with diameter-time mixing

We now introduce another lift, which we call the *d-lifting*, due to Apers, Ticozzi and Sarlette [ATS17]. First, we need the following definition. Let G be a connected, directed graph. The *tensor product*, or *direct product* of graphs G and H, denoted by

$G \otimes H$, has vertex set $V(G) \times V(H)$ and an arc $((i,j),(k,l))$ if and only if $(i,k) \in E(G)$ and $(j,l) \in E(H)$.

Proposition 3.1 (d-lifting [ATS17, Theorem 2]) *Let $M_G = (P, p^{(0)})$ be a Markov chain on a connected graph G on n vertices. Moreover, P has stationary distribution π with all strictly positive elements. Then, there exists a d-lifted Markov chain, M_G^d, on a graph G^d having $D(G) \cdot n^2$ vertices, for which $\mathcal{M}_\epsilon^d \leq D(G)$, where $D(G)$ is the diameter of G and $\epsilon > 0$ is arbitrary.*

Observation 3.1. The lifted chain's marginal mixes to π in $D(G)$ timesteps, to arbitrary precision ϵ, a remarkable fact.

In their statement of this result in [ATS17], the authors stipulate that this lift has certain restrictive properties (it belongs to the SiMRE design scenario of Table 2.1). The first is that the starting distribution for the lift is initialised according to a particular mapping $D_{\text{init}} : \Delta(V(G)) \to \Delta(V(G^d))$ i.e. $\mathcal{P}^d = \left\{ p^{(0)} \cdot \ D_{\text{init}} \,\middle|\, p^{(0)} \in \Delta(V(G)) \right\}$ in the definition of mixing of the marginal (3.2). The proposition does not contradict the conductance lower bound given in [CLP99] as discussed earlier, because this is defined for $\mathcal{P}^c$ being the set of *all* probability distributions over the lifted vertices (or equivalently, distributions with all probability concentrated at basis states). Indeed, our definition of a lifted chain allows us this choice, since $p^{d(0)}$ satisfies $p^{d(0)} = p^{(0)} \cdot \ D_{\text{init}}$ by construction, where d is the linear map induced by the lift homomorphism.. The second restriction is that the lifted chain having a marginal that has mixed does not necessarily imply that the lifted chain itself has mixed. Since we will only care about the marginal mixing to π, this is not important for us. We must also allow for the lifted chain to be reducible, that is, G^d is not a connected graph. Again, this does not concern us.

Now we describe the d-lifting. The lifting rests on the following claim: let G be a graph on n vertices and let $p, p' \in \Delta(V(G))$ be probability distributions on $V(G)$. Then, there is a set of $D(G)$ transition matrices on G, $\{P(i)\}_{i=1}^{D(G)}$, called a *stochastic bridge* such that $p' = pP(1)P(2)\cdots P(D(G)-1)P(D(G))$. We shall prove this claim in Claim 3.1.

To apply the d-lifting, for each vertex of G we create a copy of the graph $G \otimes P_T$, where $T = D(G)$ and P_k is the path graph on k vertices, then take their disjoint union, giving the graph $G^d := \biguplus_{v_0 \in V(G)} G \otimes P_T$. The vertex set $V(G^d) =$

$\{(t, v_0, v) \,|\, t \in \{0, 1, \ldots, D(G)-1\},\ v_0, v \in V(G)\}$ and $d : (t, v_0, v) \mapsto v$. One can think of a vertex (t, v_0, v) as having three registers: a time register, starting vertex and current vertex registers. We provide an illustrative diagram of the lift in Figure 3.1.

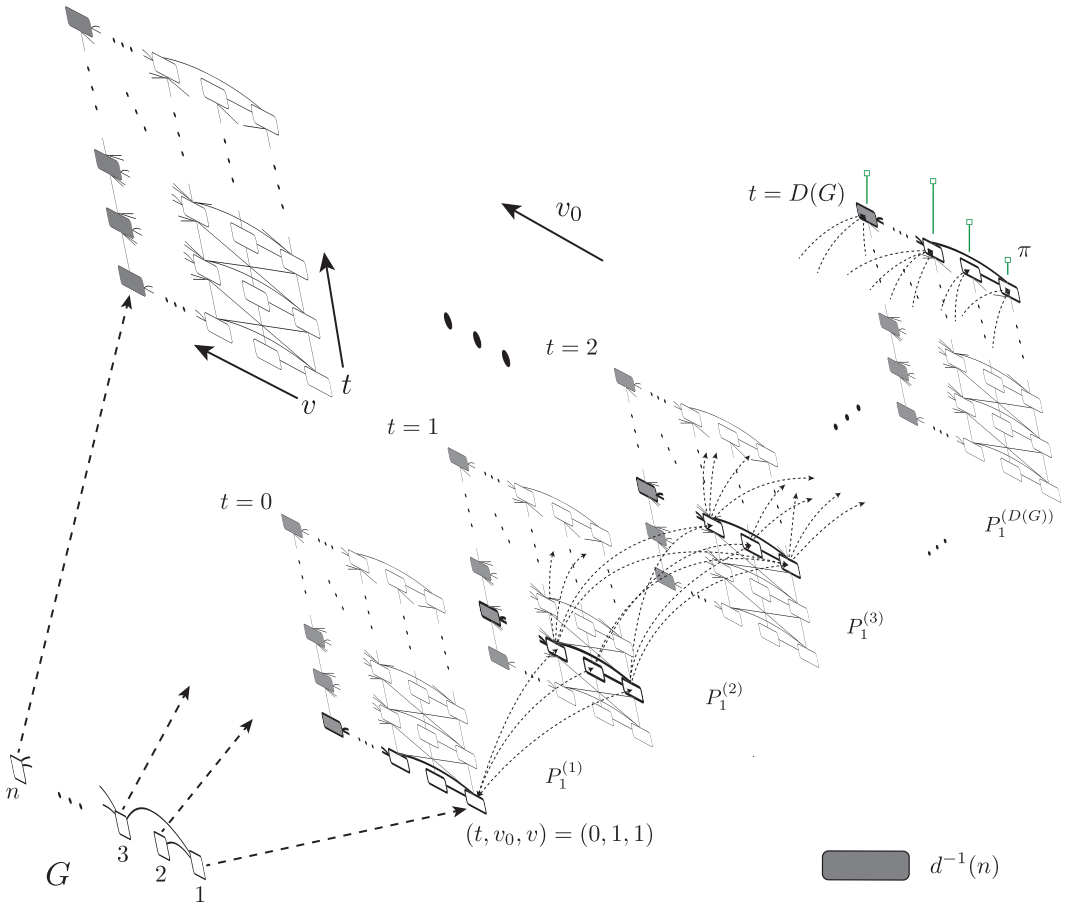

Figure 3.1: Illustration of the d-lifting. For each vertex in $V(G)$ there is a single disjoint copy of the graph $G \otimes P_{D(G)}$. At the bottom right we indicate via shaded vertices $d^{-1}(n)$ for $n \in V(G)$. We illustrate the time evolution of a walk starting at vertex $1 \in V(G)$, whose corresponding starting state in the d-lifted walk is $(t, v_0, v) = (0, 1, 1)$. The evolution, defined by a stochastic bridge $\{P_i^{(t)}\}_{t=1}^{D(G)}$ is depicted with multiple copies of $G \otimes P_{D(G)}$, one for each t, with the corresponding v vertices boldened. We see that the final distribution is π over the appropriate vertices, indicated by the green lines capped with boxes. After $t \geq D(G)$ timesteps, the marginal of the lifted chain has mixed to π and the dynamics proceed according the collapsed transition matrix P. We also indicate with arrows labelled t, v_0 and v the direction in which the lifted vertex indices (t, v_0, v) increase. Self-loops in the drawing are omitted for clarity.

The lifted walk starts by sampling a vertex, $X(0)$, from G according to the initial distribution $p^{(0)}$. Then, we walk on the $X(0)^{\text{th}}$ copy of $G \otimes P_{D(G)}$, starting at the node $(0, X(0), X(0))$. The transition probabilities are engineered using the

stochastic bridge such that t increases by one at each timestep and $P(t)$ is applied to the v space at timestep t. We ensure that the final distribution in the v space is π, the stationary distribution of the chain M_G, by taking $p' = \pi$ and $p = e_{X(0)}$ in Eq. (3.4). Convexity ensures that given an initial distribution of $p(0)$ over the value of v_0, the final distribution of v mixes to π. Marginalising gives us *exactly* the marginal distribution π after $D(G)$ timesteps. After $D(G)$ timesteps the walker stays in the same position. In practise, the stochastic bridge will be attained to some arbitrary precision δ, so we have that the marginal mixes to π for arbitrary δ.

Concretely, the initialisation map for the d-lifting, D_{init}, is

$$D_{\text{init}} : p^{(0)} \mapsto p^{d(0)} = e_0^{\mathsf{T}} \otimes \sum_{v \in V(G)} p^{(0)}[v] \cdot (e_v^{\mathsf{T}} \otimes e_v^{\mathsf{T}})$$

and the lifted transition matrix is given by

$$P^d = \sum_{i \in V(G)} \left(\sum_{t=1}^{D(G)} e_{t-1} e_t^{\mathsf{T}} \otimes e_i e_i^{\mathsf{T}} \otimes P_i^{(t)} + e_{D(G)} e_{D(G)}^{\mathsf{T}} \otimes e_i e_i^{\mathsf{T}} \otimes I_{V(G)} \right)$$

where $P_i^{(t)}$ is the t^{th} stochastic bridge transition matrix for vertex i. The first term is propagation through time and the second term is self-loops, ensuring that when the chain is sampling from π at any time $t \geq D(G)$.

3.2.1 Computing the d-lifting

The d-lifting rests on the existence of a *stochastic bridge* linking two probability distributions together. In the paper [AST18] there is partial construction of such an object. We provide a complete, constructive existence proof below.

Claim 3.1 (Stochastic Bridge) *Let G be a connected graph on n vertices and let $p \in \Delta(V(G))$ be a probability distribution on $V(G)$. Then, there is a set of $D(G)$ transition matrices on G, $\{P_i^{(t)}\}_{t=1}^{D(G)}$, called a* stochastic bridge *such that*

$$p = e_i P_i^{(1)} P_i^{(2)} \cdots P_i^{(D(G)-1)} P_i^{(D(G))} \tag{3.4}$$

for any $i \in V(G)$, where $D(G)$ is the diameter of G.

Claim 3.1 is a direct corollary of the following two lemmas.

Lemma 3.1 *Let G be a graph and $p' \in \Delta(V(G))$ be a probability distribution over the vertices of G. Then, for any $p^{(0)} = e_v^{\mathsf{T}}$, $v \in V(G)$, there exists a sequence of probability distributions $p^{(0)}, p^{(1)}, \ldots, p^{(D(G)-1)}$ of length $D(G)$ such that*

$$\sum_{v \in X} p^{(t+1)}[v] \leq \sum_{v \in X \cup N(X)} p^{(t)}[v]$$

for every $t \in \{0, 1, \ldots, D(G)-1\}$, every subset of vertices $X \subseteq V(G)$ and $p' = p^{(D(G)-1)}$.

Lemma 3.2 *Let G be a graph and $p, p' \in \Delta(V(G))$ be probability distributions over the vertices of G. Moreover, suppose that for every subset of vertices $X \subseteq V(G)$*

$$\sum_{v \in X} p'[v] \leq \sum_{v \in X \cup N(X)} p[v].$$

Then there exists a stochastic matrix P such that $p' = pP$.

Lemma 3.2 is proved in [AST18, Lemma 5], taking inspiration from Aaronson [Aar05]. We prove Lemma 3.1 by construction and reproduce the proof of Lemma 3.2 for completeness.

Proof of Lemma 3.1. For a given vertex u, we can find the sequence of probability distributions taking $p_0 = e_u$ to $p_{D(G)-1} = \pi$ as follows: find a spanning tree T_u of G and set the vertex u as its root. Orient each of the edges such that for each vertex the outgoing edges point towards child vertices and the incoming edge comes from the parent vertex. Attach to each vertex w a chain of vertices with the same label w deep enough so that there is a leaf having each vertex label in $V(G)$, in such a way that every path from the root u to each of the leaf nodes is of the same length, the diameter $D(G)$. We call this modified graph T'_u ; it is still a tree. Moreover, we denote the t^{th} *level* of T'_u, $\ell_t(T'_u)$, the set of vertices in $V(G)$ at distance t from the root u in T'_u. Let the surjection $\sigma_u : V(T'_u) \to V(G)$ return the label of the vertex v in the graph T'_u. Observe that at each level t of T'_u, there is at most one vertex with the same label, that is, $\sigma_u(v) \neq \sigma_u(v')$ for all $v, v' \in \ell_t(T'_u)$, $v \neq v'$.

As an example, consider the graph G in Figure 3.2, with the spanning tree T_1 (rooted at vertex 1) and the related tree T'_1. Let $\mathcal{D}(v, T)$ be the set of descendent leaves of the vertex v in a tree T. We then set our 'bridge schedule' as follows for

$t \in \{0, 1, \ldots, D(G)-2\}$:

$$p^{(t)}[w] = \begin{cases} \sum_{k \in \mathcal{D}(v, T'_u)} \pi[\sigma_u(k)], & v \in \ell_t(T'_u),\ \sigma_u(v) = w; \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

with $p^{D(G)-1}[w] = \pi[w]$ for all $w \in V(G)$. From Eq. 3.5 we see that $p^{(t+1)}[w] \leq p^{(t)}[w]$ for all $w \in V(G)$, since $\ell_t(T'_u) \subseteq \ell_{t+1}(T'_u)$ and $p^{(t)}[w] \geq 0$ for all $t \in \{0, \ldots, D(G)-1\}$, $w \in V(G)$. Now consider an arbitrary subset of vertices $X \subseteq V(G)$. Summing over $w \in X$, we have

$$\sum_{w \in X} p^{(t+1)}[w] \leq \sum_{w \in X} p^{(t)}[w] \leq \sum_{u \in X \cup N(X)} p^{(t)}[w],$$

again using $p^{(t)}[w] \geq 0$. The result follows. □

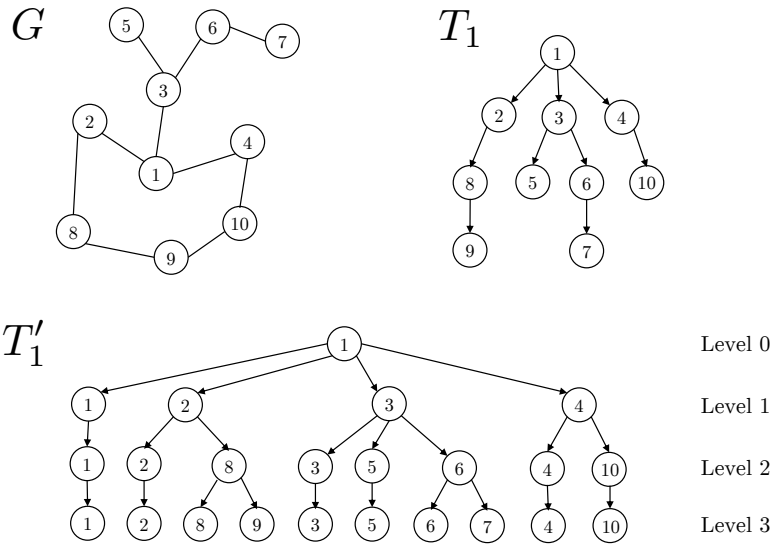

Figure 3.2: Illustration of the construction of the graph T'_u for a graph G used in the proof of Lemma 3.1. First, find a spanning tree T_u of G rooted at vertex u, orienting the edges from the root to the leaves. Then, modify T_u using the following procedure: walk through T_u. To each vertex add a chain of vertices with the same label such that the paths from the root u to every leaf are of the same length and for every vertex in $V(G)$, there is a leaf in T'_u with the same label. The resulting graph is T'_u.

Notice that it is possible to 'prune' the vertices in the d-lifted state space that do not occur at the t^{th} level of each T'_u, that is, vertices $v \in V(T'_u)$ for which $p^{(t)}[\sigma_u(v)] = 0$. To avoid complications we shall not take this into account in the analysis that

follows.

For the proof of Lemma 3.2 we will use results from *flows over capacitated networks*. An intuitive description of the maximum flow problem is to compute the maximum amount of a liquid that can flow through a network of pipes, each with different capacities. We first need the concept of an (s,t)-flow in a directed acyclic graph before describing the maximum flow problem in detail below. Following Korte and Vygen [KV12], we are given a directed graph G, arc capacities $c : E(G) \to \mathbb{R}_+$ and two specified vertices called the source and sink $s,t \in V(G)$. Then, a *flow* is a function $f : E(G) \to \mathbb{R}_+$ with $f(e) \leq c(e)$ for all $e \in E(G)$. The *excess* of a flow f at a vertex $v \in V(G)$ is given by

$$\mathsf{ex}_f(v) := \sum_{e \text{ is an outgoing arc from } v} f(e) \quad - \sum_{e \text{ is an incoming arc to } v} f(e)$$

An (s,t)-flow is a flow satisfying $\mathsf{ex}_f(s) \geq 0$, $\mathsf{ex}_f(t) = -\mathsf{ex}_f(s)$ and $\mathsf{ex}_f(v) = 0$ for all $v \in V(G) \setminus \{s,t\}$. The *value* of an (s,t)-flow is given by $\mathsf{value}(f) = \mathsf{ex}_f(s)$. The maximum-flow problem is defined as follows.

Problem 3.1. MaximumFlow

Instance: A directed graph G, arc capacities $c : E(G) \to \mathbb{R}_+$ and two specified source and sink vertices $s,t \in V(G)$.

Task: Find an (s,t)-flow of maximal value, $f^\star$.

We now have the ingredients to prove Lemma 3.2.

Proof of Lemma 3.2. The proof proceeds by constructing a graph H such that solving the flow problem on H yields the required transition matrix. We refer the reader to Figure 3.3 for an illustration of H. We first make two copies of $V(G)$, named W and W' respectively. An arc is drawn from $u \in W$ to $v \in W'$ if there is a corresponding edge in G, $\{u,v\} \in E(G)$. Each of these arcs has capacity 1. The vertices r and s are introduced, with r being the source node and s being the sink node. For each vertex $v \in W$, an arc is drawn from r to v with capacity $p[v]$. For each vertex $v \in W'$, an arc is drawn from v to s with capacity $p'[v]$. If one unit of flow can be routed from r to s, then the outgoing edges from r and the incoming edges to s will be at maximum capacity, thus encoding a linear mapping of probability mass from p to p'. Concretely, when the maximum flow $f^\star = 1$ we have

$p'[k] = \sum_{j \in V(G)} f^\star((j,k)) = \sum_{j \in V(G)} p[j]P[j,k]$ where $j \in W$ and $k \in W'$, from which we deduce $P[j,k] = f^\star((j,k))/p[j]$. From Claim 2.1 we have that P is stochastic as it is a linear map taking a probability distribution to a probability distribution. It remains to prove that $f^\star = 1$.

The *max-flow min-cut* theorem [FF56] states that the maximum flow $f^\star$ that can be routed from s to r is equal to the minimum cut value of H, $\mathsf{cut}^\star$, where the cut-value is the sum of the edge capacities for a cut disconnecting r from s. Including all of the edges in H connected to r in a cut disconnects the graph and has a cut value of 1, so $\mathsf{cut}^\star \leq 1$. We shall now lower bound $\mathsf{cut}^\star$. If we include any of the middle edges from W to W' in the cut then the value will be greater than 1, so the minimum value cut must be some combination of the edges starting at s or arriving at r. Assume now that we know the optimal cut, and let $X' \subseteq W'$ be the set of vertices in W' connected to s, corresponding to edges that are *not* part of the minimum cut, that is, the edges connecting $W' \setminus X'$ to s belong to the optimal cut. The set $X \subseteq W$ is the set of vertices in W corresponding to the vertices $X' \subseteq W'$, and $N_G(X) = N_H(X')$ is the set of vertices in W corresponding to the neighbourhood of X in G. In order to block all paths from s to r going though $W' \setminus X'$ without including any "middle edges" from W to W', we must include every edge going from from r to $(X \cup N_G(X))$ in the cut. The value of the minimum cut is thus

$$\mathsf{cut}^\star = \sum_{v \in (X \cup N_G(X))} p[v] + \sum_{v \in W' \setminus X'} p'[v] = \sum_{v \in X \cup N_G(X))} p[v] + \left(1 - \sum_{v \in X} p'[v]\right).$$

Now since we have by assumption that

$$\sum_{v \in X} p'[v] \leq \sum_{v \in X \cup N(X)} p[v]$$

for any subset of vertices $X \subseteq V(G)$, we get

$$\mathsf{cut}^\star \geq \sum_{v \in X \cup N_G(X))} p[v] + \left(1 - \sum_{v \in X \cup N_G(X))} p[v]\right) = 1,$$

and so we have $\mathsf{cut}^\star = f^\star = 1$, which yields the result. □

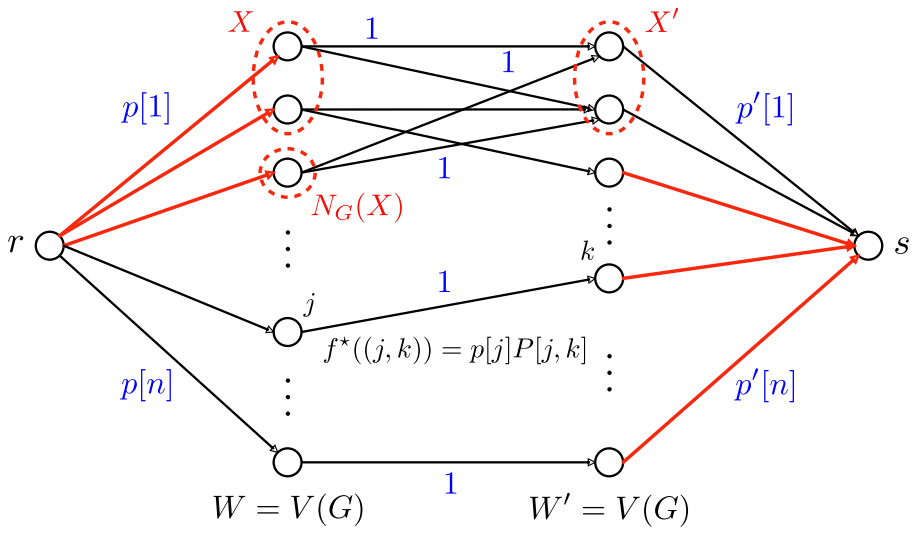

Figure 3.3: Illustration of the graph H used for proving Lemma 3.2. We wish to solve the (r,s)-flow over the graph shown in the figure, where the capacities are shown in blue. The sets W and W' are copies of $V(G)$. An arc is drawn from $u \in W$ to $v \in W'$ if there is a corresponding edge in G, $\{u,v\} \in E(G)$. Each of these arcs has capacity 1. An arc is drawn from r to each vertex in W, and from each vertex in W' to s. These arcs have capacities taken from p and p' as indicated. Solving the maximum flow problem on this graph gives the transition matrix P such that $p' = pP$, where the transition probability from i to j is given by the optimal flow from $i \in W$ to $j \in W'$. The sets marked in red, $X, X', N_G(X)$ are used in the proof of Lemma 3.2. The red arcs indicate the minimum cut used in the proof.

3.2.2 Runtime complexity

Having described the stochastic bridge construction in detail, we can prove the following.

Lemma 3.3 *Let $M_G = (P, p^{(0)})$ be a Markov chain on a connected graph G on n vertices. Moreover, P has stationary distribution π with all strictly positive elements. Then, computing the transition probabilities of the d-lifted Markov chain, M_G^d requires $O(n^4 D(G))$ time, where $D(G)$ is the diameter of G.*

Proof. For the d-lifting of a Markov chain on an n vertex graph, we are required to compute a stochastic bridge corresponding to each vertex, with $p' = \pi$ for every bridge and $p = e_i$ for the i^{th} vertex. We require n stochastic bridges, each containing $D(G)$ $n \times n$ transition matrices.

Solving for each transition matrix requires solving a max-flow problem on $2n+2$ vertices, where certain flows are given by the 'schedule' probabilities (3.5). Computing the schedule probabilities $p^{(t)}[j]$ for a given vertex i involves finding a spanning

tree T_i, walking through T_i, appending vertices, then for each $t \in [D(G)-1]$ summing up the values of the children. The complexity of this task is $O(nD(G))$. Taken over all n stochastic bridges, computing the schedule probabilities takes time $O(n^2 D(G))$.

Maximum-flow problems can be solved in time $O(|V(G)|^3)$ (see Malhotra, Pramodh Kumar, Maheshwari [MKM78]), so the total runtime complexity for solving the max flow problems is $O(n^4 D(G))$ as we solve $nD(G)$ max-flow problems on graphs, each with $2n+2$ vertices. The overall runtime complexity is the sum of the complexities for solving the max-flow problems and computing the schedules. Thus the overall runtime complexity of computing the transition probabilities for the d-lifting on a graph G with n vertices is $O(n^4 D(G))$, since solving the max-flow problems dominates the complexity of computing the schedule probabilities asymptotically. □

3.3 Sampling from the quantum average mixing distribution

We can now think about applying the d-lifting to the quantum average mixing distribution. In this way, we will be able to sample from this distribution in diameter-time (with appropriate pre-computation). We will see the main result in Section 3.3.2.

3.3.1 Computing the quantum average mixing distribution

We work in the coined model of Section 2.4.3.1 and consider an m-regular graph G on n vertices. The initial state of the walk is denoted by $|\psi(0)\rangle \in \mathcal{S}(\mathbb{C}^{mn})$, where a basis state $|k,v\rangle$ represents the walker being at vertex $v \in V(G)$, with k being a "pointer" towards a neighbour of v. The walk matrix $U \in \mathrm{U}(\mathbb{C}^{mn})$ is a unitary matrix such that $U|k,v\rangle$ only has support on basis states $|k',v'\rangle$, where v' is a neighbour of v and k' "points" to a neighbour of v'. We quote a useful identity concerning the elements of the quantum average mixing distribution of a quantum walk on a m-regular graph G on n vertices, derived in Proposition A.1.

$$\pi^q_{\psi(0)}[v] = \sum_r \langle\psi(0)|F_r D_v F_r|\psi(0)\rangle, \tag{3.6}$$

where $D_v \in \mathbb{R}^{mn\times mn}$ is the diagonal matrix with a 1 in positions corresponding to vertex v and zeros elsewhere and the $\{F_r\}$ are the idempotents of the spectral

decomposition of U. Thus, knowing the spectral decomposition of U allows us to compute the quantum average mixing distribution of the walk $\pi^q_{\psi(0)}$.

Let us now consider the computational complexity of computing the quantum average mixing distribution, $\pi^q_{\psi(0)}$, using Eq. (3.6). Computing the spectral decomposition of the transition matrix takes time $O((nm)^3)$. Each term $F_r^\dagger D_k F_r$ takes $O((nm)^2 \cdot m)$ since D_k has only m non-zero terms, taken over $(mn)^2$ elements. Taking $\langle\psi(0)|\cdots|\psi(0)\rangle$ is an additive $O(m^2)$ factor, leaving us at $O(n^2m^3)$. We then have that r can range up to nm and we perform the sum for each $v \in [n]$ for a total of $O((nm)^4)$. Now, taking $m = O(n)$ we have that computing $\pi^q(\psi(0))$ requires runtime $O(n^8)$. This gives us the following lemma.

Lemma 3.4 *Let $(U, |\psi(0)\rangle)$ be a coined quantum walk on a m-regular graph G. Then, computing the quantum average mixing distribution $\pi^q_{\psi(0)}$ takes time $O(n^4m^4) = O(n^8)$.*

We will also need the following result concerning the quantum average mixing distribution.

Lemma 3.5 *Let $(U, |\psi(0)\rangle)$ be a coined quantum walk on a connected m-regular graph G. Then, every element of the quantum average mixing distribution $\pi^q_{\psi(0)}$ is strictly positive.*

Proof. Recall Eq. (3.6), that states $\pi^q_{\psi(0)}[v] = \sum_r \langle\psi(0)|F_r D_v F_r|\psi(0)\rangle$, where $D_v = \sum_{k\in[m]} |k,v\rangle\langle k,v|$ and F_r are the idempotents of the spectral decomposition of U. Notice that

$$\langle\psi(0)|F_r D_v F_r|\psi(0)\rangle = \sum_{k\in[m]} \langle\psi(0)|\, F_r\, |k,v\rangle\langle k,v|\, F_r\, |\psi(0)\rangle = \sum_{k\in[m]} |\langle\psi(0)|\, F_r\, |k,v\rangle|^2. \tag{3.7}$$

Thus, if $\pi^q_{\psi(0)}[v] = 0$, then $\sum_r \sum_{k\in[m]} |\langle\psi(0)|\, F_r\, |k,v\rangle|^2 = 0$ and there exist $u \in V(G)$, $j,k \in [m]$ such that $\sum_r |\langle j,u|\, F_r\, |k,v\rangle|^2 = 0$. This implies that $F_r[(j,u),(k,v)] = 0$ for all r and any linear combination of the F_r has a $((j,u),(k,v))$-component of zero. In this case, for every $t \in \mathbb{N}$ we have $\langle j,u|\, U^t\, |k,v\rangle = 0$. Now this is only true if G is not connected, as it implies there is no path in G of the form $(f_v(k), \ldots, w)$, where w is the j^{th} neighbour of u satisfying $f_w(j) = u$. By contraposition we infer that G being connected implies that $\pi^q_{\psi(0)}[v] > 0$ for all $v \in V(G)$. □

This result should not be surprising since the limiting distribution of a classical ergodic Markov chain has strictly positive elements.

3.3.2 Main Result

We now have all of the pieces to prove the main result.

Theorem 3.1 *Let $(U, |\psi\rangle)$ be a coined quantum walk on a connected m-regular graph G on n vertices. Then there exists a lifted Markov chain on $n^2 D(G)$ vertices with marginal that mixes exactly to the quantum average mixing distribution $\pi^q_{\psi(0)}$ after $D(G)$ timesteps, where $D(G)$ is the diameter of G. Computing the transition probabilities for the lifted Markov chain requires $O(n^4(m^4 + D(G)))$ time.*

Proof. We will use the d-lifting of the Markov chain on G with $\pi^q_{\psi(0)}$ as the target distribution. From Lemma 3.5, $\pi^q_{\psi(0)}$ has strictly positive elements, and so satisfies the conditions for the d-lifting of Proposition 3.1. From Lemmas 3.3 and 3.4 we have that the runtime of computing $\pi^q_{\psi(0)}$ is $O(n^4 m^4)$ and that computing the d-lifting takes $O(n^4 D(G))$ time. □

Indeed, we can also generalise this result to a general quantum walk in the following way.

Corollary 3.1 *Let $(U, |\psi\rangle)$ be a general quantum walk on a connected graph G. Then there exists a lifted Markov chain on $n^2 D(G)$ vertices with marginal that mixes exactly to the quantum average mixing distribution $\pi^q_{\psi(0)}$ after $D(G)$ timesteps, where $D(G)$ is the diameter of G. Computing the transition probabilities for the lifted Markov chain requires $O(n^8)$ time.*

Here, we take $m = O(n)$, $D(G) = O(n)$ and notice that the proofs of Lemmas 3.5 and 3.4 hold for general quantum walks also.

3.4 Related work

Upon completion of this work I became aware of the extended abstract by Apers, Sarlette and Ticozzi in [AST17], which presents a similar result to the one proved here. Their result stated that for any local-stochastic process (a quantum walk is local-stochastic) that mixes to a distribution π in time $\mathcal{M}_\epsilon$ (our notation), there is a lifted chain that mixes to π in time $\mathcal{M}_\epsilon$ with exponential convergence to arbitrary total variation distance $\epsilon > 0$ from π. The proof was not publicly available at that

time. They have subsequently released the paper [AST18] proving the result. Their construction is different to the construction presented here, using the notion of *amplification* from randomised algorithms. In the paper [AST18] there is no discussion of the computational complexity of their constructions; we show that computing the lifting presented here is a polynomial-time operation.

We discuss further work and open questions in Section 5.3.1.

Chapter 4

Graph Products and Isomorphism

In this chapter we describe our approach to answering Research Question 2. As a refresher, we state the relevant question we are trying to answer below.

Research Question 2 (Graph products and isomorphism) *Can we use easy instances of* NP*-hard problems to make progress on the* NP*-intermediate problem* GraphIsomorphism*, which has thus far eluded a polynomial-time algorithm?*

In Section 2.5.4 we have discussed the history of the GraphIsomorphism problem and some of the approaches taken in the literature. We proceed in the direction suggested by Research Question 2, by reducing GraphIsomorphism to solving the clique problem (which is NP-hard) on the weak modular product of the input graphs. Solving the clique problem on perfect graphs is polynomial time. This chapter will contain one the main results of this thesis, namely, the classification of perfect and non-perfect weak modular product graphs. By making this classification we can then evaluate this angle of attack for the GraphIsomorphism problem. It is found that this approach does not lead to a generic efficient algorithm for GraphIsomorphism, but nonetheless has independent interest.

In Sections 4.1 and 4.2 we present the necessary background to construct the candidate algorithm, described in Section 4.3. The full classification is then proved in Section 4.4 followed by discussion in Section 4.5.

In this chapter we consider only *finite, simple* graphs, i.e. graphs with finite number of vertices and no self-loops or multiple edges.

4.1 The Lovász number

In this section much of the presentation follows that in [GW11]. We will first need some definitions. A *clique* is a subset of the vertices of a graph such that any two distinct vertices in the clique are adjacent. An *independent set* in a graph is a subset of the vertices such that no two vertices in the subset are adjacent. A *proper colouring* of a graph G is an assignment of colours to vertices such that every pair of adjacent vertices has a different colour. Given a graph G on n vertices, we are interested in the following quantities:

- The *clique number* of G, written as $\omega(G)$, is the size of the largest clique in G.
- The *independence number* of G, written as $\alpha(G)$, is the size of the largest independent set in G.
- The *chromatic number* of G, written as $\chi(G)$, is the minimum number of colours required to properly colour G.
- The *clique cover number* of G, written as $\bar{\chi}(G)$, is the size of the smallest clique cover in G, which is the minimum number of vertex disjoint cliques such that every vertex is in some clique.

Recall that the complement of a graph G, denoted $\overline{G}$, is the graph on the same vertices as G such that two vertices are connected in $\overline{G}$ if and only if they are not connected in G.

The following will be useful:

Observation 4.1. Let G be a graph on n vertices. Then,

1. $\alpha(G) = \omega(\overline{G})$
2. $\chi(G) = \bar{\chi}(\overline{G})$
3. $\omega(G) \leq \chi(G)$
4. $\alpha(G) \leq \bar{\chi}(G)$

Now we give the definition of a perfect graph, first stated by Berge.

Definition 4.1 (Perfect graphs)**.** A graph G is *perfect* if $\omega(G') = \chi(G')$ for all induced subgraphs G' of G.

Notable perfect graphs include bipartite graphs, their line graphs, chordal graphs and interval graphs [GLS93].

Lovász introduced a function ϑ for which $\alpha(G) \leq \vartheta(G) \leq \bar{\chi}(G)$ [Lov79], which we shall develop here. We begin by developing an SDP relaxation for $\bar{\chi}(G)$. We assign a unit vector v_i to each vertex of G. If two vertices are in the same clique of the minimum clique cover, we demand their vectors to be the same. If two vertices are not in the same clique, we demand their vectors to be as far apart as possible. Note that when k vectors are as spread out as far as possible[1], the dot product of any pair of them is $-\frac{1}{k-1}$. This means that if we have a clique cover of size k, there is an assignment of unit vectors to vertices such that every vertex in a clique is mapped to the same vector and, if two vertices are not in the same clique, the dot product of their vectors is $-\frac{1}{k-1}$.

This suggests the following SDP relaxation for the clique cover number:

$$\begin{aligned} \text{minimise} \quad & k \\ \text{subject to} \quad & \langle v_i, v_j \rangle = -\frac{1}{k-1} \quad i,j \in V(G),\ i \not\sim j,\ i \neq j \\ & \langle v_i, v_i \rangle = 1 \qquad\qquad \forall i \in V(G) \end{aligned} \tag{4.1}$$

While this doesn't follow the standard form of an SDP (2.25), the reader will note that the constraint implicitly define a standard form SDP, where the Gram matrix of the set of vectors $\{v_1, \ldots, v_{|V(G)|}\}$ is the optimisation variable. See the proof of Lemma A.2 for more on this. We can now define the Lovász ϑ function.

Definition 4.2 (Lovász-ϑ)**.** For a graph G, $\vartheta(G)$ is the optimal value of the SDP in (4.1).

We observe here that there are many different, but equivalent formulations for the Lovász-ϑ function. These are all described and their equality proved in [GLS93, Section 9.3].

Importantly, we have the following result. We defer the proof to Proposition A.2 in the appendix.

Proposition 4.1 (Sandwich theorem) *Let G be a graph. Then, $\alpha(G) \leq \vartheta(G) \leq \bar{\chi}(G)$.*

[1] This is proved rigorously in Corollary A.2, which is omitted here so as to not interrupt the main text.

Note that $\vartheta(G)$ may not be rational for non-perfect graphs. For example, $\bar{\chi}(C_5) = 3$, $\alpha(C_5) = 2$, and $\vartheta(C_5) = \sqrt{5}$. So we cannot hope to get the exact optimum for every graph. However, we can solve the program (4.1) to arbitrary accuracy using the ellipsoid algorithm [GLS93], resulting in the following result.

Proposition 4.2 *For any $\epsilon > 0$, $\vartheta(G)$ can be computed to within ϵ error in time* $\text{poly}(n, \log \frac{1}{\epsilon})$.

The polynomial-time computability of the values of the parameters α, ω, χ, and $\bar{\chi}$ for perfect graphs directly follows from Observation 4.1 since we have that $\omega(\overline{G}) \leq \vartheta(G) \leq \chi(\overline{G})$, by taking complements we can compute $\omega(G) = \chi(G)$ for any perfect G.

Corollary 4.1 *For any perfect graph G, $\alpha(G)$, $\omega(G)$, $\chi(G)$, and $\bar{\chi}(G)$ can be computed in polynomial time.*

4.1.1 Non-perfect graphs and Lovász number

We can also ask how closely ϑ approximates α for non-perfect graphs. Konyagin [Kon81] constructed a graph G such that $\alpha(G) = 2$ and $\vartheta(G) = \Omega(n^{1/3})$, which is the largest that $\vartheta(G)$ can be. Alon and Kahale [AK98] generalised this result for bounded α by proving the following: if $\alpha(G) \leq k$, then $\vartheta(G) \leq Cn^{\frac{k-1}{k+1}}$ for some constant C. When α is not bounded, Feige [Fei97] shows that there exists a graph G such that $\alpha(G) = n^{o(1)}$ and $\vartheta(G) = n^{1-o(1)}$. Håstad's results for the hardness of approximating the clique problem [Hås99] also imply that such a graph must exist. Kleinberg and Goemans showed that ϑ gives a 2-approximation for the size of the minimum vertex cover [KG98]. However, this is not very useful as the greedy algorithm for minimum vertex cover gives a 2-approximation. There are graphs for which this bound is tight, so we can do no better in general.

In Figure 4.1 we investigate the gap between α, ϑ and $\overline{\chi}$ empirically and it is found that for the majority of the 1,424,776 graphs tested there is no gap.

4.1.2 Lovász number with a quantum computer?

In light of the discussion of Section 2.5.3.2, one may well think that since the Lovász number is formulated as an SDP, one can gain a computational speedup in using a quantum SDP solver. For now, generic solvers cannot achieve this. These solvers all depend polynomially on a parameter of the SDP at hand called the *width*.

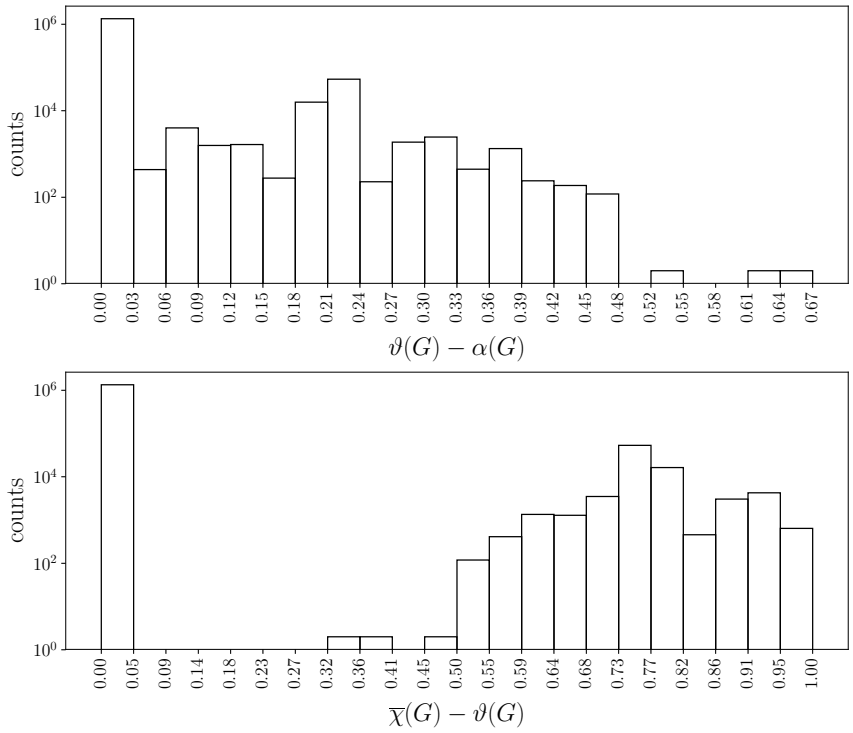

Figure 4.1: Numerical experiment to explore the gap between α, ϑ and $\overline{\chi}$. 1,424,776 graphs were generated in `sage` [SAGE] using the `graphs()` function and their α, ϑ and $\overline{\chi}$ computed. We see that for the majority of graphs G (approx 1.0×10^6 out of 1.4×10^6) we have that $\alpha(G) = \vartheta(G)$ or $\vartheta(G) = \overline{\chi}(G)$.

In [vGGdW17], the authors show that for a family of SDPs having a property called *combinability*, the width is linear in the dimension of the SDP. It follows that the quantum advantage disappears for these composable SDPs. Indeed, the Lovász-ϑ family of SDPs is composable and so has no quantum speedup, at least with general solvers. Perhaps a speedup would be possible with a specialised membership oracle.

4.2 Graph products

A *graph product* is a binary operation on graphs. Concretely, a graph product takes two graphs G_0 and G_1 and outputs a graph H with the following properties:

- The vertex set of H is the Cartesian product of the vertex sets of G_0 and G_1, namely, $V(H) = V(G_0) \times V(G_1)$.
- Two vertices (u_0, v_0), (u_1, v_1) are connected by an edge if and only if the vertices u_0, u_1, v_0, v_1 satisfy some condition related to the adjacency structure of G_0 and G_1. This condition is what differentiates the different graph products.

We call G_0 and G_1 the *factors* of the product graph H.

Graph products have been extensively studied and are of vast theoretical and practical interest, see, e.g., Hammack, Imrich and Klavžar [HIK11]. A common problem is to determine how graph invariants such as the independence number and clique number behave under the action of a particular graph product. For instance, a famous result of Lovász [Lov79] states that for graphs G and H, $\vartheta(G \boxtimes H) = \vartheta(G)\vartheta(H)$, where $\vartheta(\cdot)$ denotes the Lovász number and $\boxtimes$ is the strong graph product. More recently, a major open problem in graph theory related to graph products, Hedetniemi's conjecture, has been solved by Shitov [Shi19], decades after the problem was posed. It is straightforward to show that

$$\chi(G \otimes H) \leq \min\{\chi(G), \chi(H)\},$$

where $\otimes$ is the graph tensor product. It was long conjectured that equality holds for all G, H. Shitov [Shi19] presented a family of graphs for which the inequality is strict.

In [HIK11, Section 4.4] there is a full classification of all possible graph products. We present only the most salient points. There are four graph products that have received most attention in the literature: the strong graph product, the direct (or tensor) product, the Cartesian product and the lexicographic. These graph products are defined as follows.

Definition 4.3 (Direct, Cartesian, strong, lexicographic graph products)**.** Let G and H be graphs. We now define the following graph products on the vertex set $V(G) \times V(H)$:

- The *direct*, or *tensor product* of G and H, denoted by $G \otimes H$, has an edge $\{(x,y),(x',y')\}$ if and only if $\{x,x'\} \in E(G)$ and $\{y,y'\} \in E(H)$.
- The *Cartesian product* of G and H, denoted by $G \,\square\, H$ has an edge $\{(x,y),(x',y')\}$ if and only if either: $x = x'$ and $\{y,y'\} \in E(H)$; or $y = y'$ and $\{x,x'\} \in E(G)$.
- The *strong product* of G and H, denoted by $G \boxtimes H$ has an edge $\{(x,y),(x',y')\}$ if and only if $\{(x,y),(x',y')\} \in E(G \otimes H) \cup E(G \,\square\, H)$.
- The *lexicographic* product of G and H, denoted by $G \circ H$ has an edge

$\{(x,y),(x',y')\}$ if and only if either: $\{x,x'\} \in E(G)$; or $x = x'$ and $\{y,y'\} \in E(H)$.

These four graph products are the most studied as they satisfy the following: they are associative and projections onto the factors are weak homomorphisms[2]. Loosely speaking, the second property means that the adjacency structure of the product graph allows one to approximately infer the adjacency structure of the factor graphs. A *projection onto a factor* $\Pi_i : G_0 * G_1 \to G_i$ for some graph product $*$ is a map $\Pi_i : V(G_0) \times V(G_1) \to V(G_i)$ that takes a vertex in the direct product to a vertex in its factor like so:

$$\Pi_0 : (x,y) \mapsto x, \qquad \Pi_1 : (x,y) \mapsto y.$$

A graph product $*$ is associative if it satisfies $(G * H) * K = G * (H * K)$.

One can also consider graph products that do not satisfy these properties. One such product is the *weak modular product*, whose adjacency structure also includes information about non-adjacency in the factor graphs. This product plays a crucial role in this chapter of the thesis.

4.3 Isomorphism via the weak modular product

The weak modular product (see, e.g., Hammack, Imrich and Klavžar [HIK11]) of graphs G and H is defined as follows.

Definition 4.4 (Weak modular product)**.** Let G and H be graphs. The *weak modular product* of G and H, denoted by $G \nabla H$, has vertex set $V(G \nabla H) = V(G) \times V(H)$ and an edge $\{(x,y),(x',y')\}$ if and only if

1. either $\{x,x'\} \in E(G)$ and $\{y,y'\} \in E(H)$;

2. or $\{x,x'\} \in E(\overline{G})$ and $\{y,y'\} \in E(\overline{H})$.

The next statement is a direct consequence of the definitions of the weak modular product and the tensor product.

[2]Note that the lexicographic product does *not* satisfy this second property. Projection onto only the first, but not second, factor is a weak homomorphism.

Lemma 4.1 *For graphs G and H,*

$$G \nabla H = G \otimes H \cup \overline{G} \otimes \overline{H}. \tag{4.2}$$

Interestingly, as originally proved by Kozen [Koz78], a clique of a certain size exists in the product graph if and only if the factors are isomorphic. For completeness we state and prove Kozen's theorem, in modern language.

Proposition 4.3 (Kozen [Koz78]). *Let G and H be graphs on n vertices. Then $\omega(G \nabla H) \leq n$. Moreover, $\omega(G \nabla H) = n$ if and only if $G \cong H$.*

Proof. To see that there is no clique in $G \nabla H$ larger than n consider the following. First lay the vertices of $G \nabla H$ in an $n \times n$ grid so that the vertex (x, y) is in the same row as (x', y') if $x = x'$, and in the same column if $y = y'$. Then by the definition of the weak modular product there can be no edges between vertices in the same row or in the same column. The vertices of an n-clique will thus occupy positions on the grid such that no two vertices are in the same row or column. No larger clique can exist since there is no position in the grid where one can place a new vertex such that it does not share a row or column with any of the vertices already in the clique.

Now suppose there is an n-clique in $G \nabla H$. The vertices (x, y) in the clique represent the bijection $x \mapsto y$ for all $x \in V(G)$, $y \in V(H)$, which we denote σ. We can see that σ is an isomorphism because for all $x, x' \in V(G)$, $\sigma(x) \sim \sigma(x')$ if and only if $x \sim x'$, from the definition of the weak modular product. For the converse, suppose that $G \cong H$, with $\sigma : V(G) \to V(H)$ an isomorphism. Then from the definition of the weak modular product, we will have the collection of edges

$$\{\{(x, \sigma(x)), (x', \sigma(x'))\} \,|\, x, x' \in V(G),\ x' \neq x\} \subseteq E(G \nabla H).$$

This collection of edges induces an n-clique in $G \nabla H$ from the definitions of the weak modular product and isomorphism. Thus, an n-clique exists if and only if $G \cong H$. □

Proposition 4.3 shows that for two graphs on n vertices G and H, deciding if an n-clique exists in $G \nabla H$ is equivalent to deciding if $G \cong H$. The decision version of finding the clique number of a general graph is NP-complete. Indeed

in the paper containing Proposition 4.3 [Koz78] Kozen also proved that, under the same assumptions, deciding if a clique of size $(n-\epsilon)$ exists in $G \nabla H$ for arbitrary fixed $\epsilon > 0$ is NP-complete. This result has largely been ignored in the literature with reference to graph isomorphism as it provides evidence against an efficient algorithm for GraphIsomorphism being found via this route.

It is well known that computing the clique number of a perfect graph is polynomial-time in n for perfect graphs, via Lovász's sandwich theorem as discussed in Section 4.1.

Observation 4.2. Let G and H be graphs. If $G \nabla H$ is perfect, deciding if $G \cong H$ is polynomial-time in n.

Inspired by Observation 4.2, we evaluate all perfect weak modular product graphs in the hope it sheds some insight into solvable cases of GraphIsomorphism. Note that GraphIsomorphism for perfect graphs is GraphIsomorphism-complete, since deciding if two bipartite graphs are isomorphic is GraphIsomorphism-complete [UTN05] and all bipartite graphs are perfect.

We enumerate all pairs of graphs for which the weak modular product is perfect, using theoretical and computational tools that were not available to Kozen in 1978, most notably the Strong Perfect Graph Theorem [CRST06] amongst others. This adds to a tradition of enumerating perfect product graphs; Ravindra and Parthasarathy [RP77] and Ravindra [Rav78] found all perfect Cartesian, direct and strong product graphs. We shall prove the following result.

Theorem 4.1 *The graph $G = G_0 \nabla G_1$ is perfect if and only if one of the following holds:*

1. $G_z \in \{K_1, K_2, E_2\}$, $G_{\overline{z}}$ *arbitrary;*
2. $G_z \cong P_4$, $G_{\overline{z}} \in \{K_{1,r}, K_r \uplus K_1, P_4\}$;
3. $G_z \cong C_5$, $G_{\overline{z}} \in \{P_3, K_2 \uplus E_1, P_4, C_5\}$;
4. $G_z \cong K_r \uplus K_s$, $G_{\overline{z}}$ *is a disjoint union of stars and cliques;*
5. $G_z \cong K_{m,n}$, $G_{\overline{z}}$ *is connected and* $(P_4, \mathit{cricket}, \mathit{dart}, \mathit{hourglass})$*-free;*
6. $G_z \cong K_n$, $G_{\overline{z}}$ *(odd hole, paw)-free;*

7. $G_z \cong E_n$, $G_{\overline{z}}$ *(odd antihole, co-paw)-free;*

8. G_z, $G_{\overline{z}}$ *are complete multipartite;*

9. G_z, $G_{\overline{z}}$ *are disjoint unions of cliques;*

10. $G_z \cong K_r \uplus K_s$, $G_{\overline{z}} \cong K_{m,n}$*;*

for any m,n,r,s,z*, where* $m,n,r,s \in \mathbb{N}$*, and* $z \in \{0,1\}$*, with its (Boolean) negation denoted by* $\overline{z}$*.*

The remainder of this chapter constitutes the proof and necessary ingredients. We follow up with discussion in Section 4.5.

4.4 Perfect weak modular products

In this section, we prove Theorem 4.1, namely we enumerate the pairs (G,H) for which $G \nabla H$ is perfect. We will need some further definitions and results.

4.4.1 Classes of graphs

We now provide definitions and characterisations of families of graphs that will be of use later.

We use standard notation for named graphs; for instance, the complete, empty and path graphs on n vertices are denoted by K_n, E_n and P_n respectively. A graph G is said to be *H-free* if it does not contain H as an induced subgraph. For a named graph, we prepend the prefix *"co-"* to denote its complement, e.g., the complement of a bipartite graph is a co-bipartite graph.

A graph G on n vertices is said to be *bipartite* if $V(G) = V_0(G) \uplus V_1(G)$ such that if $x \sim x'$ then $x \in V_0(G)$ and $x' \in V_1(G)$ or $x \in V_1(G)$ and $x' \in V_0(G)$, for every $x,x' \in V(G)$. The sets $V_0(G)$ and $V_1(G)$ are said to be the *partite sets* of G. A *complete bipartite* graph $K_{m,n}$, where $|V_0(G)| = m$, $|V_1(G)| = n$, is a bipartite graph G for which every vertex in $V_0(G)$ is connected to every vertex in $V_1(G)$. A *star* is a complete bipartite graph where at least one of the partite sets has only one vertex. A *complete multipartite* graph $K_{n_1,n_2,\ldots,n_k}$ is defined similarly, with the relaxation that there are now $k \geq 2$ partite sets. Any induced subgraph covering two disjoint partite sets of a complete multipartite graph is complete bipartite.

A *triangle* is the graph K_3. A *hole* is an induced cycle. An *antihole* is an induced co-cycle. An *odd* (anti)hole has an odd number of vertices. The *diamond* or $K_{1,1,2}$, *paw* or Y, *cricket*, *dart* and *hourglass* graphs are defined in Figure 4.2.

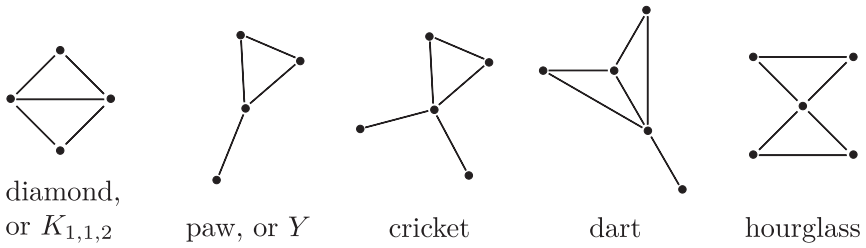

Figure 4.2: Some named graphs.

Observation 4.3. The complements of the diamond, paw, cricket, dart and hourglass are respectively: $K_2 \uplus E_2$, $P_3 \uplus E_1$, $K_{1,1,2} \uplus E_1$, $Y \uplus E_1$ and $C_4 \uplus E_1$.

We will use the following observation many times in the sequel, so often in fact we will refrain from quoting it.

Observation 4.4. The graph G is H-free if and only if $\overline{G}$ is $\overline{H}$-free.

Lemma 4.2 *Let G be nonbipartite and triangle-free. Then, G has an induced odd cycle of order ≥ 5.*

Proof. Since G is not bipartite it contains an odd cycle. Moreover, G is triangle-free so this odd cycle must have order ≥ 5. Call C the smallest odd cycle in G. Consider a chord e in C: since C has an odd number of vertices, the subgraph induced on the union of C and e contains two cycles: an even cycle and an odd cycle. This gives a contradiction since the odd cycle is smaller than C, but C is the smallest odd cycle in G. Thus, there is no chord in C and C is an induced subgraph. □

Lemma 4.3 *A graph G is a disjoint union of cliques if and only if it has no induced P_3.*

Proof. Suppose G is the disjoint union of cliques. Then, every induced subgraph on 3 vertices is either: K_3, $K_2 \uplus E_1$, or E_3, so clearly is P_3-free. Now suppose G contains P_3 as an induced subgraph. Then, we have two vertices in the same connected component that are not connected, and so G is not the disjoint union of cliques. □

The following lemma is a direct consequence of the relevant definitions.

Lemma 4.4 *A graph G is complete multipartite if and only if its complement is a disjoint union of cliques.*

Corollary 4.2 *A graph G is complete multipartite if and only if it is $(K_2 \uplus E_1)$-free.*

Proof. Combine Lemma 4.3 with Lemma 4.4. □

Lemma 4.5 *Any connected bipartite graph G that is not complete has an induced P_4.*

Proof. Let $u \in V_0(G)$ and $v \in V_1(G)$. The shortest path $P(u,v)$ between u and v has odd length. If the length of this path is 1 for all pairs (u,v), G is complete bipartite; else, the length of $P(u,v)$ is 3 or greater for some pair (u,v), and so we have an induced P_{2k} for some $k \geq 2$, proving the lemma. □

Lemma 4.6 *A complete multipartite graph G is diamond-free if and only if it is a clique or complete bipartite.*

Proof. ($\Rightarrow$) We prove the contrapositive. Suppose G is complete multipartite and not bipartite and not a clique. G must have at $k \geq 3$ partite sets, for if it has two partite sets it would be complete bipartite. Now if every partite set in G has one vertex then $G \cong K_k$, but G is not a clique by assumption. So, G has an induced $K_{1,1,2}$, a diamond.

($\Leftarrow$) If G is a clique then any induced subgraph on 4 vertices is isomorphic to $K_4 \ncong K_{1,1,2}$. Bipartite graphs are triangle-free so are trivially diamond-free. □

Lemma 4.7 (Paw-free graphs [Ola88, Theorem 1]) *A graph G is a paw-free graph if and only if each component of G is triangle-free or complete multipartite.*

Lemma 4.8 *If G is connected and (P_4, paw)-free, then it is complete multipartite.*

Proof. From Lemma 4.7, if G is connected and paw-free, then it is complete multipartite or triangle-free. If G is triangle-free then it is bipartite, for otherwise it has an odd hole, but G is P_4-free and every odd hole has an induced P_4. So, G must be bipartite if it is triangle-free. Moreover, since G is P_4-free, if it is bipartite then it is complete bipartite from Lemma 4.5. □

Lemma 4.9 *A graph G is a disjoint union of cliques and stars with two or more connected components if and only if its complement $\overline{G}$ is connected and* $(P_4, \textit{cricket, dart, hourglass})$*-free.*

Proof. ($\Rightarrow$) If G is a disjoint union of two or more cliques and stars, $\overline{G}$ is connected, by the following: suppose G has $k \geq 2$ connected components, i.e. $V(G) := V_1 \uplus \cdots \uplus V_k$ where the V_i are pairwise disconnected. If we have $x \in V_i$, $x' \in V_j$ for $i \neq j$, then $x \sim_{\overline{G}} x'$. If $x, x' \in V_i$, choose any $x'' \in V_j$ for $i \neq j$ and (x, x'', x') is a path in $\overline{G}$. Since there is a path between any $x, x' \in V(\overline{G})$, $\overline{G}$ is connected. We have that G is P_4-free, since a disjoint union of cliques is P_3-free by Lemma 4.3, disjoint unions of complete bipartite graphs are P_4-free by Lemma 4.5. Thus, $\overline{G}$ is P_4-free as P_4 is self-complementary. Now, $\overline{G}$ is (cricket, dart, hourglass)-free if and only if G is $(K_{1,1,2} \uplus E_1, Y \uplus E_1, C_4 \uplus E_1)$-free, which is necessarily true if G is $(\text{diamond}, \text{paw}, K_{2,2})$-free. This is satisfied when G is a disjoint union of stars and cliques, by considering each of the connected components piecewise.

($\Leftarrow$) We prove the contrapositive. Suppose G has one connected component. Then, $\overline{G}$ is disconnected or contains an induced P_4, since by [Sei74, Aux. Thm.], [Ler72], the complement of a connected P_4-free graph is disconnected. Now suppose G has two or more components. If G has an induced paw then it also has an induced $Y \uplus E_1$ and so $\overline{G}$ contains an induced dart by Observation 4.3. If G is paw-free then G is a disjoint union of complete multipartite graphs by Lemma 4.8. Let G be paw-free. If G has an induced diamond then it contains an induced $K_{1,1,2} \uplus E_1$ and so $\overline{G}$ contains an induced cricket. If G has no induced diamond then it is a disjoint union of cliques and complete bipartite graphs by Lemma 4.6. If G contains an induced $K_{2,2} \cong C_4$ then it has an induced $C_4 \uplus E_1$ and so $\overline{G}$ has an induced hourglass by Observation 4.3. We are left with G being the disjoint union of two or more stars and cliques, in which case $\overline{G}$ is $(P_4, \text{cricket, dart, hourglass})$-free and connected, so the result is proven. □

Lemma 4.10 *Let G be a complete multipartite graph. Then it is connected and* $(P_4, \text{dart}, \text{cricket}, \text{hourglass})$*-free.*

Proof. Trivially, G is connected. Now, observe that every induced subgraph of a complete multipartite graph is complete multipartite. Equivalently, if G has an induced subgraph that is not complete multipartite, G is not complete multipartite.

If G contains an induced $X \in \{P_4, \text{dart}, \text{cricket}, \text{hourglass}\}$, then we have a contradiction, since X is not complete multipartite. □

Lemma 4.11 *A graph G is (odd-hole, paw)-free if and only if each component of G is bipartite or complete multipartite.*

Proof. ($\Rightarrow$) If G is paw-free then by Lemma 4.7 each component is triangle-free or complete multipartite. Let X be any triangle-free component of G. Since X is odd-hole free, X is bipartite by Lemma 4.2.

($\Leftarrow$) Let X be a given component of G. Suppose X is bipartite. By definition, X has no odd hole since it contains no odd cycles. Moreover, the paw contains a triangle as an induced subgraph so it cannot be an induced subgraph of X. Now suppose X is complete multipartite. Whence, every induced subgraph of X is complete multipartite also. An induced odd hole or paw in X gives a contradiction, since neither of these graphs is complete multipartite. □

4.4.2 Auxiliary results

We list in this section results that will be used throughout the proof of Theorem 4.1. Proposition 4.4, the strong perfect graph theorem, will be of particular utility. The sequel follows directly from definitions.

Lemma 4.12 *The product graph $G \nabla H$ is perfect if and only if $\overline{G} \nabla \overline{H}$ is perfect.*

Proposition 4.4 (Strong Perfect Graph Theorem [CRST06]) *A graph G is perfect if and only if it has no odd holes or odd antiholes.*

Corollary 4.3 *A graph G is perfect if and only if both G and $\overline{G}$ have no odd holes.*

The sequel follows as a corollary of the strong perfect graph theorem, but was originally proved by Lovász in 1972 [Lov72].

Proposition 4.5 (Weak Perfect Graph Theorem) *A graph G is perfect if and only if $\overline{G}$ is perfect.*

Proposition 4.6 (Cameron, Edmonds and Lovász [CEL86, Theorem 1′]) *Let G_1 and G_2 be perfect graphs and $G := G_1 \cup G_2$ be their union with $V(G_1) = V(G_2) = V(G)$. Suppose that for any $x, x', x'' \in V(G)$, $\{x, x'\} \in E(G_1)$ and $\{x', x''\} \in E(G_2)$ implies that $\{x, x''\} \in E(G)$. Then, G is perfect.*

Proposition 4.7 (Ravindra, Parthasarathy [RP77, Theorem 3.2]) *The graph $G_1 \otimes G_2$ is perfect if and only if either*

1. *G_1 or G_2 is bipartite, or*
2. *both G_1 and G_2 are (odd hole, paw)-free.*

Corollary 4.4 *The graph $G \nabla K_n$ is perfect if and only if either: $n = 1$, $n = 2$, or G is (odd hole, paw)-free.*

Proof. From definitions, $G \nabla K_n = G \otimes K_n$. For $n = 1$ and $n = 2$, K_n is bipartite. For $n \geq 3$, observe that K_n is (odd hole, paw)-free. □

4.4.2.1 Imperfect weak modular products

For the proof of Theorem 4.1, we will need to enumerate many pairs of graphs whose weak modular product is not perfect. The main proof technique used is to find an offending odd hole or antihole in a given product graph. By the strong perfect graph theorem (Proposition 4.4), the product is thus not perfect. Lemma 4.13 drastically reduces the work required.

Lemma 4.13 *Suppose X is an induced subgraph of G and Y is an induced subgraph of H. Then, if $X \nabla Y$ is not perfect, $G \nabla H$ is not perfect. Equivalently, if $G \nabla H$ is perfect, then $X \nabla Y$ is perfect.*

Proof. We prove the contrapositive statement, that is, $G \nabla H$ being perfect implies perfection of $X \nabla Y$. Let $X = G[A]$ and $Y = H[B]$, where $A, B \subseteq V(G)$. First, we claim that $G[A] \nabla H[B] = (G \nabla H)[A \times B]$, that is, $X \nabla Y$ is an induced subgraph of $G \nabla H$. By the strong perfect graph theorem (Corollary 4.3) $G \nabla H$ is perfect and thus has no odd holes or antiholes; since $X \nabla Y$ is an induced subgraph of $G \nabla H$ it too has no odd holes or antiholes, so is perfect by the same result.

It remains to prove the claim, we do so by showing that the vertex sets and edge sets of $G[A] \nabla H[B]$ and $(G \nabla H)[A \times B]$ are identical. Now, $V(G[A] \nabla H[B]) = V(G[A]) \times V(H[B]) = A \times B$. We also have that $V((G \nabla H)[A \times B]) = A \times B$, so the vertex sets are identical. For the edge sets, from definitions we have

$$\begin{aligned} E(G[A] \nabla H[B]) = \{\{(x,y),(x',y')\} \,|\, \{x,x'\} \in E(G[A]) \wedge \{y,y'\} \in E(H[B])\} \\ \cup \left\{\{(x,y),(x',y')\} \,\middle|\, \{x,x'\} \in E(\overline{G}[A]) \wedge \{y,y'\} \in E(\overline{H}[B])\right\} \end{aligned} \tag{4.3}$$

and

$$
\begin{aligned}
E((G\nabla H)[A\times B]) &= \{\{(x,y),(x',y')\}\,|\,\{(x,y),(x',y')\}\in E(G\nabla H)\\
&\qquad\qquad \wedge (x,x'\in A)\wedge(y,y'\in B)\,\}\\
&= \{\{(x,y),(x',y')\}\,|\,\{(x,y),(x',y')\}\in E(G\nabla H)\\
&\qquad\qquad \wedge (x,x'\in V(G[A]))\wedge(y,y'\in V(H[B]))\,\}\\
&= \{\,\{(x,x'),(y,y')\}\,|\,\{x,x'\}\in E(G)\wedge\{y,y'\}\in E(H)\\
&\qquad\qquad \wedge (x,x'\in V(G[A]))\wedge(y,y'\in V(H[B]))\,\} \qquad (4.4)\\
&\cup\{\,\{(x,x'),(y,y')\}\,|\,\{x,x'\}\in E(\overline{G})\wedge\{y,y'\}\in E(\overline{H})\\
&\qquad\qquad \wedge (x,x'\in V(G[A]))\wedge(y,y'\in V(H[B]))\,\}\\
&= \{\{(x,y),(x',y')\}\,|\,\{x,x'\}\in E(G[A])\wedge\{y,y'\}\in E(H[B])\}\\
&\cup\Big\{\{(x,y),(x',y')\}\,\Big|\,\{x,x'\}\in E(\overline{G}[A])\wedge\{y,y'\}\in E(\overline{H}[B])\Big\}.
\end{aligned}
$$

Comparison of Eqs. 4.3 and 4.4 shows that $E(G[A]\nabla H[B]) = E((G\nabla H)[A\times B])$, proving the claim. □

It follows from Lemma 4.13 that if we have graph families Γ_X and Γ_Y such that every $G\in\Gamma_X$ contains an induced X and every $H\in\Gamma_Y$ contains an induced Y, where $X\nabla Y$ is not perfect, every pair in $\Gamma_X\times\Gamma_Y$ has an imperfect weak modular product. Indeed in the sequel we shall use this observation repeatedly on pairs of graph families $\Gamma_X\times\Gamma_Y$ to rule out non-perfect weak modular products.

For the upcoming results, we require the notion of an *augment* of a graph.

Definition 4.5 (Augment of a graph)**.** Let G be a graph on n vertices. A graph G' is an *augment* of G if $|V(G')| = n+1$ and G is an induced subgraph of G'.

Lemma 4.14 *Let G be a triangle-free augment of C_5. Then, $G\nabla P_3$ is not perfect.*

Proof. Figure 4.3 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

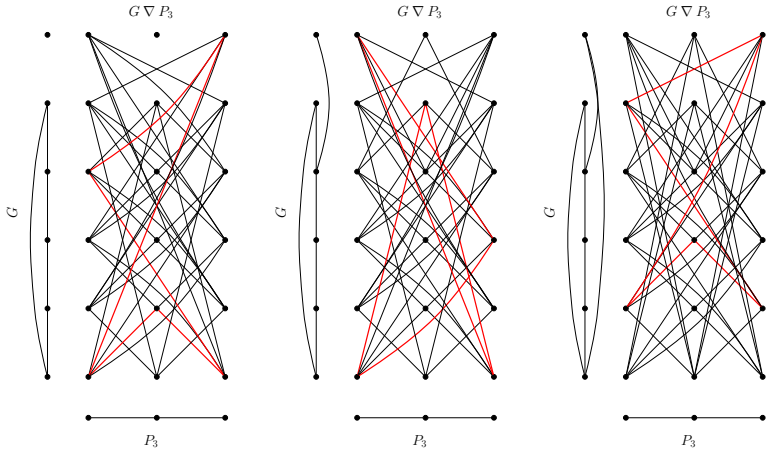

Figure 4.3: Weak modular product of G and P_3, where G is a triangle-free augment of C_5. An induced odd hole/antihole is denoted by a red, thick line.

□

Lemma 4.15 *Let G be a triangle-free augment of C_5. Then, $G \nabla (K_2 \uplus E_1)$ is not perfect.*

Proof. Figure 4.4 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

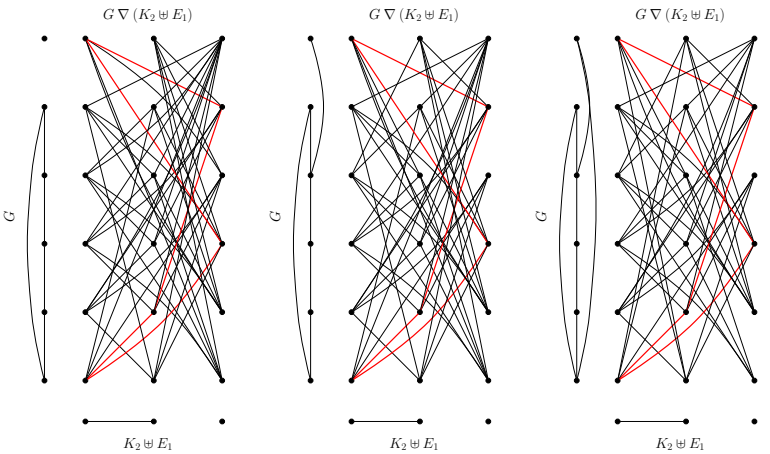

Figure 4.4: Weak modular product of G and $K_2 \uplus E_1$, where G is a triangle-free augment of C_5. An induced odd hole/antihole is denoted by a red, thick line.

□

Lemma 4.16 *Let G be a bipartite augment of P_4. Then, $G \nabla P_3$ is not perfect.*

Proof. Figure 4.5 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

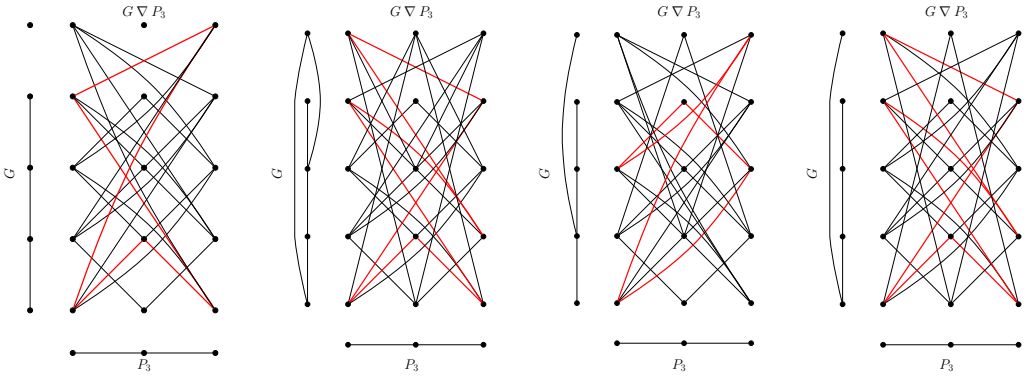

Figure 4.5: Weak modular product of G and P_3, where G is a bipartite augment of P_4. An induced odd hole/antihole is denoted by a red, thick line.

□

Lemma 4.17 *Let G be a bipartite augment of P_4. Then, $G \nabla (K_2 \uplus E_1)$ is not perfect.*

Proof. Figure 4.6 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

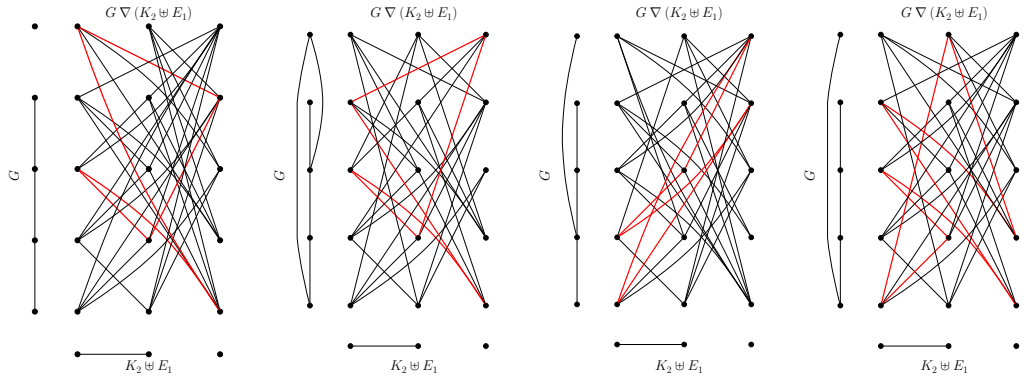

Figure 4.6: Weak modular product of G and $K_2 \uplus E_1$, where G is a bipartite augment of P_4. An induced odd hole/antihole is denoted by a red, thick line.

□

Lemma 4.18 *Let $G \in \{$cricket, dart, hourglass$\}$. Then, $G \nabla P_3$ is not perfect.*

Proof. Figure 4.7 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

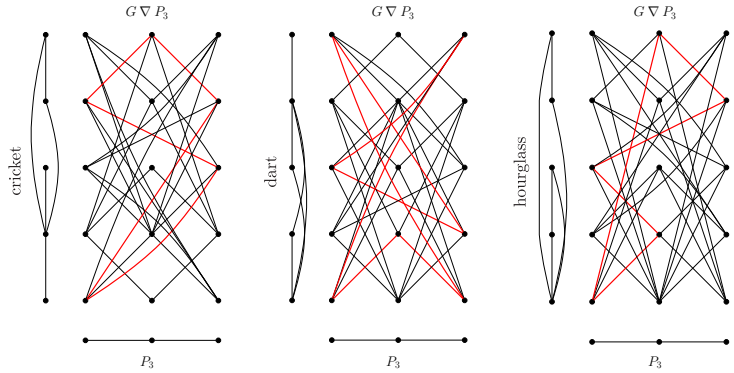

Figure 4.7: Weak modular product of G and P_3, where $G \in \{\text{cricket, dart, hourglass}\}$. An induced odd hole/antihole is denoted by a red, thick line.

□

Lemma 4.19 *Let $G \in \{K_{1,1,2}, Y, P_4 \uplus E_1, K_{2,2} \uplus E_1, P_5\}$. Then $(K_2 \uplus E_1) \nabla G$ is not perfect.*

Proof. Figure 4.8 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

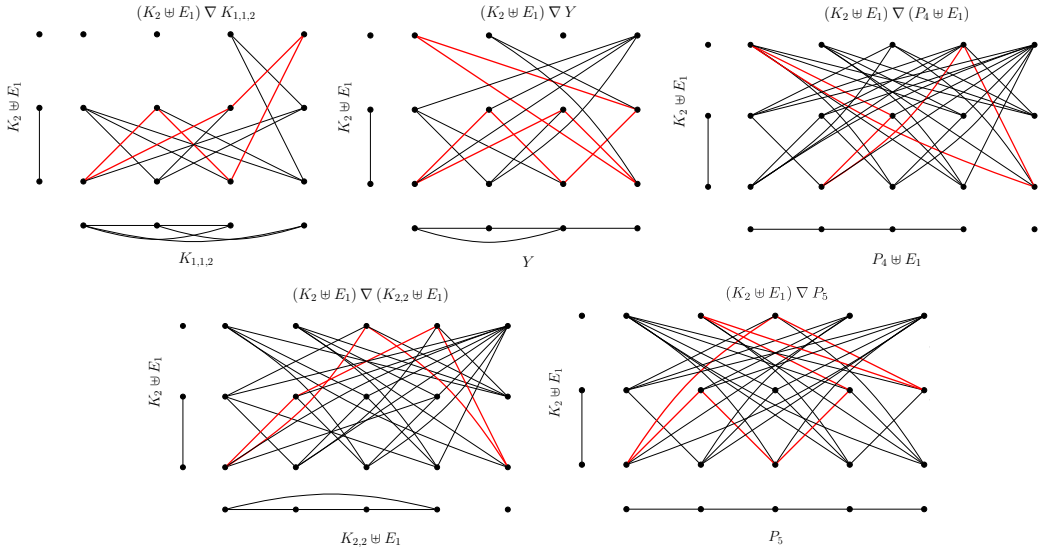

Figure 4.8: Weak modular product of G and $K_2 \uplus E_1$, where $G \in \{K_{1,1,2}, Y, P_4 \uplus E_1, K_{2,2} \uplus E_1, P_5\}$. An induced odd hole/antihole is denoted by a red, thick line.

□

Lemma 4.20 *Let $G \in \{K_2 \uplus E_2, P_3 \uplus E_1, P_5, K_{1,1,2} \uplus E_1, 3K_2\}$. Then $P_3 \nabla G$ is not perfect.*

Proof. Figure 4.9 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

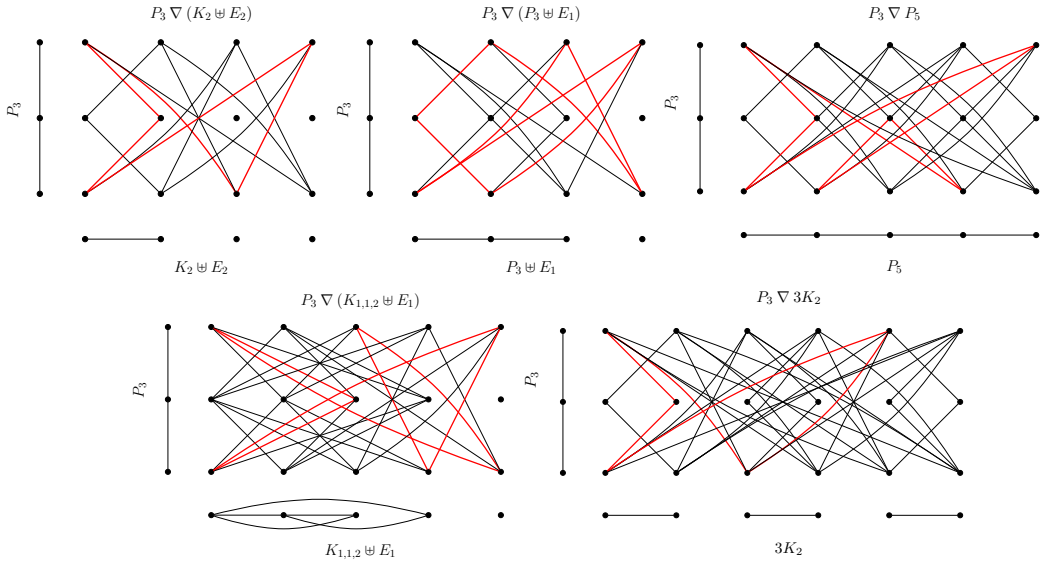

Figure 4.9: Weak modular product of G and P_3, where $G \in \{K_2 \uplus E_2, P_3 \uplus E_1, P_5, K_{1,1,2} \uplus E_1, 3K_2\}$. An induced odd hole/antihole is denoted by a red, thick line.

□

Lemma 4.21 *Let $G \in \{2K_2, Y, K_{1,1,2}, K_{2,2}\}$. Then $P_4 \nabla G$ is not perfect.*

Proof. Figure 4.10 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

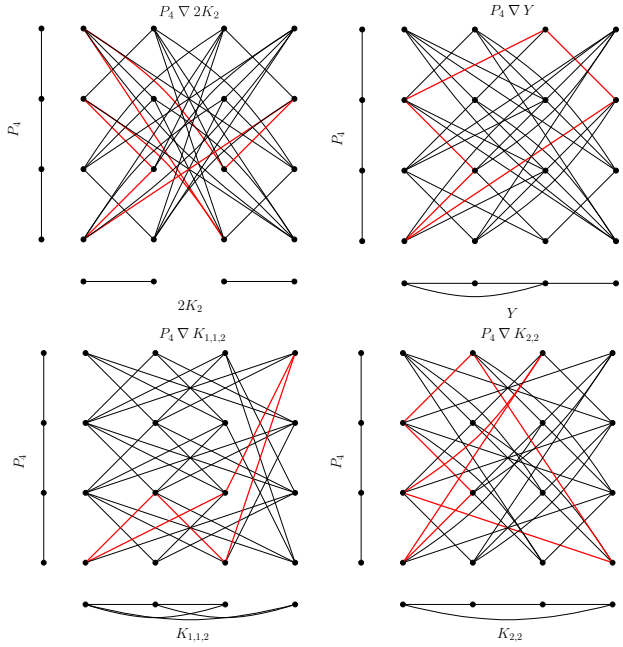

Figure 4.10: Weak modular product of G and P_4, where $G \in \{2K_2, Y, K_{1,1,2}, K_{2,2}\}$. An induced odd hole/antihole is denoted by a red, thick line.

□

Lemma 4.22 *Let $G \in \{P_3 \uplus E_1, K_2 \uplus E_2\}$. Then $K_{2,2} \nabla G$ is not perfect.*

Proof. Figure 4.11 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

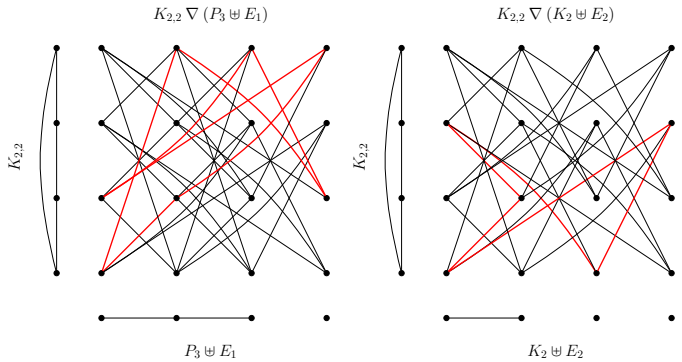

Figure 4.11: Weak modular product of G and $K_{2,2}$, where $G \in \{P_3 \uplus E_1, K_2 \uplus E_2\}$. An induced odd hole/antihole is denoted by a red, thick line.

□

Lemma 4.23 *Let $G \in \{K_3, 2K_2, K_{1,3}, K_{2,2}\}$. Then $C_5 \nabla G$ is not perfect.*

Proof. Figure 4.12 shows the relevant graph products with induced odd holes and antiholes. The lemma follows from the strong perfect graph theorem.

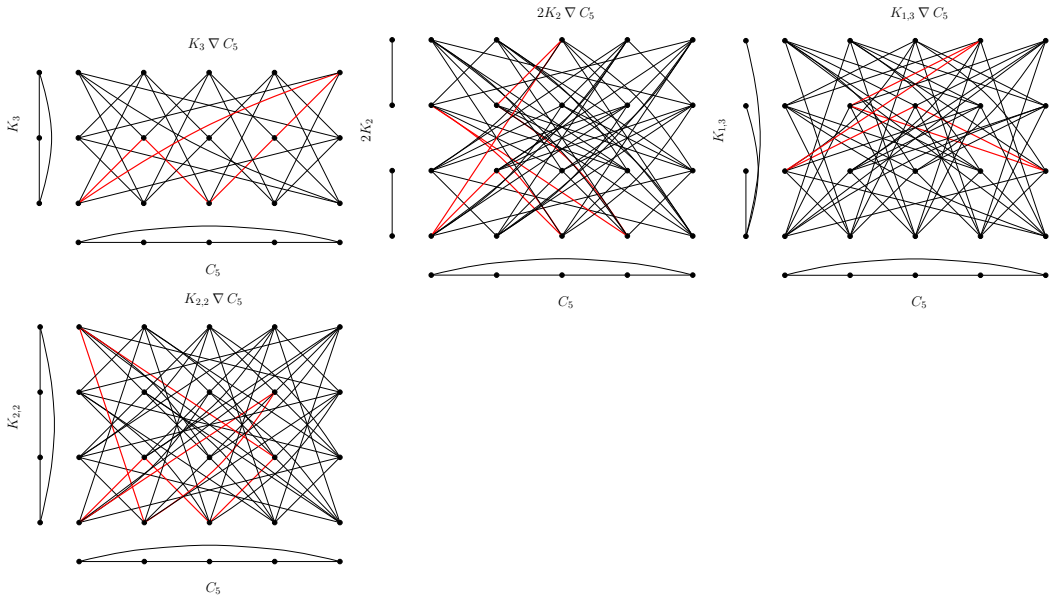

Figure 4.12: Weak modular product of G and C_5, where $G \in \{K_3, 2K_2, K_{1,3}, K_{2,2}\}$. An induced odd hole/antihole is denoted by a red, thick line.

□

4.4.2.2 Perfect weak modular products

In this section we find perfect weak modular product graphs, using tools from previous sections, notably Proposition 4.6.

For the next lemma, we require the concept of a *line graph*. For a graph G, its line graph $L(G)$ is the graph where $V(L(G)) = E(G)$ and $\{e_1, e_2\} \in E(L(G))$ if and only if the edges $e_1, e_2 \in E(G)$ share a vertex in $V(G)$.

Lemma 4.24 *Suppose G is a graph containing an induced P_3. Then, $G \nabla C_5$ is perfect if and only if $G \in \{P_3, P_4, C_5\}$.*

Proof. ($\Leftarrow$) As demonstrated by Figure 4.13 and Table 4.1, $C_5 \nabla C_5$ is the line graph $L(M)$ of the graph M, where M is defined in Figure 4.13. Observe further that M is bipartite, with partite sets $\{0,3,5,7,8\}$ and $\{1,2,4,6,9\}$ using the labelling of Figure 4.13. It is a well-known result that line graphs of bipartite graphs are perfect, and so $C_5 \nabla C_5$ is perfect. Moreover, since P_3 and P_4 are induced subgraphs of C_5, $P_3 \nabla C_5$ and $P_4 \nabla C_5$ are perfect.

($\Rightarrow$) We prove the contrapositive. First observe that $C_5 \nabla K_3$ is not perfect, by Corollary 4.4. Let G be triangle-free and suppose G is not bipartite. Then, by Lemma 4.2 G has an induced odd hole, in which case G contains an induced C_5 or an induced P_5. In the former case, $G \nabla C_5$ is perfect only if $G \cong C_5$, for if G contains a triangle-free augment of C_5, then $G \nabla C_5$ is not perfect by Lemma 4.14. In the latter case, from Lemma 4.20 $P_3 \nabla P_5$ is not perfect and so $C_5 \nabla G$ is not perfect. Now suppose G is bipartite. Then, by Lemma 4.5 G either has an induced P_4 or is a disjoint union of complete bipartites. In the former case we see $G \nabla C_5$ is perfect only if $G \nabla P_4$, for if G contains a bipartite augment of P_4, then $G \nabla P_4$ is not perfect by Lemma 4.16. In the latter case, G either contains an induced $K_{2,2}$ or is a disjoint union of stars. If G contains an induced $K_{2,2}$, $G \nabla C_5$ is not perfect by Lemma 4.23. If G is a disjoint union of stars and $G \ncong P_3$, G contains either an induced $P_3 \uplus E_1$ or an induced $K_{1,3}$. Then, $G \nabla C_5$ is not perfect by Lemmas 4.20 and 4.23 respectively. □

	$(\cdot,1)$	$(\cdot,2)$	$(\cdot,3)$	$(\cdot,4)$	$(\cdot,5)$
$(1,\cdot)$	$\{0,1\}$	$\{5,6\}$	$\{2,8\}$	$\{3,9\}$	$\{4,7\}$
$(2,\cdot)$	$\{6,7\}$	$\{0,2\}$	$\{5,9\}$	$\{4,8\}$	$\{1,3\}$
$(3,\cdot)$	$\{2,3\}$	$\{7,9\}$	$\{0,4\}$	$\{1,5\}$	$\{6,8\}$
$(4,\cdot)$	$\{8,9\}$	$\{3,4\}$	$\{1,7\}$	$\{0,6\}$	$\{2,5\}$
$(5,\cdot)$	$\{4,5\}$	$\{1,8\}$	$\{3,6\}$	$\{2,7\}$	$\{0,9\}$

Table 4.1: Assignment of vertices in $C_5 \nabla C_5$ to edges in M, with vertex labels imposed in Figure 4.13. Two vertices in $C_5 \nabla C_5$ are adjacent if and only if the corresponding edges in M per the table share a common vertex. Thus $C_5 \nabla C_5 \cong L(M)$, where $L(M)$ is the line graph of M.

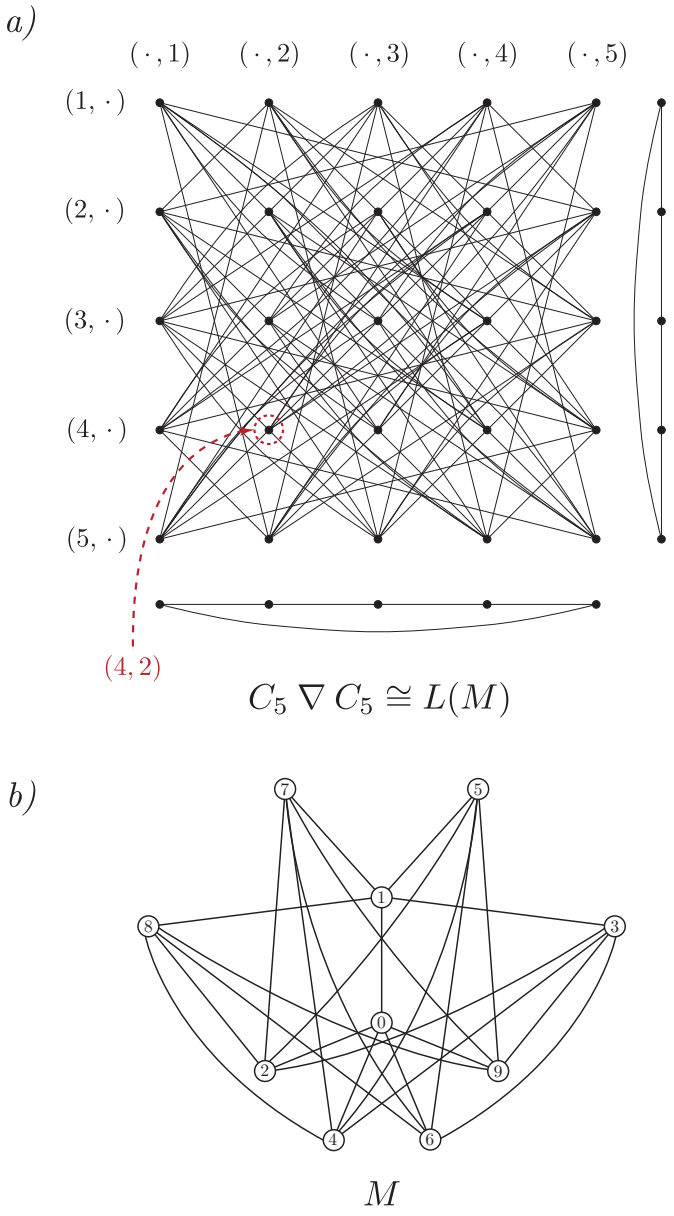

Figure 4.13: *a)* The graph $C_5 \nabla C_5$ with vertex labelling indicated. *b)* The graph we denote by M, with vertex labels. The graph $C_5 \nabla C_5$ is isomorphic to the line graph of M, i.e. $C_5 \nabla C_5 \cong L(M)$. The assignment of vertices in $C_5 \nabla C_5$ to edges in M is given by Table 4.1.

It is interesting to note that the graph $C_5 * C_5$ is not perfect, where $*$ is the strong, direct, Cartesian or lexicographic product [RP77, Rav78].

Corollary 4.5 *The graph $C_5 \nabla (K_2 \uplus E_1)$ is perfect.*

Proof. The graph $C_5 \nabla P_3$ is perfect taken with Lemma 4.12 and self-complementarity of C_5. □

Lemma 4.25 *Let G and H be the disjoint union of cliques. Then, $G \nabla H$ is perfect.*

Proof. We shall proceed by using Proposition 4.6, taking G_1 in the theorem statement as $G \otimes H$ and correspondingly, G_2 as $\overline{G} \otimes \overline{H}$. The assumptions of the theorem are satisfied, namely, G_1 and G_2 are perfect. This follows from Proposition 4.7, since G, $\overline{G}$, H and $\overline{H}$ are (odd hole, paw)-free.

Now call $G = \biguplus_i^k K_{r_i}$ and $H = \biguplus_j^\ell K_{s_j}$. It remains to show that $(x,y) \sim_{G\otimes H} (x',y')$ and $(x',y') \sim_{\overline{G}\otimes\overline{H}} (x'',y'')$ implies $(x,y) \sim_{G\otimes H \cup \overline{G}\otimes\overline{H}} (x'',y'') \equiv (x,y) \sim_{G\nabla H} (x'',y'')$, where $x,x',x'' \in V(G)$ and $y,y',y'' \in V(H)$. Now, denote the subsets of vertices comprising the cliques in G by U_i respectively, that is, the i^{th} clique of G is induced on the vertex set U_i. The j^{th} clique of H, K_{n_j}, is induced on the vertex set V_j.

From definitions, we have that $(x,y) \sim_{G\otimes H} (x',y')$ if and only if $x \in U_i$, $x' \in U_i$, $x \neq x'$ for some $i \in [k]$, and $y \in V_j$, $y' \in V_j$, $y \neq y'$ for some $j \in [\ell]$. Moreover, we have that $(x,y) \sim_{\overline{G}\otimes\overline{H}} (x',y')$ if and only if $x \in U_i$, $x' \in U_{i'}$ for $i \neq i'$, $i,i' \in [k]$ and $y \in V_j$, $y' \in V_{j'}$ for $j \neq j'$, $j,j' \in [\ell]$.

Suppose the edge $\{(x,y),(x',y')\}$ exists in $G \otimes H$ and $\{(x',y'),(x'',y'')\}$ exists in $\overline{G} \otimes \overline{H}$. We have that $x \in U_i$, $x' \in U_i$ for $x \neq x'$, $x'' \in U_{\tilde{i}}$ for $i \neq \tilde{i}$ and $y \in V_j$, $y' \in V_j$ for $y \neq y'$, $y'' \in V_{\tilde{j}}$ for $j \neq \tilde{j}$. The vertices x and x'' are in different cliques, and y and y'' are in different cliques. Thus we have that $(x,y) \sim_{\overline{G}\otimes\overline{H}} (x'',y'')$ giving that $(x,y) \sim_{G\otimes H\cup\overline{G}\otimes\overline{H}} (x'',y'')$, and so $G \nabla H$ is perfect by Proposition 4.6. □

Corollary 4.6 *Let G and H be complete multipartite graphs. Then, $G \nabla H$ is perfect.*

Proof. Take complements and use Lemmas 4.12 and 4.4. □

Lemma 4.26 *The graph $(K_r \uplus K_s) \nabla K_{m,n}$ is perfect.*

Proof. Let $G = (K_r \uplus K_s)$ and $H = K_{m,n}$. Call the vertices in G comprising the cliques U_0 and U_1 respectively, such that $V(G) = U_0 \uplus U_1$. Likewise, for brevity call the vertices in H comprising the partite sets V_0 and V_1 respectively, so that $V(H) = V_0 \uplus V_1$. From the definition of the weak modular product $G \nabla H$, we have that a vertex $(x,y) \in U_{z_1} \times V_{z_2}$ is adjacent to a vertex in $(x',y') \in U_{z_3} \times V_{z_4}$ if and only if $x \neq x'$, $y \neq y'$ and either: $z_1 = z_3$ and $z_2 \neq z_4$; or $z_1 \neq z_3$ and $z_2 = z_4$ for $z_1,z_2,z_3,z_4 \in \{0,1\}$. There are no other edges. Notice that $G \nabla H$ comprises a

complete bipartite graph with partite sets $U_0 \times V_0 \cup U_1 \times V_1$ and $U_0 \times V_1 \cup U_1 \times V_0$, with a perfect matching removed. Since $G \nabla H$ is bipartite, it is perfect. □

We now require a number of auxiliary lemmas to prove Proposition 4.8.

Lemma 4.27 *Suppose* $(K_r \uplus K_s) \nabla (G \uplus K_1)$ *and* $(K_r \uplus K_s) \nabla (H \uplus K_1)$ *are perfect. Then,* $(K_r \uplus K_s) \nabla (G \uplus H)$ *is perfect.*

Proof. For brevity, we denote $(K_r \uplus K_s) \nabla (G \uplus H)$ by Λ and $V(\Lambda)$ by U. We draw the structure of Λ in Figure 4.14. Observe that U is partitioned into four disjoint subsets of vertices: $V(K_r) \times V(G)$, $V(K_r) \times V(H)$, $V(K_s) \times V(G)$ and $V(K_s) \times V(H)$, which we respectively denote U_1, U_2, U_3 and U_4. Now for the sake of contradiction suppose Λ is not perfect. Then, by the strong perfect graph theorem it contains an induced odd hole or antihole, which we call X. The vertices of X, $V(X)$, cannot lie solely in one partitioned subset of U, for in this case X is an induced subgraph of a perfect graph and we have a contradiction. Each of the $\Lambda[U_i]$ is perfect by assumption.

Suppose now X has vertices in two of the partitioned subsets, *i.e.* $V(X) \subseteq U_i \cup U_j$ for $i,j \in \{1,2,3,4\}$, $i \neq j$ and $V(X) \not\subseteq U_i$, $V(X) \not\subseteq U_j$. If $V(X) \subseteq U_1 \cup U_2$ or $V(X) \subseteq U_3 \cup U_4$, then X is an induced subgraph of a disjoint union of perfect graphs and we have a contradiction. If $V(X) \subseteq U_1 \cup U_3$ or $V(X) \subseteq U_2 \cup U_4$ then X is an induced subgraph of $(K_r \uplus K_s) \nabla G$ or $(K_r \uplus K_s) \nabla H$ respectively, which are perfect by assumption, yielding a contradiction. We also obtain a contradiction when $V(X) \subseteq U_1 \cup U_4$ or $V(X) \subseteq U_2 \cup U_3$, as $\overline{\Lambda}[U_1 \cup U_4]$ and $\overline{\Lambda}[U_2 \cup U_3]$ are disjoint unions of perfect graphs and so $\Lambda[U_1 \cup U_4]$, $\Lambda[U_2 \cup U_3]$ are perfect by the weak perfect graph theorem, Proposition 4.5.

Assume now that X has vertices lying in three of the partitioned subsets, *i.e.* $V(X) \cap U_i = \emptyset$ for exactly one $i \in \{1,2,3,4\}$. Furthermore, let $i = 4$ without loss of generality, so that $V(X) \subseteq U_1 \cup U_2 \cup U_3$. Observe that every vertex of U_2 is adjacent to every vertex of U_3 in Λ by definition, and vice versa. Moreover, U_2 is adjacent to every vertex of U_1 in $\overline{\Lambda}$ and vice versa. Suppose now that at least two vertices of X lie in U_2. By assumption there is at least one vertex of X in each of U_3 and U_1. Thus, $\Lambda[V(X)]$ and $\overline{\Lambda}[V(X)]$ both contain a triangle and we have a contradiction with Corollary 4.3, as neither X nor $\overline{X}$ are an odd hole. Now suppose only one vertex of X lies in U_2. Thus, X is an induced subgraph of $(K_r \uplus K_s) \nabla (G \uplus K_1)$ and we have a contradiction since $(K_r \uplus K_s) \nabla (G \uplus K_1)$ is perfect by assumption.

Finally, assume there is a vertex from X in every partition of U, *i.e.* $V(X) \cap U_i \neq \emptyset$ for every $i \in \{1,2,3,4\}$. By the pigeonhole principle, there is at least one $j \in \{1,2,3,4\}$ such that U_j has two or more vertices from X since $|X| \geq 5$. Moreover, for any j there exist $j', j'' \in \{1,2,3,4\}$ where $j \neq j'$, $j \neq j''$, $j' \neq j''$ such that every vertex in U_j is connected to every vertex in $U_{j'}$ in Λ and every vertex in U_j is connected to every vertex in $U_{j''}$ in $\overline{\Lambda}$. By the argument in the previous paragraph, this contradicts the assumption that X is an odd hole or antihole. □

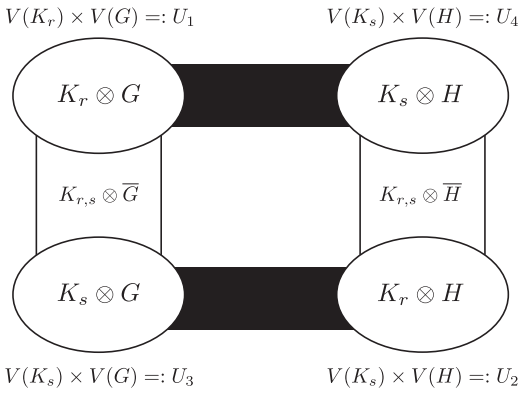

Figure 4.14: Structure of the graph $(K_r \uplus K_s) \nabla (G \uplus H)$ with the vertex set partition (U_1, U_2, U_3, U_4) defined in the proof of Lemma 4.27. Every vertex in U_1 is adjacent to every vertex in U_4 and vice versa. Every vertex in U_2 is adjacent to every vertex in U_3 and vice versa.

Corollary 4.7 *Suppose* $(K_r \uplus K_s) \nabla (G \uplus K_1)$ *and* $(K_r \uplus K_s) \nabla (H \uplus K_1)$ *are perfect. Then,* $(K_r \uplus K_s) \nabla (\biguplus_{i=1}^{k_1} G \uplus \biguplus_{i=1}^{k_2} H)$ *is perfect for any* $k_1, k_2 \in \mathbb{N}$.

Proof. The graph $(K_r \uplus K_s) \nabla E_3 \cong K_{r,s} \otimes K_3$ by the definition of the weak modular product, which is perfect by Proposition 4.7. Thus, we have that $(G \uplus E_2) \nabla (K_r \uplus K_s) \cong (G \uplus K_1 \uplus K_1) \nabla (K_r \uplus K_s)$ is perfect by Lemma 4.27, and the same argument applies for $(H \uplus K_1 \uplus K_1) \nabla (K_r \uplus K_s)$. Now inductively apply Lemma 4.27 with the above and the result follows. □

Lemma 4.28 *The graph* $(K_r \uplus K_s) \nabla (K_{1,m} \uplus K_1)$ *is perfect.*

Proof. For brevity we denote the graph $(K_r \uplus K_s) \nabla (K_{1,m} \uplus K_1)$ by G. Furthermore, we impose the vertex labelling of Figure 4.15. We show that G satisfies the definition of a perfect graph, namely that $\omega(X) = \chi(X)$ for all induced subgraphs X. First, observe that $\chi(G)$ is 3-colourable, according to the colouring in Figure 4.15. Now, let $z \in \{0,1\}$ and $\bar{z}$ be its binary complement and let X be an induced subgraph of

G. Suppose X includes a vertex from $U_z \times V_{1,2}$, a vertex from $U_z \times V_0$ and a vertex from $U_{\overline{z}} \times V_{1,1}$. Then X contains a triangle. From inspection of Figure 4.15 one sees that G is K_4-free, so $\omega(X) = 3$ in this case. Moreover, since G has a 3-colouring, $\omega(X) = \chi(X) = 3$. If X contains no three such vertices, we see from Figure 4.15 that X is bipartite and so $\omega(X) = \chi(X) = 2$. □

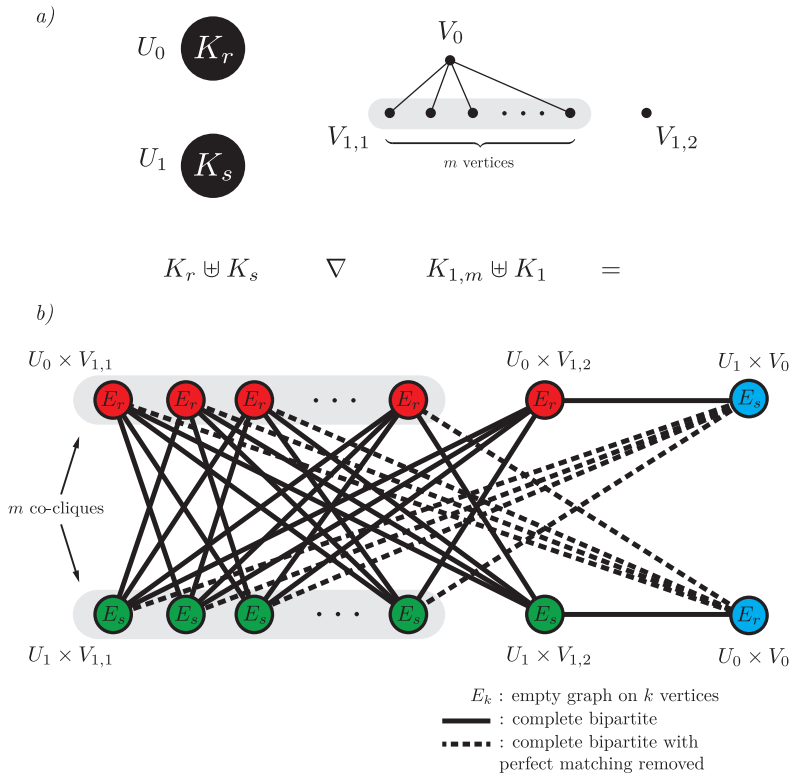

Figure 4.15: *a)* Illustration of the graphs $K_r \uplus K_s$ and $K_{1,m} \uplus K_1$ with vertex labellings indicated. Illustration of the graph $(K_r \uplus K_s) \nabla (K_{1,m} \uplus K_1)$, with vertex labelling and a 3-colouring indicated. Each node represents an empty graph on either r or s vertices. A full line represents a fully bipartite graph induced over the end nodes and a dashed line represents a complete bipartite graph with a perfect matching removed.

Proposition 4.8 *Let $G = K_r \uplus K_s$ and let H be a disjoint union of stars and cliques. Then, $G \nabla H$ is perfect.*

Proof. Follows immediately from Corollary 4.7, Lemma 4.28 and Lemma 4.25. □

Lemma 4.29 *The graph $P_4 \nabla K_{1,r}$ is perfect for any $r \geq 1$.*

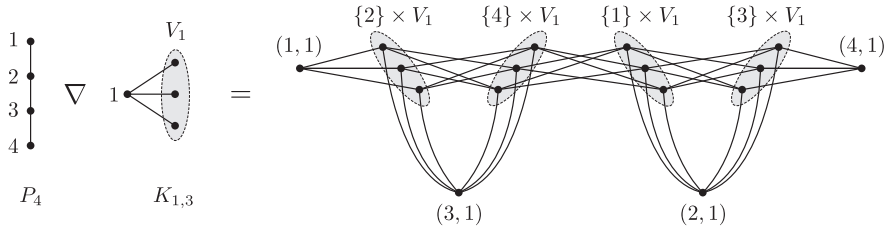

Figure 4.16: Illustration of the graph $P_4 \nabla K_{1,r}$, with $r = 3$ for concreteness. The sets $\{i\} \times V_1$ for $i \in \{1,2,3,4\}$ each have cardinality r. The subgraphs of $P_4 \nabla K_{1,r}$ induced on the sets $\{2,4\} \times V_1$, $\{1,3\} \times V_1$ and $\{1,4\} \times V_1$ are complete bipartite graphs with a perfect matching removed. The vertex $(1,1)$ is connected to all vertices in $\{2\} \times V_1$. The vertex $(4,1)$ is connected to all vertices in $\{3\} \times V_1$. The vertex $(3,1)$ is connected to all vertices in $\{2\} \times V_1$ and $\{4\} \times V_1$. The vertex $(2,1)$ is connected to all vertices in $\{1\} \times V_1$ and $\{3\} \times V_1$.

Proof. The case $r = 1$ is trivial by Corollary 4.4. Impose now the vertex labelling from Figure 4.16. We have drawn $P_4 \nabla K_{1,r}$ for $r = 3$ in Figure 4.16, the result readily generalises to any $r \geq 2$. We now use the strong perfect graph theorem. Observe that any induced subgraph of $P_4 \nabla K_{1,r}$ not including both vertices $(3,1)$ and $(2,1)$ is bipartite, so is perfect. Thus, any induced subgraph containing an odd hole or antihole must include either $(3,1)$ or $(2,1)$. Without loss of generality, consider an odd cycle $C = x_1, x_2, \ldots, x_{2k+2}$ (for $k \geq 2$) starting and beginning at vertex $(3,1)$, i.e. $x_1 = x_{2k+2} = (3,1)$. Clearly, either $x_2 \in \{2\} \times V_1$ and $x_{2k+1} \in \{4\} \times V_1$ or $x_2 \in \{4\} \times V_1$ and $x_{2k+1} \in \{2\} \times V_1$. Then, the vertices $\{x_1, x_2, x_{2k+1}\}$ induce a triangle and so $(P_4 \nabla K_{1,r})[C]$ is not an odd hole. Thus, $P_4 \nabla K_{1,r}$ is odd hole-free. Furthermore, observe that the only neighbour common to x_2 and x_{2k+1} is x_1. Thus, there is no diamond in $P_4 \nabla K_{1,r}$ with a degree-2 vertex at $(3,1)$. But every vertex in an odd antihole on 7 or more vertices is a degree-2 vertex in some diamond, so $(3,1)$ cannot be a vertex of an odd antihole, and so $P_4 \nabla K_{1,r}$ is odd antihole-free. □

Corollary 4.8 *The graph $P_4 \nabla (K_1 \uplus K_r)$ is perfect for any $r \geq 1$.*

Proof. Taking complements, Lemmas 4.29 and 4.12 give the result. □

4.4.2.3 Full case analysis

We have finally gathered the required ingredients to prove Theorem 4.1. The proof constitutes a case analysis over all pairs of finite, simple graphs, which has been split into Lemmas 4.30—4.42. They are tied up in the proof of Theorem 4.1.

In this subsection we let the binary variable $z \in \{0,1\}$ be arbitrary and $\overline{z}$ be its complement. We do this for brevity, so we can make statements such as "$G_z \nabla G_{\overline{z}}$ is

perfect if and only if G_z has property $\mathcal{P}_A$ and $G_{\overline{z}}$ has property $\mathcal{P}_B$, for any $z \in \{0,1\}$". The above statement is equivalent to the statements: "$G_0 \nabla G_1$ is perfect if and only if either: G_0 has property $\mathcal{P}_A$ and G_1 has property $\mathcal{P}_B$, or G_1 has property $\mathcal{P}_A$ and G_0 has property $\mathcal{P}_B$".

Without further ado, we proceed with the case analysis.

Lemma 4.30 *Suppose $G_z \cong K_r \uplus K_s$ for $r+s \geq 3$ and $G_{\overline{z}}$ is paw-free. Then $G_z \nabla G_{\overline{z}}$ is perfect if and only if either*

1. *$G_z \cong K_2 \uplus E_1$ and $G_{\overline{z}} \cong C_5$; or*
2. *$G_z \cong K_m \uplus K_1$ for $m \in \mathbb{N}$ and $G_{\overline{z}} \cong P_4$; or*
3. *$G_{\overline{z}} \in \{K_m, K_{m,n}\}$ for $m,n \in \mathbb{N}$; or*
4. *$G_{\overline{z}}$ is a disjoint union of cliques and stars with two or more connected components.*

Proof. $G_{\overline{z}}$ is paw-free, so by Lemma 4.7, it is a disjoint union of complete multipartite and triangle-free graphs. Moreover, by assumption G_z has an induced $K_2 \uplus E_1$.

First, suppose that $G_{\overline{z}}$ has a triangle-free component X. Moreover, suppose that X is bipartite and not complete. Then, by Lemma 4.5 X has an induced P_4. Lemma 4.17 states that the weak modular product of a bipartite augment of P_4 with $K_2 \uplus E_1$ is not perfect, so if $G_{\overline{z}} \ncong P_4$, $G_z \nabla G_{\overline{z}}$ is not perfect. Suppose $G_{\overline{z}} \cong P_4$. If $r = 1$ or $s = 1$, then $G_z \nabla G_{\overline{z}} \cong (K_1 \uplus K_n) \nabla P_4$ for some n and so is perfect by Corollary 4.8, giving case (2). If $r \geq 2$ and $s \geq 2$, then G_z has an induced $2K_2$ and $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.21.

Suppose now that X is nonbipartite, so has an odd hole by Lemma 4.2. Moreover, suppose the largest hole has length greater than 6. Then, X has an induced P_5 and by Lemma 4.19 $G_z \nabla G_{\overline{z}}$ is not perfect. Now suppose that the largest odd hole in X is a C_5. If $G_z \ncong K_2 \uplus E_1$ then G_z contains either $2K_2$ or K_3 by assumption. Thus, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.23. If $G_z \cong (K_2 \uplus E_1)$, by Corollary 4.5 and Lemma 4.15 $G_z \nabla G_{\overline{z}}$ is perfect if and only if $G_{\overline{z}} \cong C_5$, giving case (1).

Now suppose $G_{\overline{z}}$ is a disjoint union of complete multipartite graphs. If $G_{\overline{z}}$ has an induced diamond then $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.19. By Lemma 4.6, a complete multipartite graph is diamond-free if it is a disjoint union of cliques

and complete bipartites. Suppose $G_{\overline{z}}$ is diamond-free. If $G_{\overline{z}}$ is connected, then $G_{\overline{z}} \in \{K_m, K_{m,n}\}$ for $m, n \in \mathbb{N}$ and $G_z \nabla G_{\overline{z}}$ is perfect by Lemmas 4.25 and 4.26, giving case (3). Now suppose $G_{\overline{z}}$ has two or more connected components. If $G_{\overline{z}}$ has an induced $K_{2,2}$, then $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.19; else $G_{\overline{z}}$ is a disjoint union of stars and cliques, in which case $G_z \nabla G_{\overline{z}}$ is perfect by Proposition 4.8, giving case (4). □

Lemma 4.31 *Suppose G_z is a disjoint union of cliques with k connected components, where $k \in \mathbb{N} \setminus \{2\}$. Moreover, suppose $G_{\overline{z}}$ has an induced P_3. Then $G_z \nabla G_{\overline{z}}$ is perfect if and only if either*

1. *$G_z \in \{K_1, K_2\}$; or*
2. *$G_z \cong K_r$ for $r \in \mathbb{N}$ and $G_{\overline{z}}$ is (odd hole, paw)-free; or*
3. *$G_z \cong E_k$ and $G_{\overline{z}}$ is (odd antihole, co-paw)-free*

Proof. Suppose $k \geq 3$. If G_z is empty we get case (3) from Corollary 4.4 and Lemma 4.12. Assume G_z is nonempty. Then, G_z has an induced $K_2 \uplus E_2$ and $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.20. Now assume $k = 1$, i.e. $G_z \cong K_r$ for some $r \in \mathbb{N}$. Then we get cases (1) and (2) from Corollary 4.4. □

Lemma 4.32 *Suppose G_z is a disjoint union of complete multipartite graphs and triangle-free graphs. Moreover, suppose that $G_{\overline{z}}$ is connected, (P_4, cricket, dart, hourglass)-free and contains an induced paw. Then $G_z \nabla G_{\overline{z}}$ is perfect if and only if $G_z \cong K_{m,n}$.*

Proof. Suppose $G_{\bar{z}}$ has an induced $K_2 \uplus E_1$. Then, by Lemma 4.19 $G_z \nabla G_{\overline{z}}$ is not perfect. Now, assume G_z is $(K_2 \uplus E_1)$-free. By Corollary 4.2, a $(K_2 \uplus E_1)$-free graph is complete multipartite. Moreover, assume that G_z has an induced diamond. $G_z \nabla G_{\overline{z}}$ contains $Y \nabla K_{1,1,2}$, so is not perfect by Lemma 4.19 as the paw contains $K_2 \uplus E_1$ as an induced subgraph. We now assume G_z is diamond-free, so is a clique or complete bipartite by Lemma 4.6 and Corollary 4.2. If G_z is a clique, then from Corollary 4.4 $G_z \nabla G_{\overline{z}}$ is not perfect. If G_z is complete bipartite then $G_z \nabla G_{\overline{z}}$ is perfect, from taking complements and using Proposition 4.8 along with Lemmas 4.9 and 4.12. □

Lemma 4.33 *Suppose G_z is a disjoint union of complete multipartite graphs and triangle-free graphs, containing an induced P_4. Moreover, suppose that $G_{\overline{z}}$ is complete multipartite and contains an induced P_3. Then $G_z \nabla G_{\overline{z}}$ is perfect if and only if either*

1. *$G_z \cong C_5$, $G_{\overline{z}} \cong P_3$; or*
2. *$G_z \cong P_4$, $G_{\overline{z}} \in \{K_n, K_{1,n}\}$.*

Proof. We denote by X a component of G_z containing an induced P_4. Suppose X is bipartite. By Lemma 4.16, if $G_z \ncong P_4$ $G_z \nabla G_{\overline{z}}$ is not perfect. We thus assume $G_z \cong P_4$. If $G_{\overline{z}}$ has an induced diamond, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.21. In accordance with Lemma 4.6, we thus let $G_{\overline{z}}$ be a clique or complete bipartite. If $G_{\overline{z}}$ is a clique $G_z \nabla G_{\overline{z}}$ is perfect by Corollary 4.4. If $G_{\overline{z}}$ is complete bipartite it either contains $K_{2,2}$ or is a star. In the former case $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.21; in the latter case is perfect by Lemma 4.29. This gives us case (2).

We now assume that X is nonbipartite, so by Lemma 4.2 X contains an odd hole. If the largest odd hole in X has length greater than 6, then X has an induced P_5 and $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.20. We thus assume X contains C_5 as an induced subgraph. By Lemma 4.14, if $G_z \ncong C_5$, $G_z \nabla G_{\overline{z}}$ is not perfect. Now let $G_z \cong C_5$, which contains P_4 as an induced subgraph. Recall that $G_{\overline{z}}$ is complete multipartite. If $G_{\overline{z}}$ has an induced diamond, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.21. With regard to Lemma 4.6, we thus let $G_{\overline{z}}$ be complete bipartite since $G_{\overline{z}}$ being a clique contradicts Lemma 4.3. Hence, $G_{\overline{z}}$ either contains $K_{2,2}$ or is a star. In the former case $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.21. In the latter, if $G_{\overline{z}} \cong K_{1,r}$ for $r \geq 3$ then $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.23. $C_5 \nabla K_{1,2}$ is perfect by Lemma 4.24 and we have case (1). □

Lemma 4.34 *Suppose G_z is a disjoint union of complete multipartite graphs. Moreover, suppose that $G_{\overline{z}}$ is complete multipartite. Then $G_z \nabla G_{\overline{z}}$ is perfect if and only if either*

1. *$G_{\overline{z}} \cong K_n$*
2. *$G_z \cong K_r \uplus K_s$, $G_{\overline{z}} \cong K_{m,n}$; or*
3. *G_z and $G_{\overline{z}}$ are complete multipartite.*

Proof. If G_z is connected, $G_z \nabla G_{\overline{z}}$ is perfect by Corollary 4.6 and we get case (3). We now assume G_z has more than one connected component, so from Corollary 4.2 G_z has an induced $K_2 \uplus E_1$. If $G_{\overline{z}}$ has an induced diamond, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.19. Suppose now $G_{\overline{z}}$ is diamond-free. From Lemma 4.6, $G_{\overline{z}}$ is either a clique or complete bipartite. If $G_{\overline{z}}$ is a clique $G_z \nabla G_{\overline{z}}$ is perfect by Corollary 4.4, as G_z, being a disjoint union of complete multipartites, is (odd hole, paw)-free. This falls into case (1). If $G_{\overline{z}} \cong K_{1,1} \cong K_2$, $G \nabla G_{\overline{z}}$ is perfect by Corollary 4.4, belonging to case (1). We assume now that $G_{\overline{z}} \cong K_{m,n}$ with $m+n \geq 3$, whence $G_{\overline{z}}$ has an induced P_3. If G_z has an induced diamond then G_z has an induced $K_{1,1,2} \uplus E_1$, as it has more than one connected component, in which case $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.20 (observe that $G_{\overline{z}}$ has an induced P_3 by Lemma 4.3). Now assume G_z is a disjoint union of cliques and complete bipartites. If G_z contains an induced P_3, it contains $P_3 \uplus E_1$ and $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.20. Finally, we let G_z be a disjoint union of cliques. If $G_z \cong K_m \uplus K_n$, then $G_z \nabla G_{\overline{z}}$ is perfect by Lemma 4.26; else G_z has three or more connected components and $G_z \nabla G_{\overline{z}}$ is not perfect, since G_z contains $K_2 \uplus E_2$ as an induced subgraph and from Lemma 4.20, $P_3 \nabla (K_2 \uplus E_2)$ is not perfect. This gives us case (2) and completes the proof. □

Lemma 4.35 *Suppose G_z has an induced paw. Furthermore, suppose $G_{\overline{z}}$ is connected and (P_4, cricket, dart, hourglass)-free. Then $G_z \nabla G_{\overline{z}}$ is perfect if and only if either*

1. *$G_{\overline{z}} \cong K_{m,n}$, G_z is connected and (P_4, cricket, dart, hourglass)-free; or*
2. *$G_{\overline{z}} \in \{K_1, K_2\}$.*

Proof. First, suppose that $G_{\overline{z}}$ has an induced $K_2 \uplus E_1$, in which case $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.19. From Corollary 4.2, a connected $(K_2 \uplus E_1)$-free graph is complete multipartite. We thus assume $G_{\overline{z}}$ is complete multipartite. If $G_{\overline{z}} \cong K_n$, we get case (2) from Corollary 4.4. We then let $G_{\overline{z}} \ncong K_n$ for all $n \in \mathbb{N}$. Either $G_{\overline{z}}$ has an induced $K_{1,1,2}$, or is complete bipartite by Lemma 4.6. In the former case $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.19, using the fact that Y contains $K_2 \uplus E_1$ as an induced subgraph. In the latter case we have two scenarios: *i.* G_z is connected and (P_4, cricket, dart, hourglass)-free, in which case $G_z \nabla G_{\overline{z}}$ is perfect from Proposition 4.8 taken with Lemmas 4.12 and 4.9. This belongs to case (1). In

scenario *ii.* G_z is either disconnected or has an induced P_4, cricket, dart or hourglass. If G_z is disconnected it contains an induced $P_3 \uplus E_1$. Recall that $G_{\overline{z}}$ contains an induced P_3 by Lemma 4.3 and the assumption that $G_{\overline{z}} \ncong K_n$ for all $n \in \mathbb{N}$, so $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.20. If G_z contains any of $\{P_4, \text{cricket, dart, hourglass}\}$, then by Lemmas 4.21 and 4.18 $G_z \nabla G_{\overline{z}}$ is not perfect. This gives us case (1). □

Lemma 4.36 *Suppose G_z has an induced P_3 and $\overline{G_{\overline{z}}} \cong rK_2$. Then, $G_z \nabla G_{\overline{z}}$ is perfect if and only if*

1. *$\overline{G_z}$ is a disjoint union of cliques (equiv. G_z is complete multipartite); or*
2. *$\overline{G_{\overline{z}}} \cong K_2$ (equiv. $G_{\overline{z}} \cong E_2$); or*
3. *$\overline{G_z}$ is a disjoint union of stars and cliques and $\overline{G_{\overline{z}}} \cong 2K_2$ (equiv. G_z is connected and $(P_4, \text{cricket, dart, hourglass})$-free, $G_{\overline{z}} \cong K_{2,2}$).*
4. *$\overline{G_z} \cong K_{m,n}$ for $m, n \in \mathbb{N}$ and $\overline{G_{\overline{z}}} \cong 2K_2$ (equiv. $G_z \cong K_m \uplus K_n$ and $G_{\overline{z}} \cong K_{2,2}$).*

Proof. If $\overline{G_z}$ is a disjoint union of cliques, we have that G_z and $G_{\overline{z}}$ are both complete multipartite by Lemma 4.4. Corollary 4.6 gives us perfection of $G_z \nabla G_{\overline{z}}$, case (1). Now suppose $\overline{G_z}$ has an induced P_3. If $r \geq 3$, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.20 and Lemma 4.12. If $\overline{G_{\overline{z}}} \cong K_2$, we have case (2) from Corollary 4.4 and Lemma 4.12. Now suppose $\overline{G_{\overline{z}}} \cong 2K_2$. If we suppose that $\overline{G_z}$ is paw-free we have cases (3) and (4) from Lemma 4.30 and Lemma 4.12. If $\overline{G_{\overline{z}}}$ has an induced paw $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.20, as $G_{\overline{z}}$ has an induced $P_3 \uplus E_1$ (by Observation 4.3). □

Lemma 4.37 *Suppose G_z has an induced P_3 and is triangle-free. Moreover, suppose that $\overline{G_{\overline{z}}}$ is a disjoint union of stars with induced P_3. Then, $G_z \nabla G_{\overline{z}}$ is perfect if and only if*

1. *$G_z \cong C_5$, $\overline{G_{\overline{z}}} \cong P_3$ (equiv. $G_{\overline{z}} \cong K_2 \uplus E_1$); or*
2. *$G_z \cong K_{m,n}$ for $m, n \geq 2$; or*
3. *G_z is a disjoint union of stars with two or more connected components, $\overline{G_{\overline{z}}} \cong K_{1,r}$ (equiv. $G_{\overline{z}} \cong K_1 \uplus K_r$); or*
4. *$G_z \cong K_{1,r}$; or*
5. *$G_z \cong P_4$, $\overline{G_{\overline{z}}} \cong K_{1,r}$ (equiv. $G_{\overline{z}} \cong K_1 \uplus K_r$) for $r \geq 2$.*

Proof. First suppose that G_z is nonbipartite, so by Lemma 4.2 it has an odd hole. If the largest odd hole in G_z has size seven or larger, then G_z has an induced P_5. By Lemmas 4.3, 4.4 and Corollary 4.2, $G_{\overline{z}}$ has an induced P_3 if and only if $\overline{G_{\overline{z}}}$ is disconnected, as $G_{\overline{z}}$ is $(K_2 \uplus E_1)$-free only when disconnected. Thus, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.20 if $\overline{G_{\overline{z}}}$ is disconnected. Let $\overline{G_{\overline{z}}}$ be connected. Then, $G_{\overline{z}} \cong K_r \uplus K_1$ for some $r \geq 2$ and so $G_{\overline{z}}$ contains an induced $K_2 \uplus E_1$. In this case $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.17 as P_5 is a bipartite augment of P_4.

Now suppose G_z has an induced C_5. If $G_z \ncong C_5$, then $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.14. We now let $G_z \cong C_5$, in which case $\overline{G_z} \cong C_5$. If $\overline{G_{\overline{z}}} \ncong K_{1,2}$, then either it contains an induced $P_3 \uplus E_1$, or it contains an induced $K_{1,3}$. By Lemmas 4.20 and 4.23 respectively, $G_z \nabla G_{\overline{z}}$ is not perfect in both cases. If $\overline{G_{\overline{z}}} \cong K_{1,2}$, $G_{\overline{z}} \cong K_2 \uplus E_1$ and $G_z \nabla G_{\overline{z}}$ is perfect by Corollary 4.5, giving case (1).

Now we suppose G_z is bipartite. Suppose also that it has an induced P_4, so has a connected component that is not complete bipartite by Lemma 4.5. Moreover, $\overline{G_z}$ also has an induced P_4 since P_4 is self-complementary. Now assume that $\overline{G_{\overline{z}}}$ has two or more connected components, in which case it has an induced $P_3 \uplus E_1$ and thus $G_{\overline{z}}$ has an induced paw by Observation 4.3. Then, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.21. Now suppose that $\overline{G_{\overline{z}}}$ is connected, so $\overline{G_{\overline{z}}} \cong K_{1,r}$ and $G_{\overline{z}} \cong K_r \uplus E_1$ for $r \geq 2$. If $\overline{G_z} \ncong P_4$, by Lemmas 4.12 and 4.16 $G_z \nabla G_{\overline{z}}$ is not perfect. If $\overline{G_z} \cong P_4$, $G_z \cong P_4$ and $G_{\overline{z}} \cong K_r \uplus E_1$ for $r \geq 2$. Thus, $G_z \nabla G_{\overline{z}}$ is perfect by Corollary 4.8, giving case (5).

Now we suppose G_z is a disjoint union of complete bipartites. First suppose G_z has an induced $K_{2,2}$. If it has two or more connected components then G_z contains an induced $K_{2,2} \uplus E_1$ and $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.19. If G_z is connected, then $G_z \cong K_{m,n}$, $\overline{G_z} \cong K_m \uplus K_n$. By Proposition 4.8 and Lemma 4.12 $G_z \nabla G_{\overline{z}}$ is perfect, giving case (2). Now let G_z be a disjoint union of stars. Moreover, suppose G_z has two or more connected components so has induced $2K_2$; equivalently $\overline{G_z}$ has an induced $K_{2,2}$. Also, let $\overline{G_{\overline{z}}}$ have two or more connected components so has induced $P_3 \uplus E_1$. From Lemma 4.22 we have that $K_{2,2} \nabla (P_3 \uplus E_1)$ is not perfect, and so by Lemma 4.12 $G_z \nabla G_{\overline{z}}$ is not perfect. If $\overline{G_{\overline{z}}}$ is connected $G_{\overline{z}} \cong K_1 \uplus K_r$, so by Proposition 4.8 and Lemma 4.12 $G_z \nabla G_{\overline{z}}$ is perfect, giving case (3). Finally, suppose G_z is connected. Then, $G_z \cong K_{1,r}$, $\overline{G_z} \cong K_1 \uplus K_r$ and $G_z \nabla G_{\overline{z}}$ is perfect by

Proposition 4.8 and Lemma 4.12, giving case (4). This completes the proof. □

Lemma 4.38 *Let G be a $(K_2 \uplus E_2)$-free graph such that $\alpha(G) \geq 3$. Then $G \nabla P_3$ is perfect if and only if either: G is connected and $(P_4, \text{cricket, dart, hourglass})$-free; or $G \cong E_n$.*

Proof. ($\Rightarrow$) If $G \cong E_n$ then $G \nabla P_3$ is perfect, since $E_n \nabla P_3 \cong K_n \nabla (K_2 \uplus E_1)$ and the latter is perfect by Corollary 4.4. Moreover, one can see that if G is connected and (P_4,cricket, dart, hourglass)-free, $G \nabla H$ is perfect by taking complements and using Lemmas 4.12 and 4.9 with Proposition 4.8.

($\Leftarrow$) We prove the contrapositive, namely that $G \nabla P_3$ is not perfect when G is a $(K_2 \uplus E_2)$-free graph with $\alpha(G) \geq 3$ that is either disconnected, or contains an induced P_4, cricket, dart or hourglass.

Suppose G is disconnected and nonempty. Then, if G has more than three connected components, it has an induced $K_2 \uplus E_2$, a contradiction. So G must have two components. If both components of G are cliques then $\alpha(G) = 2$ and we have a contradiction, so at least one component of G contains a P_3 by Lemma 4.3. Now, since $(P_3 \uplus E_1) \nabla P_3$ is not perfect from Lemma 4.20, $G \nabla P_3$ is not perfect.

Now suppose G is connected. For $X \in \{\text{cricket, dart, hourglass}\}$, $X \nabla P_3$ is not perfect by Lemma 4.18, so for any G containing induced X, $G \nabla P_3$ is not perfect. Finally, suppose that G contains an induced P_4. Since P_4 is self complementary, by assumption we have that $\overline{G}$ is diamond-free, has an induced P_4 and $\omega(\overline{G}) \geq 3$. We shall now show that $\overline{G} \nabla (K_2 \uplus E_1)$ is not perfect, and the result follows from Lemma 4.12. To wit, suppose $\overline{G}$ contains an induced paw. Then, we have that $\overline{G} \nabla (K_2 \uplus E_1)$ is not perfect by Lemma 4.19. Now suppose $\overline{G}$ is paw-free. From Lemma 4.7 $\overline{G}$ is either *i.* triangle-free or *ii.* is complete multipartite. Case *i.* contradicts the assumption that $\omega(\overline{G}) \geq 3$. Case *ii.* taken with the assumption that $\overline{G}$ is diamond-free implies that $\overline{G}$ is complete bipartite, by Lemma 4.6. But $\overline{G}$ contains an induced P_4, contradicting Lemma 4.5. Thus $\overline{G} \nabla (K_2 \uplus E_1)$ is not perfect when G is connected, contains an induced P_4 and is $(K_2 \uplus E_2)$-free with $\alpha(G) \geq 3$. The result follows. □

Lemma 4.39 *Suppose G_z has an induced P_3 and contains a triangle. Moreover, suppose that $\overline{G_{\overline{z}}}$ is a disjoint union of stars with induced P_3. Then, $G_z \nabla G_{\overline{z}}$ is perfect*

if and only if $\overline{G_z}$ is connected and (P_4, cricket, dart, hourglass)-free and $\overline{G_{\overline{z}}} \cong K_{1,r}$ (equiv. G_z is a disjoint union of stars and cliques and $G_{\overline{z}} \cong K_1 \uplus K_r$).

Proof. Suppose $\overline{G_z}$ has an induced $K_2 \uplus E_2$. Then, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemmas 4.12 and 4.20. Now suppose that $\overline{G_z}$ is $(K_2 \uplus E_2)$-free. By Lemma 4.12 and Lemma 4.38 $G_z \nabla G_{\overline{z}}$ is not perfect if $\overline{G_z}$ is disconnected or contains an induced P_4, cricket, dart or hourglass. We thus assume that $\overline{G_z}$ is connected and (P_4, cricket, dart, hourglass)-free. Now if $\overline{G_{\overline{z}}}$ is connected we have that $G_{\overline{z}} \cong K_1 \uplus K_r$ and G_z is a disjoint union of cliques and stars by Lemma 4.9, and so by Proposition 4.8 $G_z \nabla G_{\overline{z}}$ is perfect.

Now assume $\overline{G_{\overline{z}}}$ is disconnected. If $\overline{G_z} \cong K_n$, then $G_z \cong E_n$ contradicting the assumptions of the lemma. Finally, suppose $\overline{G_z} \ncong K_n$, in which case $\overline{G_z}$ has an induced P_3 by Lemma 4.3. From assumptions, $\overline{G_{\overline{z}}}$ has an induced $P_3 \uplus E_1$; Lemmas 4.12 and 4.20 give that $G_z \nabla G_{\overline{z}}$ is not perfect. □

Lemma 4.40 *Suppose G_z has an induced P_3 and $\overline{G_{\overline{z}}}$ is a disjoint union of complete bipartites, containing an induced $K_{2,2}$. Then $G_z \nabla G_{\overline{z}}$ is perfect if and only if $\overline{G_z}$ is connected and (P_4, cricket, dart, hourglass)-free and $\overline{G_{\overline{z}}} \cong K_{m,n}$ (equiv. G_z is a disjoint union of stars and cliques, $G_{\overline{z}} \cong K_m \uplus K_n$).*

Proof. Observe that $\overline{G_z}$ has an induced $K_2 \uplus E_1$. Now suppose $\overline{G_{\overline{z}}}$ has two or more connected components. Then, $\overline{G_{\overline{z}}}$ has an induced $K_{2,2} \uplus E_1$. Thus, by Lemmas 4.19 and 4.12 $G_z \nabla G_{\overline{z}}$ is not perfect. Now suppose $\overline{G_{\overline{z}}}$ is connected, i.e. $\overline{G_{\overline{z}}} \cong K_{m,n}$. Moreover, suppose that $\overline{G_z}$ is not connected. Either $\overline{G_z}$ is a disjoint union of cliques or has an induced $P_3 \uplus E_1$, by Lemma 4.3. In the former case, we contradict the assumption that G_z has an induced P_3. In the latter case Lemmas 4.20 and 4.12 give that $G_z \nabla G_{\overline{z}}$ is not perfect.

Now suppose that $\overline{G_z}$ is connected. If $\overline{G_z}$ contains an induced P_4, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemmas 4.21 and 4.12. Let $\overline{G_z}$ be P_4-free. If $\overline{G_z}$ contains an induced cricket, dart, or hourglass then by Lemmas 4.18 and 4.12 $G_z \nabla G_{\overline{z}}$ is not perfect, since $K_{2,2}$ contains P_3. Otherwise, Lemma 4.12 and Proposition 4.8 give that $G_z \nabla G_{\overline{z}}$ is perfect. □

Lemma 4.41 *Suppose G_z has an induced P_3 and $\overline{G_{\overline{z}}}$ is bipartite with induced P_4. Then $G_z \nabla G_{\overline{z}}$ is perfect if and only if $\overline{G_{\overline{z}}} \cong P_4$ and $\overline{G_z} \in \{C_5, P_4, K_1 \uplus K_s\}$ (equiv. $G_{\overline{z}} \cong P_4$ and $G_z \in \{C_5, P_4, K_{1,s}\}$).*

Proof. Observe that $\overline{G_z}$ has an induced $K_2 \uplus E_1$. Let us suppose that $\overline{G_{\overline{z}}} \ncong P_4$. Then, by Lemmas 4.17 and 4.12 $G_z \nabla G_{\overline{z}}$ is not perfect. Thus, we suppose that $\overline{G_{\overline{z}}} \cong P_4$. Suppose that $\overline{G_z}$ has an induced paw. Then, by Lemmas 4.21 and 4.12 $G_z \nabla G_{\overline{z}}$ is not perfect. Now let $\overline{G_z}$ be paw-free, that is, by Lemma 4.7 $\overline{G_z}$ is a disjoint union of complete multipartite and triangle-free graphs.

First, assume that $\overline{G_z}$ has triangle-free component X that is not complete bipartite. If X is bipartite, $G_z \nabla G_{\overline{z}}$ is perfect if and only if $\overline{G_z} \cong P_4$, from Lemmas 4.16 and 4.12. Now suppose X is nonbipartite. By Lemma 4.2 it contains an odd hole. If the largest odd hole in X has seven or more vertices, X contains an induced P_5 and so by Lemmas 4.20 and 4.12 $G_z \nabla G_{\overline{z}}$ is not perfect. Now suppose the largest odd hole in X is a C_5. By Lemmas 4.14, 4.24 and 4.12 $G_z \nabla G_{\overline{z}}$ is perfect if and only if $\overline{G_z} \cong C_5$.

Now assume $\overline{G_z}$ is a disjoint union of complete multipartites and $\overline{G_z}$ is disconnected, for otherwise; by Lemma 4.2 it is $(K_2 \uplus E_1)$-free. However, $\overline{G_z}$ has an induced $K_2 \uplus E_1$ by assumption. If $\overline{G_z}$ has three or more components, it contains an induced $K_2 \uplus E_2$, and so by Lemmas 4.20 and 4.12 $G_z \nabla G_{\overline{z}}$ is not perfect. We thus assume $\overline{G_z}$ has two connected components. By Lemma 4.6 $\overline{G_z}$ has an induced diamond or is a disjoint union of cliques and complete bipartite graphs. In the former case, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemmas 4.21 and 4.12. We now assume the latter. If $\overline{G_z}$ contains $K_{2,2}$ as an induced subgraph, $G_z \nabla G_{\overline{z}}$ is not perfect by Lemmas 4.21 and 4.12. Moreover, if $\overline{G_z}$ contains a star (that is not also a clique) then it contains P_3 and thus $P_3 \uplus E_1$, since it has two components. Lemmas 4.20 and 4.12 give us that $G_z \nabla G_{\overline{z}}$ is not perfect in this case. Assume $\overline{G_z}$ is a disjoint union of cliques. If $\overline{G_z}$ contains $2K_2$, $G_z \nabla G_{\overline{z}}$ is not perfect from Lemmas 4.21 and 4.12. This leaves $\overline{G_z} \cong K_1 \uplus K_s$, in which case $G_z \nabla G_{\overline{z}}$ is perfect from Corollary 4.8. □

Lemma 4.42 *Suppose G_z has an induced P_3 and $\overline{G_{\overline{z}}}$ is nonbipartite and triangle-free. Then, $G_z \nabla G_{\overline{z}}$ is perfect if and only if $\overline{G_{\overline{z}}} \cong C_5$ and $\overline{G_z} \in \{K_2 \uplus E_1, P_4, C_5\}$ (equiv. $G_{\overline{z}} \cong C_5$ and $G_z \in \{P_3, P_4, C_5\}$).*

Proof. Note that $\overline{G_z}$ has an induced $K_2 \uplus E_1$. By Lemma 4.2, $\overline{G_{\overline{z}}}$ has an odd hole. If the largest odd hole has seven or more vertices, $\overline{G_{\overline{z}}}$ contains an induced P_5. Then, by Lemmas 4.19 and 4.12 $G_z \nabla G_{\overline{z}}$ is not perfect. Now suppose $\overline{G_{\overline{z}}}$ contains an induced C_5. By Lemmas 4.15 and 4.12, $G_z \nabla G_{\overline{z}}$ is not perfect if $\overline{G_{\overline{z}}} \ncong C_5$. So we assume

$\overline{G_{\overline{z}}} \cong C_5$, in which case $G_{\overline{z}} \cong C_5$. Then, Lemmas 4.24 and 4.12 give the result. □

Theorem 4.1 (Restated for convenience) *The graph $G = G_0 \nabla G_1$ is perfect if and only if one of the following holds:*

1. $G_z \in \{K_1, K_2, E_2\}$, $G_{\overline{z}}$ *arbitrary;*
2. $G_z \cong P_4$, $G_{\overline{z}} \in \{K_{1,r}, K_r \uplus K_1, P_4\}$;
3. $G_z \cong C_5$, $G_{\overline{z}} \in \{P_3, K_2 \uplus E_1, P_4, C_5\}$;
4. $G_z \cong K_r \uplus K_s$, $G_{\overline{z}}$ *is a disjoint union of stars and cliques;*
5. $G_z \cong K_{m,n}$, $G_{\overline{z}}$ *is connected and* $(P_4, \textit{cricket}, \textit{dart}, \textit{hourglass})$*-free;*
6. $G_z \cong K_n$, $G_{\overline{z}}$ *(odd hole, paw)-free;*
7. $G_z \cong E_n$, $G_{\overline{z}}$ *(odd antihole, co-paw)-free;*
8. G_z, $G_{\overline{z}}$ *are complete multipartite;*
9. G_z, $G_{\overline{z}}$ *are disjoint unions of cliques;*
10. $G_z \cong K_r \uplus K_s$, $G_{\overline{z}} \cong K_{m,n}$;

for any m, n, r, s, z, where $m, n, r, s \in \mathbb{N}$, and $z \in \{0, 1\}$, with its (Boolean) negation denoted by $\overline{z}$.

Proof. ($\Rightarrow$) We prove the forward direction for each case in turn: Case (1) follows directly from Corollary 4.4 and Lemma 4.12. For case (2), perfection of $P_4 \nabla P_4$ follows as a direct corollary of Lemma 4.24, as C_5 contains P_4 as an induced subgraph. $P_4 \nabla K_{1,r}$ and $P_4 \nabla (K_1 \uplus K_r)$ are perfect by Lemma 4.29 and Corollary 4.8 respectively. Case (3) follows directly from Lemma 4.24 and Corollary 4.5. Proposition 4.8 gives case (4). Proposition 4.8 taken with Lemmas 4.12 and 4.9 give case (5). Corollary 4.4 yields (6). For case (7), combine Corollary 4.4 and Lemma 4.12. Corollary 4.6 gives (8). Lemma 4.25 yields (9). Finally, (10) is proven by Lemma 4.26.

($\Leftarrow$) We show that the weak modular product of any pair of graphs not falling into cases (1)–(10) is not perfect. First suppose G_z is P_3-free. Then, by Lemma 4.3 G_z is a disjoint union of cliques. If $G_{\overline{z}}$ is a disjoint union of cliques then we have case (9). Thus we suppose $G_{\overline{z}}$ is not a disjoint union of cliques, so has an induced P_3 by

Lemma 4.3. If G_z has k connected components, where $k \in \mathbb{N} \setminus \{2\}$, by Lemma 4.31, the only cases where $G_z \nabla G_{\overline{z}}$ is perfect belong to cases (1), (6) and (7). If $k = 2$, either $G_z \cong E_2$, and we have case (1), or G_z is nonempty and contains an induced $K_2 \uplus E_1$. If $G_{\overline{z}}$ has an induced paw, then $G_z \nabla G_{\overline{z}}$ is not perfect by Lemma 4.19. If $G_{\overline{z}}$ is paw-free then by Lemma 4.30 the only pairs which give a perfect product fall under cases (3), (2), (10) and (4). This completes the proof for the case when G_z is P_3-free.

Now, suppose G_z contains an induced P_3. If $G_{\overline{z}}$ is empty we have a perfect product only under the conditions of case (7) from Lemma 4.12 and Corollary 4.4, so assume $G_{\overline{z}}$ is nonempty. Furthermore, suppose $\alpha(G_{\overline{z}}) \geq 3$. By Lemma 4.20, $G_z \nabla G_{\overline{z}}$ is not perfect if $G_{\overline{z}}$ has an induced $K_2 \uplus E_2$, as G_z contains an induced P_3. We thus consider the case when $G_{\overline{z}}$ is $(K_2 \uplus E_2)$-free. By Lemma 4.38, $G_z \nabla G_{\overline{z}}$ is not perfect if $G_{\overline{z}}$ is disconnected or contains an induced P_4, cricket, dart, or hourglass. So we assume $G_{\overline{z}}$ is connected and (P_4, cricket, dart, hourglass)-free. Now, if G_z has an induced paw, by Lemma 4.35 the only cases when $G_z \nabla G_{\overline{z}}$ is perfect belong to cases (5) and (1). We thus assume G_z is paw free, so is a disjoint union of complete multipartite and triangle-free graphs by Lemma 4.7. If $G_{\overline{z}} \cong K_n$, it is P_3-free by Lemma 4.3. Then, we have a perfect product if and only if the conditions of case (6) are satisfied by Corollary 4.4. Suppose $G_{\overline{z}}$ has induced P_3. Furthermore, if $G_{\overline{z}}$ has an induced paw, by Lemma 4.32 $G_z \nabla G_{\overline{z}}$ is perfect only in case (5). Now let $G_{\overline{z}}$ be paw-free, so by Lemma 4.8 is complete multipartite as it is also P_4-free. We now suppose G_z has a triangle-free component X that is not complete bipartite, and so has an induced P_4 by Lemmas 4.5 and 4.2. We then have $G_z \nabla G_{\overline{z}}$ is perfect only in cases (3), (1) and (2) from Lemma 4.33. Assume G_z is P_4-free, so by Lemma 4.8 is a disjoint union of complete multipartites. Then, from Lemma 4.34 $G_z \nabla G_{\overline{z}}$ is perfect only in cases (6), (10) and (8). This covers the case when G_z contains an induced P_3 and $G_{\overline{z}}$ is nonempty with $\alpha(G_{\overline{z}}) \geq 3$.

We now consider the case when G_z contains an induced P_3 and $G_{\overline{z}}$ is nonempty with $\alpha(G_{\overline{z}}) \leq 2$. If $\alpha(G_{\overline{z}}) = 1$, $G_{\overline{z}} \cong K_n$ and by Corollary 4.4 is perfect if and only if G_z is (odd hole, paw)-free, falling into case (6). We are left with $\alpha(G_{\overline{z}}) = 2$ or, equivalently, $\omega(\overline{G_{\overline{z}}}) = 2$. Since $\omega(\overline{G_{\overline{z}}}) = 2$, $\overline{G_{\overline{z}}}$ is triangle-free. Let us first consider the case when $\overline{G_{\overline{z}}}$ is nonbipartite. Then, by Lemma 4.42 $G_z \nabla G_{\overline{z}}$ is only perfect in

case (3). Now suppose $\overline{G_{\overline{z}}}$ is bipartite. If $\overline{G_{\overline{z}}}$ contains an induced P_4, by Lemma 4.41 $G_z \nabla G_{\overline{z}}$ is only perfect in cases (2) and (3). Thus we suppose $\overline{G_{\overline{z}}}$ is a disjoint union of complete bipartite graphs, as from Lemma 4.5 these are the only bipartite graphs that are P_4-free. Now suppose $\overline{G_{\overline{z}}}$ has an induced $K_{2,2}$. Then, by Lemma 4.40, $G_z \nabla G_{\overline{z}}$ is only perfect in case (4). We thus assume that $\overline{G_{\overline{z}}}$ is a disjoint union of stars. If $\overline{G_{\overline{z}}} \cong rK_2$, by Lemma 4.36, $G_z \nabla G_{\overline{z}}$ is only perfect in cases (9), (1), (5) and (10). Now assume that $\overline{G_{\overline{z}}}$ is a disjoint union of stars with an induced P_3. Since G_z has an induced P_3, $\overline{G_z}$ is nonempty. We then consider three scenarios: *i.* $\alpha(\overline{G_z}) = 1$, *ii.* $\alpha(\overline{G_z}) = 2$ and *iii.* $\alpha(\overline{G_z}) \geq 3$. In scenario *i.*, $G_z \cong K_n$, yet G_z has an induced P_3 by assumption, contradicting Lemma 4.3. In scenario *ii.*, $\omega(G_z) = 2$ and so G_z is triangle-free. Lemma 4.37 gives us that $G_z \nabla G_{\overline{z}}$ is only perfect in cases (3), (5), (4) and (2). Finally, in scenario *iii.*, $\omega(G_z) \geq 3$ so G_z contains a triangle. From Lemma 4.39, $G_z \nabla G_{\overline{z}}$ is only perfect in case (4).

This completes the proof as we have enumerated all pairs of graphs. □

4.5 Discussion

Theorem 4.1 gives us a characterisation of all pairs of graphs whose weak modular product is perfect. This induces an algorithm for GraphIsomorphism on these graphs: compute the Lovász number of the product graph via an SDP solver. If the computed $\vartheta > n - \delta$ for some approximation error $\delta \in (0, 0.5)$ then the graphs are isomorphic; otherwise not. In light of the discussion of Section 4.3, it is natural to compute a polynomial time upper bound on the runtime of this induced GraphIsomorphism algorithm and to ask if any of the cases (1)–(10) (as defined in Theorem 4.1) fall into classes of graphs for which there is no existing efficient graph isomorphism algorithm.

The current state of the art runtime for computing the approximate value of a semidefinite program [LSW15] is

$$\widetilde{O}\left(n'\left((n')^2 + (m')^\omega + s\right)\right), \tag{4.5}$$

where n' is the number of variables, m' the number of constraints, s the sparsity, $\omega \in [2, 2.373)$ the (still unknown) optimal exponent for matrix multiplication and $\widetilde{O}$ denotes that we ignore polylogarithmic factors. Inspection of the Lovász-

ϑ SDP (4.1) for a graph G gives that $n' = \frac{1}{2}(|V(G)|(|V(G)|-1))$, $m' = \left|E(\overline{G})\right| + 1$ and $s = \max_{v \in V(G)} |N(v)|$. The worst-case values for these quantities, assuming G has n vertices, are $n' = O(n^2)$, $m' = O(n^2)$ and $s = O(n)$ respectively. Recall that the algorithm induced by Theorem 4.1 computes $\vartheta(G \nabla H)$, where G and H have n vertices. This results in an overall worst-case runtime complexity of $\widetilde{O}\left(n^{4(1+\omega)}\right)$.

We analyse each case of Theorem 4.1 in turn with respect to existing algorithms in the literature, assuming $|V(G)| = |V(H)| = n$. Cases (1)–(3) admit trivial constant-time algorithms to check isomorphism. In case (4), there is a simple algorithm: find the connected components, in the case when the disjoint union of stars and cliques has two connected components, count the neighbours of each vertex and compare; otherwise the graphs are trivially non-isomorphic. It is well-known that computing connected components has time complexity $O(|V(G)| + |E(G)|)$ — a worst case bound here is $O(n^2)$. In case (5), one can use the previous algorithm after taking complements, or, observe that the two graphs in question are cographs[3] and so admit an efficient GraphIsomorphism algorithm [CLB81], again with $O(|V(G)| + |E(G)|) = O(n^2)$ runtime complexity. Cases (6) and (7) admit trivial algorithms by counting vertex neighbours, at $O(n^2)$ cost in runtime. Cases (8) and (9) are cographs so have an efficient algorithm, as mentioned earlier. Case (10) is trivial by counting connected components, with $O(n^2)$ runtime complexity.

Thus, the technique developed in this chapter does not lead to an algorithm for GraphIsomorphism on any new graph families, nor does it lead to an improved runtime-complexity in these cases, since all cases have an algorithm running in time $O(n^2)$, as compared with the bound of $\widetilde{O}\left(n^{4(1+\omega)}\right)$ for the approach deriving from Theorem 4.1.

[3] A cograph is a graph for which every connected induced subgraph has diameter at most 2.

Chapter 5

Conclusion

As we approach the end of the thesis, we make some concluding remarks. We briefly summarise the contributions of the thesis in Section 5.1. In Section 5.2 we present a critical assessment of the thesis work. Section 5.3 lays out future work to be undertaken. For convenience we restate the research questions posed in Chapter 1.

Research Question 1 (Lifted Markov chains and quantum walks) *Which computational resources are required for a classical random walk to replicate the mixing dynamics of a quantum walk?*

Research Question 2 (Graph products and isomorphism) *Can we use easy instances of* NP*-hard problems to make progress on the* NP*-intermediate problem* GraphIsomorphism*, which has thus far eluded a polynomial-time algorithm?*

5.1 Summary of contributions

In response to Research Question 1 we have seen the following in Chapter 3. For a graph G on n vertices

- We present a lifted Markov chain whose marginal mixes to any target distribution π over $V(G)$ that has all positive elements in $D(G)$ timesteps, which is optimal. This builds on prior work, which we extend to a rigorous proof of correctness and more algorithmic focus.
- We show that this lifted Markov chain has $O(n^2 D(G))$ states and requires $O(n^4 D(G))$ time to compute the transition probabilities.
- We show that for any quantum walk on a connected graph, the average mixing distribution has all positive elements. Thus, the lifted Markov chain described

above can sample from this distribution.

In Chapter 4 we address Research Question 2:

- Following Kozen [Koz78], we formulate GraphIsomorphism as an instance of an NP-hard problem: finding the clique number of the weak modular product of the candidate graphs. We observe that for perfect graphs, finding the clique number is polynomial-time.

- Combining the previous two observations, we have that GraphIsomorphism is efficiently solvable for graphs with a perfect weak modular product. This leads us to an algorithm for GraphIsomorphism. We enumerate all pairs of graphs for which the weak modular product is perfect and non-perfect. This tells us when the algorithm is efficient.

- We compare the algorithm with other GraphIsomorphism algorithms in the literature, finding that the proposed algorithm covers only cases that already admit efficient algorithms.

5.2 Critical assessment

We shall now analyse how well the work described in Chapters 3 and 4 addresses Research Questions 1 and 2.

5.2.1 Lifted Markov chains

Comparing Research Question 1 to the thesis contributions listed in 5.1 we see that we have a partial answer to the question, since we have a lifted Markov chain that allows one to sample from the average mixing distribution of a quantum walk. The result is rigorously proven with explicit bounds on the computational resources required to compute the lifting. These bounds grow modestly with the graph size. We make some further comments on the result:

- The proposed scheme allows one to very quickly sample from the quantum average mixing distribution. If the lifted Markov chain is sampled from midway through its evolution, the sampled distribution has no relation to the underlying quantum dynamics in following sense: suppose one were to sample from the lifted Markov chain at time $t \in \{0, \ldots, D(G)\}$. The probability of

obtaining a vertex v is not related to either of the probabilities[1] $Q_t(v|\psi(0))$ or $\overline{Q}_t(v|\psi(0))$; the first being the probability of measuring the vertex v after running the quantum walk for t timesteps having started in state $|\psi(0)\rangle$ and the second being the Cesàro average of this probability. The dynamics of the lifted chain *can* be matched to the quantum walk in this way at the price of the mixing time to the marginal being lower bounded by the quantum average mixing time [AST18].

- The lifted Markov chain we construct is necessarily non-reversible to achieve diameter-time mixing. Non-reversible Markov chains are often seen as less representative of "natural" stochastic dynamics than reversible chains. One wonders, in light of the lower bounds on mixing times in Table 2.2, how would demanding a reversible chain affect the results?

- On the other hand, if we wish to leave to the domain of lifted Markov chains, the construction of the stochastic bridge schedule in Lemma 3.1 suggests a randomised algorithm for sampling from the quantum average mixing distribution with potentially smaller computational overhead than using the d-lifting.

5.2.2 Graph products and isomorphism

We now examine the contributions listed in Section 5.1 in relation to Research Question 2. In terms of answering the question itself, we have acquired some evidence supporting a negative answer. We have associated GraphIsomorphism to instances of an NP-hard problem and found the corresponding "easy" instances of GraphIsomorphism. It was not expected that this would lead to a isomorphism algorithm for all graphs, due to the problem's long history of being unsolved. It was hoped that new families of graphs would be discovered for which GraphIsomorphism is polynomial. This unfortunately did not come to pass.

The proof of Theorem 4.1 sheds a lot of light onto the weak modular product. This graph product has not been well-studied, for reasons discussed in Section 4.2. Using modern tools, such as the `sage` computational package [SAGE], the ISGCI graph classes database [ISGCI] and the strong perfect graph theorem [CRST06] we have been able to understand when this product is perfect and when it is not. To

[1] defined in Section 2.4.

quote Paul Seymour [Sey06]: *"Perfect graphs have come to be recognised as having a natural place in the world"*. As such it is an important question for graph theorists is to know whether a given family of graphs is perfect or not.

5.3 Future research

This work raises a number of open questions, a few of which we list here.

5.3.1 Lifted Markov chains

Better bounds. Can we find tighter upper bounds, or even lower bounds on the preparation time for the d-lifting? Can we find a smaller footprint lifting that mixes to the marginal in diameter-time, with arbitrary target distribution?

Liftings and hitting times. Having examined the relation of lifting to the quantum mixing time in detail, what about the *quantum hitting time*? This is the expected number of timesteps it takes for a quantum walk starting at a given vertex to hit another 'target' vertex. Indeed, most of the useful quantum walks based algorithms rely on the superiority of the quantum hitting time to the classical equivalent. I believe that in this case it is impossible for a lifting to convey anything better than a constant speedup relative to a classical walk, which would give a separation between quantum walks and lifting in this scenario (in favour of quantum walks). It would be good to investigate this further.

Non-diffusive behaviour. Liftings have been introduced to mimic diffusive behaviour, that is, mixing of quantum walks. It would be interesting to see if non-diffusive, more "quantum" behaviours could be replicated with lifted Markov chains. Can one use liftings to engineer perfect state transfer [Bos03] for instance?

Optimised quantum walks. Can we find a quantum walk that mixes in diameter-time to some useful distribution? Given the same resources, how do lifted Markov chains compare with quantum walks for mixing? Concretely, suppose the register for a lifted Markov chain is the same size as a quantum coin register, how do they compare in general and specific cases?

Specific walks. The construction presented here uses many nodes to mimic quan-

tum mixing on an arbitrary connected graph. Can we specialise to quantum walks with known mixing times? Initial work to this end has been conducted in [AST18], where a lifted Markov chain on a d-dimensional lattice is found to have the same mixing time as the equivalent quantum walk.

5.3.2 Graph products and isomorphism

Other easy instances of the clique problem. We have shown that if computing the maximum clique for a family of graphs Γ is polynomial in the number of vertices then we have an efficient algorithm for GraphIsomorphism. We have done this for the case where Γ is the set of perfect graphs. For instance, it is known that finding maximum cliques in planar graphs is polynomial [PY81]. It might be useful to investigate planar perfect weak modular products, amongst other families Γ. We observe that the clique problem is fixed parameter intractable [CHKX06], that is, there is no parameter when fixed that makes the problem polynomial time. It has also been shown that this problem is hard to approximate [Hås99].

More reductions from GraphIsomorphism. In this thesis, we have examined one particular reduction to the clique problem. There are many other NP-hard problems; indeed since GraphIsomorphism $\in$ NP there is a reduction from isomorphism to any of these problems. One can then examine easy cases as we have done for the clique problem.

Unclassified perfect product graphs. There are a large number of different graph products [HIK11], only a small number of which have their perfect examples enumerated. It would be useful to complete this classification. Perhaps some of these long-ignored products would be found to have some other utility as a result of their being perfect.

Appendix A

Proofs for the sake of completeness

In this appendix we provide proofs that are helpful to the main text, but disrupt the flow.

A.1 Markov chains and stochastic matrices

Claim A.1 (Claim 2.1 in main text) *Let Ω be a finite sample space and suppose $P \in \mathrm{L}(\mathbb{R}^{|\Omega|})$ is a linear map. Then P maps every distribution $\pi \in \Delta(\Omega)$ to another distribution $\pi' \in \Delta(\Omega)$, that is, $\pi' = \pi P$, only if $P \in \mathrm{Stoch}(\mathbb{R}^{|\Omega|})$. Moreover, let $\pi'' = \pi P'$ for $P' \in \mathrm{Stoch}(\mathbb{R}^{|\Omega|})$ and $\pi \in \Delta(\Omega)$. Then, $\pi'' \in \Delta(\Omega)$.*

Proof. Suppose for the sake of contradiction that there exist $x', y' \in \Omega$ such that $P[x, y] < 0$. Now, consider the probability distribution $\pi[y] = \delta_y^{y'}$. Let $\pi' = \pi P$. Then $\pi'[y'] < 0$. This contradicts our assumption that $\pi' \in \Delta(\Omega)$, so $P[x, y] \geq 0$ for all $x, y \in \Omega$. Now, let $\pi \in \Delta(\Omega)$ and $\pi' := \pi P$. From assumptions, $\sum_{y\in\Omega} \pi'[y] = 1$. Thus $\sum_{y\in\Omega} \sum_{x\in\Omega} \pi[x]P[x, y] = 1$. Now, let $\pi[x] = \delta_x^{x'}$ for some $x' \in \Omega$. Then, $\sum_{y\in\Omega} P[x, y] = 1$. Since we can make x' arbitrary this implies that $\sum_{y\in\Omega} P[x, y] = 1$ for all $x \in \Omega$ and so $P \in \mathrm{Stoch}(\mathbb{R}^{|\Omega|})$.

For the second claim, the y^{th} element of $\pi P' = \pi''$ for $y \in \Omega$, $\pi''[y] = \sum_{x\in\Omega} \pi[x]P'[x, y]$. Now, $\pi''[y] \geq 0$, since $\pi[x] \geq 0$ and $P'[x, y] \geq 0$ for all $x, y \in \Omega$. Moreover, $\sum_{y\in\Omega} \pi''[y] = 1$, since

$$\sum_{y\in\Omega} \pi''[y] = \sum_{y\in\Omega} \sum_{x\in\Omega} \pi[x]P'[x, y] = \sum_{x\in\Omega} \pi[x] \sum_{y\in\Omega} P'[x, y].$$

Now, $\sum_{y\in\Omega} P'[x, y] = 1$ for all $y \in \Omega$ as P is a stochastic matrix, so $\sum_{y\in\Omega} \pi''[y] = \sum_{x\in\Omega} \pi[x] = 1$. Thus, $\pi P' = \pi'' \in \Delta(\Omega)$. □

A.2 Quantum average mixing bounds

In this section we shall prove the sequence of results that lead to Proposition 2.1 in Section 2.4.3.4 of the text. The proofs have been adapted from those of Godsil and Zhan [GZ17].

Proposition A.1 *Let $\{F_i\}_{i\in[m]}$ be the spectral idempotents of the quantum walk transition matrix U. Let $|\psi(0)\rangle$ be the initial state. For any subset S of the vertices, the average probability that the quantum walk is on some vertex of S converges to*

$$\sum_r \langle\psi(0)|\, F_r D_S F_r\, |\psi(0)\rangle\,,$$

where D_S is the diagonal matrix with ones in elements corresponding to vertices in S, and zeros in the remaining elements.

Proof. Consider the spectral decomposition of U

$$U = \sum_r e^{i\theta_r} F_r.$$

It suffices to show that

$$\frac{1}{T}\sum_{t=0}^{T-1} (U^t)^\dagger D_S U^t$$

converges to

$$\sum_r F_r D_S F_r$$

as T goes to infinity. We have

$$\begin{aligned}(U^t)^* D_S U^t &= \left(\sum_r e^{-it\theta_r} F_r\right) D_S \left(\sum_s e^{it\theta_s} F_r\right)\\ &= \sum_r F_r D_S F_r + \sum_{r\neq s} e^{it(\theta_s-\theta_r)} F_r D_S F_s.\end{aligned}$$

Note that for all r and s, the entries in $F_r D_S F_r$ and $F_r D_S F_s$ are constants, and remain unchanged when we take the average and the limit. Further

$$\frac{1}{T}\left|\sum_{t=0}^{T-1} e^{it(\theta_s-\theta_r)}\right| = \frac{1}{T}\left|\frac{1-e^{iT(\theta_s-\theta_r)}}{1-e^{i(\theta_s-\theta_r)}}\right| \leq \frac{1}{T}\frac{2}{\left|1-e^{i(\theta_s-\theta_r)}\right|}$$

which converges to zero as T goes to infinity. Hence the only term that survives in

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}(U^t)^* D_S U^t$$

is

$$\sum_r F_r D_S F_r.$$

□

Lemma A.1 *For a quantum walk with spectral decomposition*

$$U=\sum_r \lambda_r F_r,$$

we have

$$\sum_j\left|\frac{1}{T}\sum_{t=0}^{T-1}Q_t(j|\psi(0))-\sum_r\langle\psi(0)|\,F_rD_jF_r\,|\psi(0)\rangle\right|\le\frac{2}{T}\sum_{r\neq s}\sum_j\frac{\sqrt{F_r[j,j]F_s[j,j]}}{|\lambda_r-\lambda_s|}.$$

Proof. First note that for any r and s,

$$\begin{aligned}|\langle\psi(0)|\,F_rD_jF_s\,|\psi(0)\rangle| &= |\langle F_re_j\,|\psi(0)\rangle\,\langle F_se_j\,|\psi(0)\rangle| \\ &\le |\langle F_re_j|\psi(0)\rangle|\cdot|\langle F_se_j|\psi(0)\rangle| \\ &\le \sqrt{F_r[j,j]}\||\psi(0)\rangle\|\sqrt{F_s[j,j]}\||\psi(0)\rangle\| \quad \text{(Cauchy-Schwarz)} \\ &= \sqrt{F_r[j,j]F_s[j,j]}.\end{aligned}$$

Let $\lambda_r=e^{i\theta_r}$ for some θ_r. By Proposition A.1 for the j^{th} arc,

$$\begin{aligned}
\left|\frac{1}{T}\sum_{t=0}^{T-1} Q_t(j|\psi(0)) - \sum_r \langle\psi(0)|\, F_r D_j F_r\, |\psi(0)\rangle\right| &= \left|\frac{1}{T}\sum_{t=0}^{T-1}\sum_{r\neq s} \mathrm{e}^{\mathrm{i}t(\theta_s-\theta_r)}\, \langle\psi(0)|\, F_r D_j F_s\, |\psi(0)\rangle\right| \\
&= \frac{1}{T}\left|\sum_{r\neq s}\left(\sum_{t=0}^{T-1} \mathrm{e}^{\mathrm{i}t(\theta_s-\theta_r)}\right) \langle\psi(0)|\, F_r D_j F_s\, |\psi(0)\rangle\right| \\
&= \frac{1}{T}\left|\sum_{r\neq s} \frac{1-\mathrm{e}^{iT(\theta_s-\theta_r)}}{1-\mathrm{e}^{\mathrm{i}(\theta_s-\theta_r)}}\, \langle\psi(0)|\, F_r D_j F_s\, |\psi(0)\rangle\right| \\
&\leq \frac{1}{T}\sum_{r\neq s}\left|\frac{1-\mathrm{e}^{\mathrm{i}T(\theta_s-\theta_r)}}{1-\mathrm{e}^{\mathrm{i}(\theta_s-\theta_r)}}\right| |\langle\psi(0)|\, F_r D_j F_s\, |\psi(0)\rangle| \\
&\leq \frac{1}{T}\sum_{r\neq s} \frac{2}{|\mathrm{e}^{\mathrm{i}\theta_s}-\mathrm{e}^{\mathrm{i}\theta_r}|} |\langle\psi(0)|\, F_r D_j F_s\, |\psi(0)\rangle| \\
&\leq \frac{2}{T}\sum_{r\neq s} \frac{\sqrt{F_r[j,j]F_s[j,j]}}{|\lambda_r-\lambda_s|}.
\end{aligned}$$

Summing over all arcs yields the result. □

One immediate consequence is that we can bound the mixing time of a quantum walk by its eigenvalue differences. This is an analogy to Lemma 4.3 in Aharonov [AAKV01].

Corollary A.1 (Proposition 2.1 in main text) *For a $\ell \times \ell$ transition matrix U with spectral decomposition*

$$U = \sum_r \lambda_r F_r,$$

we have

$$M_\epsilon \leq \frac{2\ell}{\epsilon}\sum_{r\neq s}\frac{1}{|\lambda_r-\lambda_s|}.$$

Proof. Since

$$\sum_r F_r = I, \quad \text{and} \quad F_r^2 = F_r$$

for all r and j we have

$$0 \leq F_r[j,j] \leq 1.$$

Lemma A.1 reduces to

$$\sum_j \left| \frac{1}{T} \sum_{t=0}^{T-1} Q_t(j|\psi(0)) - \sum_r \langle \psi(0)| F_r D_j F_r |\psi(0)\rangle \right| \leq \frac{2}{T} \sum_{r \neq s} \sum_j \frac{\sqrt{F_r[j,j] F_s[j,j]}}{|\lambda_r - \lambda_s|} \leq \frac{2\ell}{T} \sum_{r \neq s} \frac{1}{|\lambda_r - \lambda_s|}.$$

Thus for all T such that

$$T \geq \frac{2\ell}{\epsilon} \sum_{r \neq s} \frac{1}{|\lambda_r - \lambda_s|},$$

the right hand side is bounded above by ϵ. □

A.3 Lovasz-ϑ semidefinite program

A.3.1 Sets of vectors with minimal pairwise inner product

The following results are used to justify the definition of the Lovász-ϑ SDP in Section 4.1.

Lemma A.2 *Let $k, n \in \mathbb{N}$ such that $k \leq n+1$ and let $v_i, \ldots, v_k$ be unit vectors in $\mathbb{R}^n$. Then we have*

$$\min_{v_i,\ldots,v_k \in \mathcal{S}(\mathbb{R}^n)} \sum_{i \neq j} \langle v_i, v_j \rangle \geq -k \tag{A.1}$$

Proof. We shall formulate the optimisation problem (A.1) as a semidefinite program then use weak duality to show the bound. Recall that a Gram matrix X of a set of vectors $\{v_1, \ldots, v_k\}$ is the matrix whose entries are given by $X[i,j] = \langle v_i, v_j \rangle$. Moreover, any $n \times n$ symmetric positive semidefinite matrix realises a Gram matrix for some set of n vectors. We can thus solve (A.1) using the Gram matrix like so:

$$\begin{aligned} \min \quad & \sum_{i \neq j} X[i,j] \\ \text{s.t.} \quad & X[i,j] = X[j,i] \qquad \forall i,j \in [k] \\ & X[i,i] = 0 \qquad \forall i \in [k] \\ & X \succeq 0 \end{aligned} \tag{A.2}$$

where the first line of constraints enforces $\langle v_i, v_j \rangle = \langle v_j, v_i \rangle$ and the second line enforces that $\langle v_i, v_i \rangle = 1$. Rewriting in SDP standard form (2.23) we can cast (A.2) as

$$\begin{aligned}
\min \quad & \mathrm{Tr}((J_k - I_k)X) && \text{(A.3a)}\\
\text{s.t.} \quad & \mathrm{Tr}((E_{i,j} - E_{j,i})X) = 0 \qquad \forall i,j \in [k] && \text{(A.3b)}\\
& \mathrm{Tr}(E_{i,i}X)) = 1 \qquad \forall i \in [k] && \text{(A.3c)}\\
& X \succeq 0 && \text{(A.3d)}
\end{aligned}$$

where J_k is the $k \times k$ all-ones matrix and $E_{i,j} := e_i e_j^\mathsf{T}$. We denote the set of constraints (A.3b) by $\mathcal{B}$ and the set of constraints (A.3c) by $\mathcal{C}$. We can now form the dual program (See Eq. 2.24 for the standard form).

$$\begin{aligned}
\max \quad & \sum_{i \in \mathcal{C}} y[i] \\
\text{s.t.} \quad & \sum_{i \in \mathcal{B}} y[i](E_{i,j} - E_{j,i}) + \sum_{i \in \mathcal{C}} y[i] E_{i,i} \preceq J_k - I_k && \text{(A.4)}\\
& y \in \mathbb{R}^{|\mathcal{B}|+|\mathcal{C}|}.
\end{aligned}$$

By weak duality [BV09, Chapter 5], for any feasible (X, y) pair we have $\mathrm{Tr}(CX) \geq b^\mathsf{T} y$ where $\mathrm{Tr}(CX)$, $b^\mathsf{T} y$ are the objective functions of the primal and dual respectively. For feasible X we choose the Gram matrix of any set of k vectors from $\mathcal{S}(\mathbb{R}^n)$. For feasible y, we choose the vector with zeros in elements corresponding to $\mathcal{B}$ and negative ones for the elements corresponding to $\mathcal{C}$. The left hand side of the constraint in (A.4) thus becomes

$$\sum_{i \in \mathcal{C}} y[i] E_{i,i} = -I_k,$$

and so the constraint is satisfied, since $J_k = \mathbf{1}\mathbf{1}^\mathsf{T} \succeq 0$. This gives a dual objective value of $\sum_{i \in \mathcal{C}} y[i] = -k$, and the result follows from weak duality. □

We can now show that the minimum in the problem (A.1) is in fact attained. We take the following result from [KKMS98, Lemma 4.1]

Lemma A.3 *For all positive integers k and n such that $k \leq n+1$, there exist k unit vectors in $\mathbb{R}^n$ such that the dot product of any distinct pair is $-1/(k-1)$.*

Proof. It suffices to prove the lemma for $n = k-1$. (For other values of n, we make

the coordinates of the vectors 0 in all but the first $k-1$ coordinates.) We begin by proving the claim for $n=k$. We explicitly provide unit vectors $v_1^{(k)},\ldots,v_k^{(k)} \in \mathbb{R}^k$ such that $\left\langle v_i^{(k)}, v_j^{(k)} \right\rangle = -1/(k-1)$ for $i \neq j$. The vector $v_i^{(k)}$ is $-\sqrt{\frac{1}{k(k-1)}}$ in all coordinates except the i^{th} coordinate. In the i^{th} coordinate $v_i^{(k)}$ is $\sqrt{\frac{k-1}{k}}$. It is straightforward to verify that the vectors are unit length and that their dot products are exactly $-\frac{1}{k-1}$. As given, the vectors are in a k-dimensional space. Note, however, that the dot product of each vector with the all-ones vector $\mathbf{1}_k$ is 0. Thus, we have that all k of the vectors lie in a $(k-1)$-dimensional hyperplane of the k-dimensional space. This proves the lemma. □

By combining Lemmas A.2 and A.3 we are led to the following result.

Corollary A.2 *Let $k,n \in \mathbb{N}$ such that $k \leq n+1$ and let $v_i,\ldots,v_k$ be unit vectors in $\mathbb{R}^n$. Then we have*

$$\min_{v_i,\ldots,v_k \in \mathcal{S}(\mathbb{R}^n)} \sum_{i \neq j} \langle v_i, v_j \rangle = -k, \tag{A.5}$$

with the minimum being attained by vectors satisfying

$$\langle v_i, v_j \rangle = -\frac{1}{k-1} \quad \text{for all} \quad i \neq j. \tag{A.6}$$

Proof. By Lemma A.2 we have that $\min_{v_i,\ldots,v_k \in \mathcal{S}(\mathbb{R}^n)} \sum_{i \neq j} \langle v_i, v_j \rangle \geq -k$. We now show that the set of vectors from Lemma A.3 saturates this inequality. The sum in the minimisation has $k(k-1)$ terms, each of which takes on the value $-1/(k-1)$, giving a total of $-k$. This same set of vectors satisfies Eq. A.6 by construction. □

A.3.2 Lovász's sandwich theorem

We now prove Lovász's famous "sandwich theorem".

Proposition A.2 (Proposition 4.1) *Let G be a graph. Then, $\alpha(G) \leq \vartheta(G) \leq \bar{\chi}(G)$.*

Proof. As described in the discussion of Section 4.1, any clique cover corresponds to a feasible solution of the Lovász-ϑ SDP (4.1) with an objective function value equal to the size of the clique cover. This implies that $\vartheta(G) \leq \bar{\chi}(G)$. It remains to show that $\alpha(G) \leq \vartheta(G)$. Suppose that $v_1,\ldots,v_s$ are the SDP solution vectors corresponding to a maximal independent set of size $s=\alpha(G)$ and let $v = \sum_{i=1}^{s} v_i$.

Then $v^\mathsf{T} v \geq 0$. It is also true that

$$v^\mathsf{T} v = \left(\sum_{i=1}^{s} v_i\right)^\mathsf{T} \left(\sum_{i=1}^{s} v_i\right) = \sum_{i=1}^{s} v_i^\mathsf{T} v_i + \sum_{i \neq j} v_i^\mathsf{T} v_j = s + \sum_{i \neq j} v_i^\mathsf{T} v_j.$$

We thus have that $s + \sum_{i \neq j} v_i^T v_j \geq 0$. There are $s(s-1)$ terms in the sum, so, by averaging, there exist some distinct i and j such that

$$v_i^\mathsf{T} v_j \geq -\frac{s}{s(s-1)} = -\frac{1}{s-1}.$$

Since $v_i^\mathsf{T} v_j = -\frac{1}{\vartheta(G)-1}$ by the SDP constraints, $\alpha(G) = s \leq \vartheta(G)$. Therefore, we can conclude that $\alpha(G) \leq \vartheta(G) \leq \bar{\chi}(G)$. □

Bibliography

[AAKV01] D. Aharonov, A. Ambainis, J. Kempe and U. Vazirani. *Quantum walks on graphs*. In *Proc. thirty-third Annu. ACM Symp. Theory Comput. - STOC '01*, *50–59*. ACM Press, New York, NY, USA (2001). [Cited on pages 40, 43, 44, 48, 49, 50, and 132.]

[Aar05] S. Aaronson. *Quantum computing and hidden variables*. *Phys. Rev. A*, ***71***, *032325* (2005). [Cited on page 72.]

[Aar16] S. Aaronson. $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$. In J. F. Nash, Jr. and M. T. Rassias, editors, *Open Problems in Mathematics*. Springer International Publishing (2016). [Cited on page 26.]

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st ed. (2009). [Cited on pages 25 and 53.]

[ABI+18] A. Ambainis *et al.* *Quantum Speedups for Exponential-Time Dynamic Programming Algorithms* (2018). `arXiv:1807.05209`. [Cited on page 60.]

[ABTW03] A. Ahmadi, R. Belk, C. Tamon and C. Wendler. *On Mixing in Continuous-time Quantum Walks on Some Circulant Graphs*. *Quantum Info. Comput.*, ***3***, *611* (2003). [Cited on page 49.]

[AdFDJ03] C. Andrieu, N. de Freitas, A. Doucet and M. I. Jordan. *An Introduction to MCMC for Machine Learning*. *Mach. Learn.*, ***50***, *5* (2003). [Cited on page 34.]

[ADZ93] Y. Aharonov, L. Davidovich and N. Zagury. *Quantum random walks*. *Phys. Rev. A*, ***48***, *1687* (1993). [Cited on page 40.]

[AGJK19] A. Ambainis, A. Gilyén, S. Jeffery and M. Kokainis. *Quadratic speedup for finding marked vertices by quantum walks* (2019). `arXiv:1903.07493`. [Cited on page 47.]

[AIP08] A. Alzaga, R. Iglesias and R. Pignol. *Spectra of symmetric powers of graphs and the Weisfeiler-Lehman refinements* (2008). `arXiv:0801.2322`. [Cited on page 64.]

[AK98] N. Alon and N. Kahale. *Approximating the independence number via the ϑ-function*. *Mathematical Programming*, ***80***, *253* (1998). [Cited on page 84.]

[AKR04] A. Ambainis, J. Kempe and A. Rivosh. *Coins make quantum walks faster*. In *SODA '05 Proc. Sixt. Annu. ACM-SIAM Symp. Discret. algorithms*, *1188*. Association for Computing Machinery (2004). [Cited on pages 41 and 47.]

[Amb04] A. Ambainis. *Quantum walk algorithm for element distinctness*. In *45th Annu. IEEE Symp. Found. Comput. Sci.*, vol. 0354, *1–33* (2004). [Cited on page 47.]

[AMR+19] A. Atserias *et al. Quantum and non-signalling graph isomorphisms*. *Journal of Combinatorial Theory, Series B*, ***136***, *289* (2019). [Cited on page 64.]

[AS18] S. Apers and A. Sarlette. *Quantum Fast-Forwarding: Markov Chains and Graph Property Testing* (2018). `arXiv:1804.02321`. [Cited on page 48.]

[AST17] S. Apers, A. Sarlette and F. Ticozzi. *Fast Mixing with Quantum Walks vs. Classical Processes*. In *Quantum Information Processing (QIP) 2017*. Seattle, United States. `<hal-01395592>` (2017). [Cited on page 79.]

[AST18] S. Apers, A. Sarlette and F. Ticozzi. *Simulation of quantum walks and fast mixing with classical processes*. *Phys. Rev. A*, ***98***, *032115* (2018). [Cited on pages 66, 71, 72, 80, 125, and 127.]

[ATS03] D. Aharonov and A. Ta-Shma. *Adiabatic Quantum State Generation and Statistical Zero Knowledge*. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, *20–29*. ACM, New York, NY, USA (2003). [Cited on page 63.]

[ATS17] S. Apers, F. Ticozzi and A. Sarlette. *Lifting Markov Chains To Mix Faster: Limits and Opportunities* (2017). `arXiv:1705.08253`. [Cited on pages 36, 38, 39, 66, 67, 68, and 69.]

[AvDK+08] D. Aharonov *et al. Adiabatic Quantum Computation Is Equivalent to Standard Quantum Computation*. *SIAM Review*, ***50***, *755* (2008). [Cited on page 27.]

[Bab15] L. Babai. *Graph Isomorphism in Quasipolynomial Time* (2015). `arXiv:1512.03547`. [Cited on pages 61 and 62.]

[Bac10] D. Bacon. *Reading List: Graph Isomorphism* (2010). URL `http://dabacon.org/pontiff/?p=4148`. [Cited on page 62.]

[BDX04] S. Boyd, P. Diaconis and L. Xiao. *Fastest Mixing Markov Chain on a Graph*. *SIAM Rev.*, ***46***, *667* (2004). [Cited on page 36.]

[BGK18] S. Bravyi, D. Gosset and R. König. *Quantum advantage with shallow circuits*. *Science*, ***362***, *308* (2018). [Cited on page 31.]

[BLS13] I. Boussaïd, J. Lepagnot and P. Siarry. *A survey on optimization metaheuristics*. *Information Sciences*, ***237***, *82* (2013). Prediction, Control and Diagnosis using Advanced Neural Computations. [Cited on page 55.]

[BN16] D. Berry and L. Novo. *Corrected Quantum Walk for Optimal Hamiltonian Simulation*. *Quantum Info. Comput.*, ***16***, *1295* (2016). [Cited on page 48.]

[Bos03] S. Bose. *Quantum Communication through an Unmodulated Spin Chain*. *Phys. Rev. Lett.*, ***91***, *207901* (2003). [Cited on pages 41 and 126.]

[BP09] A. R. Barghi and I. Ponomarenko. *Non-Isomorphic Graphs with Cospectral Symmetric Powers*. *Electron. J. Comb.*, ***16***, *R120* (2009). [Cited on page 64.]

[BR03] C. Blum and A. Roli. *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison*. *ACM Comput. Surv.*, ***35***, *268* (2003). [Cited on page 55.]

[BS18] F. G. S. L. Brandão and K. Svore. *Quantum Speed-ups for Semidefinite Programming*. In *58th Annu. IEEE Symp. Found. Comput. Sci.*, *403–414* (2018). [Cited on page 57.]

[BV09] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press (2009). [Cited on pages 54 and 134.]

[CC82] A. Cardon and M. Crochemore. *Partitioning a graph in* $O(|A| \log_2 |V|)$. *Theor. Comput. Sci.*, ***19***, *85* (1982). [Cited on page 62.]

[CEL86] K. Cameron, J. Edmonds and L. Lovász. *A note on perfect graphs*. *Period. Math. Hungarica*, ***17***, *173* (1986). [Cited on page 94.]

[CFG01] A. M. Childs, E. Farhi and S. Gutmann. *An example of the difference between quantum and classical random walks*. *Quantum Inf. Process.*, ***1***, *35* (2001). [Cited on page 41.]

[CFI92] J.-Y. Cai, M. Fürer and N. Immerman. *An Optimal Lower Bound on the Number of Variables for Graph Identification*. ***12***, *389* (1992). [Cited on page 62.]

[CG04] A. M. Childs and J. Goldstone. *Spatial search by quantum walk*. *Phys. Rev. A*, ***70***, *022314* (2004). [Cited on page 41.]

[Chi09] A. M. Childs. *Universal computation by quantum walk*. *Phys. Rev. Lett.*, ***102***, *180501* (2009). [Cited on pages 50 and 51.]

[CHKX06] J. Chen, X. Huang, I. A. Kanj and G. Xia. *Strong computational lower bounds via parameterized complexity*. *Journal of Computer and System Sciences*, ***72***, *1346* (2006). [Cited on page 127.]

[CLB81] D. Corneil, H. Lerchs and L. Burlingham. *Complement reducible graphs*. *Discret. Appl. Math.*, ***3***, *163* (1981). [Cited on page 122.]

[CLP99] F. Chen, L. Lovász and I. Pak. *Lifting Markov chains to speed up mixing*. In *Proc. thirty-first Annu. ACM Symp. Theory Comput. - STOC '99*, *275–281*. ACM Press, New York, New York, USA (1999). [Cited on pages 36, 38, 39, 68, and 69.]

[CLR19] S. Chakraborty, K. Luh and J. Roland. *On analog quantum algorithms for the mixing of Markov chains* (2019). `arXiv:1904.11895`. [Cited on page 50.]

[CRST06] M. Chudnovsky, N. Robertson, P. Seymour and R. Thomas. *The strong perfect graph theorem*. *Ann. Math.*, ***164***, *51* (2006). [Cited on pages 89, 94, and 125.]

[CTV17] E. T. Campbell, B. M. Terhal and C. Vuillot. *Roads towards fault-tolerant universal quantum computation*. Nature, ***549***, *172* (2017). [Cited on page 30.]

[CvD10] A. M. Childs and W. van Dam. *Quantum algorithms for algebraic problems*. *Rev. Mod. Phys.*, ***82***, *1* (2010). [Cited on page 63.]

[CZ95] J. I. Cirac and P. Zoller. *Quantum Computations with Cold Trapped Ions*. *Phys. Rev. Lett.*, ***74***, *4091* (1995). [Cited on page 27.]

[D2X] *The D-Wave 2X^{TM} Quantum Computer: Technology Overview* (2015). URL `http://www.dwavesys.com/sites/default/files/D-Wave2XTechCollateral_0915F.pdf`. [Cited on page 60.]

[DHN00] P. Diaconis, S. Holmes and R. M. Neal. *Analysis of a nonreversible Markov chain sampler*. *Ann. Appl. Probab.*, ***10***, *726* (2000). [Cited on pages 36, 37, and 38.]

[Dia08] P. Diaconis. *The Markov chain Monte Carlo revolution*. *Bull. Am. Math. Soc.*, ***46***, *179* (2008). [Cited on page 35.]

[DWM04] M. H. Devoret, A. Wallraff and J. M. Martinis. *Superconducting Qubits: A Short Review* (2004). `arXiv:cond-mat/0411174`. [Cited on page 27.]

[ECR+07] G. S. Engel *et al.* *Evidence for wavelike energy transfer through quantum coherence in photosynthetic systems*. Nature, ***446***, *782 EP* (2007). [Cited on page 40.]

[EH99] M. Ettinger and P. Hoyer. *A Quantum Observable for the Graph Isomorphism Problem* (1999). `arXiv:quant-ph/9901029`. [Cited on page 63.]

[EHSW06] D. Emms, E. R. Hancock, S. Severini and R. C. Wilson. *A matrix representation of graphs and its spectrum as a graph invariant*. Electron. J. Comb., ***13***, *R34* (2006). [Cited on page 63.]

[FAJ04] A. P. Flitney, D. Abbott and N. F. Johnson. *Quantum walks with history dependence*. Journal of Physics A: Mathematical and General, ***37***, *7581* (2004). [Cited on page 53.]

[FB17] G. França and J. Bento. *Markov Chain Lifting and Distributed ADMM*. *IEEE Signal Process. Lett.*, ***24***, *294* (2017). [Cited on page 37.]

[Fei97] U. Feige. *Randomized graph products, chromatic numbers, and the Lovász ϑ-function*. Combinatorica, ***17***, *79* (1997). [Cited on page 84.]

[FF56] L. R. Ford and D. R. Fulkerson. *Maximal Flow Through a Network*. Can. J. Math., ***8***, *399–404* (1956). [Cited on page 75.]

[FG98] E. Farhi and S. Gutmann. *Quantum Computation and Decision Trees*. *Phys. Rev. A*, ***58***, *28* (1998). [Cited on page 41.]

[FGG14] E. Farhi, J. Goldstone and S. Gutmann. *A Quantum Approximate Optimization Algorithm* (2014). `arXiv:1411.4028v1`. [Cited on page 59.]

[FGGS00] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser. *Quantum Computation by Adiabatic Evolution* (2000). arXiv:quant-ph/0001106. [Cited on page 60.]

[FST06] L. Fedichkin, D. Solenov and C. Tamon. *Mixing and Decoherence in Continuous-time Quantum Walks on Cycles*. Quantum Info. Comput., ***6***, *263* (2006). [Cited on page 49.]

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of* NP*-Completeness*. W. H. Freeman & Co., New York, NY, USA (1979). [Cited on pages 53 and 61.]

[GLS93] M. Grötschel, L. Lovász and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag Berlin Heidelberg (1993). [Cited on pages 83 and 84.]

[GMW91] O. Goldreich, S. Micali and A. Wigderson. *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems*. *J. ACM*, ***38***, *690* (1991). [Cited on page 61.]

[Gro97] L. K. Grover. *Quantum Mechanics Helps in Searching for a Needle in a Haystack*. *Phys. Rev. Lett.*, ***79***, *325* (1997). [Cited on page 30.]

[Gro10] M. Grohe. *Fixed-Point Definability and Polynomial Time on Graphs with Excluded Minors*. In *2010 25th Annu. IEEE Symp. Log. Comput. Sci.*, *179–188* (2010). [Cited on page 61.]

[GW95] M. X. Goemans and D. P. Williamson. *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*. *J. ACM*, ***42***, *1115* (1995). [Cited on page 54.]

[GW11] A. Gupta and D. Witmer. *15-859(E): Linear and Semidefinite Programming (Advanced Algorithms). Lecture 11.* (2011). URL https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15859-f11/www/notes/lecture11.pdf. [Cited on page 82.]

[GZ17] C. Godsil and H. Zhan. *Discrete-Time Quantum Walks and Graph Structures* (2017). arXiv:1701.04474. [Cited on pages 40, 44, 50, and 130.]

[Hås99] J. Håstad. *Clique is hard to approximate within a factor of* $n^{1-\epsilon}$. *Acta. Math.*, ***182***, *105* (1999). [Cited on pages 84 and 127.]

[HIK11] R. H. Hammack, W. Imrich and S. Klavžar. *Handbook of product graphs*. CRC Press (2011). [Cited on pages 86, 87, and 127.]

[HM17] A. W. Harrow and A. Montanaro. *Quantum computational supremacy*. *Nature*, ***549***, *203* (2017). [Cited on page 65.]

[HWO+17] S. Hadfield *et al. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz* (2017). arXiv:1709.03489. [Cited on page 60.]

[ISGCI] H. N. de Ridder *et al. Information System on Graph Classes and their Inclusions (ISGCI)* (2014). URL `http://www.graphclasses.org`. [Cited on page 125.]

[JSS10] K. Jung, D. Shah and J. Shin. *Distributed averaging via lifted markov chains. IEEE Trans. Inform. Theory, 634–647* (2010). [Cited on page 37.]

[Kan98] B. E. Kane. *A silicon-based nuclear spin quantum computer*. Nature, ***393***, *133* (1998). [Cited on page 27.]

[Ken06] V. Kendon. *Quantum walks on general graphs. Int. J. Quantum Inf.*, ***04***, *791* (2006). [Cited on page 43.]

[KF09] D. Koller and N. Friedman. *Probabilistic graphical models : principles and techniques*. MIT Press (2009). [Cited on page 35.]

[KG98] J. Kleinberg and M. X. Goemans. *The Lovász Theta Function and a Semidefinite Programming Relaxation of Vertex Cover. SIAM J. Discret. Math.*, ***11***, *196* (1998). [Cited on page 84.]

[KKMS98] D. Karger, D. Karger, R. Motwani and M. Sudan. *Approximate Graph Coloring by Semidefinite Programming. J. ACM*, ***45***, *246* (1998). [Cited on page 134.]

[KLM01] E. Knill, R. Laflamme and G. J. Milburn. *A scheme for efficient quantum computation with linear optics*. Nature, ***409***, *46* (2001). [Cited on page 27.]

[KMOR10] H. Krovi, F. Magniez, M. Ozols and J. Roland. *Finding Is as Easy as Detecting for Quantum Walks*. In S. Abramsky *et al.*, editors, *Automata, Languages and Programming*, *540–551*. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). [Cited on page 47.]

[KN12] M. Kieferová and D. Nagaj. *Quantum walks on necklaces and mixing*. International Journal of Quantum Information, ***10***, *1250025* (2012). [Cited on page 49.]

[Kon81] S. V. Konyagin. *Systems of vectors in Euclidean space and an extremal problem for polynomials*. Mathematical Notes, ***29***, *33* (1981). [Cited on page 84.]

[Koz78] D. Kozen. *A clique problem equivalent to graph isomorphism*. ACM SIGACT News, ***10***, *50* (1978). [Cited on pages 21, 88, 89, and 124.]

[KPRS09] J. Kallrath, P. M. Pardalos, S. Rebennack and M. Scheidt. *Optimization in the Energy Industry*. Springer-Verlag Berlin Heidelberg (2009). [Cited on page 53.]

[KV12] B. H. Korte and J. Vygen. *Combinatorial optimization : theory and algorithms*. Springer-Verlag Berlin Heidelberg (2012). [Cited on pages 53 and 74.]

[KXB+16] D. Korenkevych *et al. Benchmarking Quantum Hardware for Training of Fully Visible Boltzmann Machines* (2016). `arXiv:1611.04528`. [Cited on page 61.]

[KYR+17] J. King *et al. Quantum Annealing amid Local Ruggedness and Global Frustration* (2017). `arXiv:1701.04579`. [Cited on page 61.]

[Las02] J. B. Lasserre. *An Explicit Equivalent Positive Semidefinite Program For Nonlinear 0-1 Programs. Siam J. Optim.*, ***12***, *756* (2002). [Cited on page 54.]

[Lau03] M. Laurent. *A Comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre Relaxations for 0–1 Programming*. Math. Oper. Res., ***28***, *470* (2003). [Cited on page 54.]

[LCE+10] N. B. Lovett *et al. Universal quantum computation using the discrete-time quantum walk*. Phys. Rev. A, ***81***, *042330* (2010). [Cited on pages 50 and 51.]

[Ler72] H. Lerchs. *On the clique–kernel structure of graphs*. Tech. Report Dept. of Comp. Sci., Univ. of Toronto (1972). [Cited on page 93.]

[Lov72] L. Lovász. *A characterization of perfect graphs*. J. Comb. Theory, Ser. B, ***13***, *95* (1972). [Cited on page 94.]

[Lov79] L. Lovász. *On the Shannon capacity of a graph*. IEEE Trans. Inf. Theory, ***25***, *1* (1979). [Cited on pages 83 and 86.]

[LPW09] D. A. Levin, Y. Y. Peres and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Society (2009). [Cited on pages 33, 34, 35, 36, and 55.]

[LR19] N. Lindzey and A. Rosmanis. *A Tight Lower Bound for Index Erasure* (2019). `arXiv:1902.07336`. [Cited on page 63.]

[LSW15] Y. T. Lee, A. Sidford and S. C.-W. Wong. *A Faster Cutting Plane Method and Its Implications for Combinatorial and Convex Optimization*. In *Proc. 56th Annu. IEEE Symp. Found. Comput. Sci., 1049–1065* (2015). [Cited on page 121.]

[Mac00] C. MacCluer. *The Many Proofs and Applications of Perron's Theorem*. SIAM Review, ***42***, *487* (2000). [Cited on page 41.]

[McG10] M. McGettrick. *One Dimensional Quantum Walks with Memory*. Quantum Info. Comput., ***10***, *509* (2010). [Cited on page 53.]

[MCvD+16] A. Montanaro *et al. Quantum algorithms: an overview*. npj Quantum Inf., ***2***, *15023* (2016). [Cited on page 31.]

[Mey96a] D. A. Meyer. *From quantum cellular automata to quantum lattice gases*. Journal of Statistical Physics, ***85***, *551* (1996). [Cited on page 40.]

[Mey96b] D. A. Meyer. *On the absence of homogeneous scalar unitary cellular automata*. Physics Letters A, ***223***, *337* (1996). [Cited on page 40.]

[MK82] Molloy and M. K. *Performance Analysis Using Stochastic Petri Nets*. IEEE Trans. Comput., ***C-31***, *913* (1982). [Cited on page 35.]

[MKM78] V. Malhotra, M. Kumar and S. Maheshwari. *An $O(|V|^3)$ algorithm for finding maximum flows in networks*. Inf. Process. Lett., ***7***, *277* (1978). [Cited on page 77.]

[MNRS11] F. Magniez, A. Nayak, J. Roland and M. Santha. *Search via Quantum Walk*. SIAM Journal on Computing, ***40***, *142* (2011). [Cited on page 47.]

[Mon15] A. Montanaro. *Quantum speedup of Monte Carlo methods*. Proc. R. Soc. A Math. Phys. Eng. Sci., ***471***, *20150301* (2015). [Cited on page 56.]

[Mon19] A. Montanaro. *Quantum speedup of branch-and-bound algorithms* (2019). arXiv:1906.10375. [Cited on page 60.]

[MP14] B. D. McKay and A. Piperno. *Practical graph isomorphism, II*. *J. Symb. Comput.*, ***60***, *94* (2014). [Cited on pages 61 and 62.]

[MPA10] F. L. Marquezino, R. Portugal and G. Abal. *Mixing times in quantum walks on two-dimensional grids*. *Phys. Rev. A*, ***82***, *042341* (2010). [Cited on page 49.]

[MPAD08] F. L. Marquezino, R. Portugal, G. Abal and R. Donangelo. *Mixing times in quantum walks on the hypercube*. *Phys. Rev. A*, ***77***, *042312* (2008). [Cited on page 49.]

[MRLAG08] M. Mohseni, P. Rebentrost, S. Lloyd and A. Aspuru-Guzik. *Environment-assisted quantum walks in photosynthetic energy transfer*. *The Journal of Chemical Physics*, ***129***, *174106* (2008). [Cited on page 40.]

[MRS+17] P. W. Mills *et al. On quantum invariants and the graph isomorphism problem* (2017). arXiv:1711.09842. [Cited on page 63.]

[NAoSM19] E. National Academies of Sciences and Medicine. *Quantum Computing: Progress and Prospects*. The National Academies Press, Washington, DC (2019). [Cited on page 27.]

[NC10] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge (2010). [Cited on pages 27 and 30.]

[Ola88] S. Olariu. *Paw-free graphs*. *Inf. Process. Lett.*, ***28***, *53* (1988). [Cited on page 92.]

[OMA+17] J. S. Otterbach *et al. Unsupervised Machine Learning on a Hybrid Quantum Computer* (2017). arXiv:1712.05771. [Cited on page 60.]

[PH02] I. C. Parmee and P. Hajela. *Optimization in Industry*. Springer-Verlag London (2002). [Cited on page 53.]

[PK03] P. M. Pardalos and V. Korotkikh. *Optimization and Industry: New Frontiers*. Springer US (2003). [Cited on page 53.]

[Pon91] I. N. Ponomarenko. *The isomorphism problem for classes of graphs closed under contraction*. *J. Sov. Math.*, ***55***, *1621* (1991). [Cited on page 61.]

[Pre13] J. Preskill. *Sufficient Condition on Noise Correlations for Scalable Quantum Computing*. *Quantum Info. Comput.*, ***13***, *181* (2013). [Cited on page 30.]

[Pre18] J. Preskill. *Quantum Computing in the NISQ era and beyond*. *Quantum*, ***2***, *79* (2018). [Cited on page 65.]

[PY81] C. H. Papadimitriou and M. Yannakakis. *The clique problem for planar graphs*. *Information Processing Letters*, ***13***, *131* (1981). [Cited on page 127.]

[Rav78] G. Ravindra. *Perfectness of normal products of graphs*. *Discrete Math.*, ***24***, *291* (1978). [Cited on pages 89 and 104.]

[RBB03] R. Raussendorf, D. E. Browne and H. J. Briegel. *Measurement-based quantum computation on cluster states. Phys. Rev. A*, ***68****, 022312* (2003). [Cited on page 27.]

[RP77] G. Ravindra and K. Parthasarathy. *Perfect product graphs. Discrete Math.*, ***20****, 177* (1977). [Cited on pages 89, 95, and 104.]

[RS16] K. Ramanan and A. Smith. *Bounds on Lifting Continuous Markov Chains to Speed Up Mixing* (2016). `arXiv:1606.03161`. [Cited on page 37.]

[Rud02] T. Rudolph. *Constructing physically intuitive graph invariants* (2002). `arXiv:quant-ph/0206068`. [Cited on page 63.]

[SAGE] W. Stein *et. al. Sage Mathematics Software.* URL `http://sagemath.org`. [Cited on pages 85 and 125.]

[Sei74] D. Seinsche. *On a property of the class of n-colorable graphs. J. Comb. Theory, Ser. B*, ***16****, 191* (1974). [Cited on page 93.]

[Sev05] S. Severini. *Graphs of unitary matrices* (2005). `arXiv:math/0303084`. [Cited on page 43.]

[Sey06] P. Seymour. *How the proof of the strong perfect graph conjecture was found. Gazette des Mathématiciens*, ***109****, 69* (2006). [Cited on page 126.]

[Shi19] Y. Shitov. *Counterexamples to Hedetniemi's conjecture* (2019). `arXiv:1905.02167`. [Cited on page 86.]

[Sho97] P. W. Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. Sci. Stat. Comput.*, ***26****, 1484* (1997). [Cited on pages 30, 62, and 63.]

[Sin93] A. Sinclair. *Algorithms for Random Generation and Counting: A Markov Chain Approach.* Birkhäuser Boston, Boston, MA (1993). [Cited on page 34.]

[SP19] I. Sinayskiy and F. Petruccione. *Open quantum walks. The European Physical Journal Special Topics*, ***227****, 1869* (2019). [Cited on page 41.]

[Sze04] M. Szegedy. *Quantum speed-up of Markov chain based algorithms.* In *45th Annu. IEEE Symp. Found. Comput. Sci., 32–41* (2004). [Cited on pages 43 and 46.]

[UTN05] R. Uehara, S. Toda and T. Nagoya. *Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. Discret. Appl. Math.*, ***145****, 479* (2005). [Cited on pages 62 and 89.]

[VA12] S. E. Venegas-Andraca. *Quantum walks: a comprehensive review. Quantum Inf. Process.*, ***11****, 1015* (2012). [Cited on page 40.]

[vAG18] J. van Apeldoorn and A. Gilyén. *Improvements in Quantum SDP-Solving with Applications* (2018). `arXiv:1804.05058`. [Cited on page 59.]

[vAG19] J. van Apeldoorn and A. Gilyén. *Quantum algorithms for zero-sum games* (2019). `arXiv:1904.03180`. [Cited on page 59.]

[vGGdW17] J. van Apeldoorn, A. Gilyén, S. Gribling and R. de Wolf. *Quantum SDP-Solvers: Better Upper and Lower Bounds*. In *58th Annu. IEEE Symp. Found. Comput. Sci.*, *403–414* (2017). [Cited on pages 59 and 85.]

[Vuc16] M. Vucelja. *Lifting—A nonreversible Markov chain Monte Carlo algorithm*. *Am. J. Phys*, ***84***, *958* (2016). [Cited on page 37.]

[Wat08] J. Watrous. *Quantum Computational Complexity* (2008). `arXiv:0804.3401`. [Cited on page 31.]

[Wat18] J. Watrous. *The Theory of Quantum Information*. Cambridge University Press (2018). [Cited on pages 21 and 22.]

[WL68] B. Y. Weisfeiler and A. A. Lehman. *Reduction of a graph to a canonical form and an algebra which appears in the process*. *Nauchno-Technicheskaya Informatsiya, Ser. 2*, ***9***, *12* (1968). [Cited on page 62.]

[YAG12] M.-H. Yung and A. Aspuru-Guzik. *A quantum-quantum Metropolis algorithm*. *Proc. Natl. Acad. Sci. U. S. A.*, ***109***, *754* (2012). [Cited on page 57.]

[ZKT85] V. N. Zemlyachenko, N. M. Korneenko and R. I. Tyshkevich. *Graph isomorphism problem*. *J. Sov. Math.*, ***29***, *1426* (1985). [Cited on page 62.]