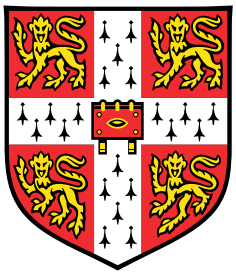
Near-term quantum computational utility through simulation and algorithms



Joshua James Craig Cudby

University of Cambridge

This thesis is submitted for the degree of
Doctor of Philosophy

Jesus College

November 2025

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration or content generated by artificial intelligence, except as declared in the preface and specified in the text. It is not substantially the same as any work that has already been submitted, or is being concurrently submitted, for any degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Abstract

This thesis examines the theoretical foundations that enable *near-term quantum computational utility*. Quantum utility occurs when a quantum computer contributes meaningfully to solving a computational task that a classical computer cannot alone replicate without additional time or loss of accuracy. Unlike *quantum advantage*, which requires provable superiority over all classical methods, quantum utility requires only practical usefulness, employing quantum devices as effective tools for scientific discovery. Recognising this distinction is essential for directing research towards pragmatic goals achievable with current and emerging hardware.

The thesis comprises a three-part investigation encompassing theoretical limits, algorithmic design, and hybrid applications. First, it characterises the boundary between classically simulable and intrinsically quantum computations by analysing a specific class of simulable circuits. Second, it designs and analyses specialised quantum algorithms for structured problems using the framework of parameterized complexity. Finally, it demonstrates pragmatic algorithm design through a hybrid classical–quantum approach to a key problem in computational genomics, illustrating how theoretical progress can translate into tangible near-term utility.

"Y'know, it's very hard to talk quantum using a language originally designed to tell other monkeys where the ripe fruit is." – Sweeper, "Night Watch" by Terry Pratchett.

Acknowledgements

First and foremost, I must thank my supervisor, Sergii. Thank you for the interesting discussions, for offering a wide range of opportunities, and for always keeping your students' interests in mind.

I am forever grateful to my family for their support throughout my life. Special thanks to my parents for being as supportive as I could ask for, and for always encouraging and enabling me at school and in life. I owe all my success to you.

My friends, peers and colleagues in the CMS have hugely improved my time in Cambridge. Mitchell and Wilf – thank you for inducting me into the PhD life, for the many lunch breaks and other procrastinations, and even for the occasional productive discussion. Thanks also to Campbell and Orson for the conference trips and the resulting chaos. The QPG team also played a significant role in the last two years of my thesis – special thanks to Floyd and Hitham for the cafe days and work travels.

Life in Cambridge, despite appearances, is not all about work. During my years here, I have met many amazing people, too many to name here.

From undergrad, special mentions to Emma, Hannah, Helena, Henry, Laura, Matt, and Satvik for keeping me sane during those crazy-busy terms, and to Theo for being the best supo partner I could hope for.

From postgrad, thanks to the *Roost*, the *Survivors*, and the *Thrivers*, for the many coffee mornings and MCR nights. The Sicily trip will forever be a highlight, despite the damage to my reputation on a popular holiday letting website. To my housemates for the last two years: Alex, Sara, Sarah, Steph and Theo. Thank you for all the kitchen time, the parties and the *Traitors*. I'm so glad that we didn't get those LPS houses.

Occasionally, I even manage to maintain a life outside Cambridge. Shoutout to "Cudby's school friends" – David, Evan, James, John, Lewis, Matt, Rob, Ronan, and Tom – for over a decade of fun. Maybe one day you'll even talk to my other friends. Also to Verity for being a great friend whom I don't see nearly often enough.

Massive thanks to my cat, Toastie, for keeping my laptop warm and checking my keyboard still works when I was writing this thesis.

Mhairi, I couldn't ever express how grateful I am for you and your support. Love you.

Table of contents

List of figures **xv**

List of tables **xvii**

Nomenclature **xix**

0.1 Acknowledgement of collaboration 1

1 Introduction **3**

1.1 Thesis summary 5

1.2 Quantum mechanics 5

1.2.1 Fundamental particles 5

1.2.2 Fermionic operators 5

1.2.3 Majorana operators 6

1.3 Quantum computing 7

1.3.1 Quantum circuits 7

1.3.2 Pauli and Clifford operators and the Clifford hierarchy 7

1.3.3 Magic states, stabilizer rank and classical simulation 8

1.3.4 Matchgates, Gaussian operators and Gaussian states 9

1.3.5 Cost and complexity in (quantum) computing 13

1.3.6 Quantum algorithms and subroutines 14

2 A brief review of computational genomics and pangenomics **21**

2.1 Core concepts in genomics 21

2.2 DNA sequencing and sequence assembly 22

2.3 Pangenomes and pangenomics 23

2.4 Sequence alignment and string selection problems 24

2.5 Quantum computing in genomics 24

3 Simulation and learning of matchgate circuits 27
3.1 Simulating matchgate circuits . . . 28
3.1.1 An explicit set of constraints for Gaussian states . . . 28
3.1.2 Triples of Gaussian states . . . 33
3.1.3 An upper bound on the fermionic Gaussian fidelity . . . 35
3.1.4 Gaussian decompositions of magic states . . . 36
3.2 Learning Gaussian operations . . . 41
3.2.1 Problem setup and result . . . 41
3.2.2 Finding unsigned entries of the orthogonal matrix . . . 43
3.2.3 Fixing signs up to row-wise sign errors . . . 44
3.2.4 Fixing row-wise sign errors . . . 46
3.2.5 Complexity of learning Gaussian operations . . . 46
3.3 The matchgate hierarchy . . . 47
3.3.1 Defining the matchgate hierarchy . . . 47
3.3.2 Learning matchgate hierarchy operations . . . 48

4 Parameterized quantum algorithms for closest string problems 53
4.1 String processing problems . . . 55
4.2 Quantum algorithms for the closest string problem . . . 56
4.2.1 A trivial quantum search algorithm . . . 56
4.2.2 A quantum random walk algorithm for the CSP . . . 58
4.2.3 A Quantum dynamic programming algorithm for the CSP . . . 61
4.3 A quantum algorithm for the closest substring problem . . . 62

5 Pangenome-guided sequence assembly via binary optimisation 63
5.1 Pangenome-guided assembly as a QUBO problem . . . 66
5.1.1 Pangenome-guided assembly as an optimisation . . . 66
5.1.2 Optimisation to QUBO . . . 67
5.1.3 Optimisation to HUBO . . . 70
5.1.4 Binary to adiabatic optimisation . . . 71
5.1.5 Evaluation methods . . . 72
5.2 Classical approaches . . . 73
5.2.1 Pangenome-guided assembly versus *de novo* assemblers . . . 73
5.2.2 Comparing optimisation approaches to *pathfinder* . . . 73
5.3 Quantum annealing . . . 77
5.4 Quantum adiabatic optimisation algorithms . . . 77
5.4.1 QAOA applied to the QUBO formulation . . . 79

5.4.2 QAOA applied to the HUBO formulation 82
5.5 Discussion and future work 86

6 Conclusions **89**

Appendix A Simulating matchgate circuits **107**
A.1 Triples of Gaussian states 107
A.2 An ϵ-net for the Gaussian states 109
A.3 Symmetry-restricted Gaussian rank of two copies of a magic state . . . 113
A.3.1 $a_0^i \neq 0$, $i = 1, 2, 3$ 114
A.3.2 $a_0^3 = 0,\ a_0^i \neq 0$, $i = 1, 2$ 115
A.3.3 $a_0^2 = a_0^3 = 0,\ a_0^1 \neq 0$ 116

Appendix B Learning Gaussian operations **117**
B.1 Bounding the error of learning signs of Q 117
B.2 Bounding the distance between M_Q and $M_{Q'}$ 118

Appendix C Quantum search via random walks **121**
C.1 Markov chains, eigenvalue gaps and congestion 121
C.2 A quantum search algorithm 124
C.3 A Markov chain on a perfect tree 124

Appendix D Bespoke transpilation methods for HUBO-QAOA **129**
D.1 The main loop 131
D.2 Chain selection, interaction computing and chain ending logic 131
D.2.1 Superset and Subset 131
D.2.2 ComputeInteraction 132
D.2.3 EndChain 132

List of figures

1.1 The T-gadget 9
1.2 A stabilizer-rank-based classical simulation procedure 10
1.3 The SWAP-gadget 13
1.4 Schematic representation of QAOA 19

2.1 A pangenome graph 23

3.1 A Gaussian-rank-based classical simulation procedure 37

5.1 The pangenome-guided sequence assembly procedure 65
5.2 Converting a graph-traversal problem to a QUBO problem 69
5.3 Performance of classical QUBO solvers 76
5.4 Performance of quantum annealing QUBO solver 78
5.5 Heavy-hex edge colouring 80
5.6 Performance of QAOA simulation 81
5.7 Performance of QAOA hardware experiment 82
5.8 Sample density of optimised QAOA circuit 83
5.9 Bespoke HUBO-QAOA transpilation 85

D.1 An interaction chain for transpilation 130

List of tables

3.1 Constraints on an 8-qubit, symmetry-restricted Gaussian state 40

5.1 Performance of pangenome-guided assembly 74
5.2 Performance of classical QUBO solvers 75
5.3 Performance of quantum annealing QUBO solver 77
5.4 Bespoke HUBO-QAOA circuit compilation 86

A.1 Constraints on an 8-qubit, symmetry-restricted Gaussian state 114

Nomenclature

Roman Symbols

$\mathcal{C}$ The Clifford group

$\mathcal{G}$ The matchgate group

$G(A, B)$ A matchgate acting on the even-parity subspace as A and the odd-parity subspace as B with $\det(A) = \det(B)$

G_n The set of Gaussian states on n qubits

$\mathbb{1}(T)$ The indicator function for a Boolean expression T, equal to 1 when T is true and 0 otherwise

M_Q A (fermionic) Gaussian operator whose action is defined by the special orthogonal matrix Q

$\mathcal{P}$ The Pauli group

Greek Symbols

$\Gamma(\rho)$ The correlation matrix of a state ρ

γ_μ The μ^{th} Majorana operator for a system of n fermions, $\mu \in [2n]$

γ_S The Majorana monomial formed by the ordered product of the Majorana operators indexed by S

$A \Delta B$ The symmetric difference of two sets

Quantum States

$|M\rangle$ The canonical Gaussian magic state $|M\rangle = (|0000\rangle + |1111\rangle)/\sqrt{2}$

Strings

$d_H(s,\, s')$ The Hamming distance between two strings

$d_L(s,\, s')$ The Levenshtein distance between two strings

Σ A finite alphabet; for bioinformatics applications, usually the 4-element alphabet $\{A,\, C,\, G,\, T\}$

Σ^* The set of strings over Σ of any length

$\Sigma^{\leq n}$ The set of strings over Σ of length no more than n

Σ^n The set of strings over Σ of length n

s A string i.e. a finite sequence from an alphabet Σ

$s[i,\, j]$ The substring of s from i to j, inclusive

Other Symbols

$[n]$ The set $\{1, 2, \ldots, n\}$

$[A,\, B]$ The commutator $[A,\, B] = AB - BA$.

$\{A,\, B\}$ The anti-commutator $\{A,\, B\} = AB + BA$.

Acronyms / Abbreviations

CSP The Closest String Problem: given a set of strings, find a string that has distance at most d to any input

CSSP The Closest Substring Problem: given a set of strings, find a string that has distance at most d to some substring of any input

HUBO Higher-order Unconstrained Binary Optimisation

NISQ Noisy Intermediate-Scale Quantum

QAOA The Quantum Approximate Optimisation Algorithm

QUBO Quadratic Unconstrained Binary Optimisation

0.1 Acknowledgement of collaboration

Collaboration with peers has led to many fruitful ideas and results presented in this thesis. My primary collaborator has always been my supervisor, Sergii Strelchuk (SS). The proofs in this thesis were all derived by me. I produced all figures presented, including the code used to generate them. All code used to obtain numerical results was written by me, unless otherwise stated in the text.

Chapter 3 This chapter is based on material from [1, 2], collaborations with SS. The concept of studying magic and rank-based simulation of matchgate circuits was SS's; I derived the results and implemented all code. The idea to formulate a matchgate hierarchy and to study learning fermionic Gaussian operations was SS's. I derived the hierarchy definition and the learning algorithms.

Chapter 4 This chapter is based on material from [3]. The work was done as part of the Quantum Pangenomics (QPG) project in the Wellcome Leap Quantum4Bio competition. Useful discussions with many team members inspired the idea to study the fine-grained complexity of classically hard problems, most notably with SS, Richard Durbin (RD), and James Bonfield (JB). The choice to study closest string problems and the algorithms presented are exclusively my own work.

Chapter 5 The content of this chapter comes from [4], a collaboration with SS, RD, JB, and Chenxi Zhou (CZ) as part of the QPG project. The conception of pangenome-guided sequence alignment as an optimisation task is due to RD and CZ. The idea to map to QUBO was my own. JB led the generation of input data, as well as classical pre- and post-processing, and benchmarked existing classical state-of-the-art bioinformatics tools. I performed testing of both classical and quantum annealing solvers on QUBO problems. I mapped the QUBO problems to QAOA circuits and optimised those circuits. I also mapped the optimisation problems to HUBO formulations and wrote the custom transpilers to improve the circuit depth.

Chapter 1

Introduction

Quantum computing is an emerging technology that leverages the physical laws governing matter and energy at atomic scales to perform computational tasks, often more efficiently than any classical counterpart. In 1982, Richard Feynman [5] first proposed using quantum systems for computation; namely, for simulating physics. Since then, quantum algorithms have been proposed across a broad range of problem domains. An algorithm that arguably secured the field's long-term significance was proposed during the field's early years: Shor's 1994 integer factorisation algorithm [6]. The algorithm has two features of enduring scientific and historical importance. First, it achieves an exponential speed-up over the best classical algorithms that are currently known, requiring only $\mathcal{O}(\text{poly}(\log(N)))$ quantum gates to factorise an integer N, compared with $\mathcal{O}\left(\exp\left(\log(N)^{1/3}\right)\right)$ classical operations [7]. Second, and equally importantly, the algorithm has profound practical implications. Modern RSA encryption protocols rely on the computational difficulty of prime factorisation; quantum computers challenge this notion. A fully functioning, large-scale quantum computer could pose a significant threat to contemporary cybersecurity. Why, then, has this threat not been realised?

Current quantum devices remain limited in scale and are hindered by noise and operational errors. These challenges are practical and temporary rather than fundamental. Although the largest commercially available quantum computers presently contain fewer than 200 qubits, hardware roadmaps suggest this figure may exceed 1,000 within the next few years. Moreover, quantum error-correcting codes can reduce error rates by encoding logical qubits within collections of physical qubits, and it is an area of active research to find codes with lower qubit overheads [8, 9]. The quantum threshold theorem (e.g., [10]) informally states that if physical error rates p are below some threshold value p_{th}, then logical error rates can be made arbitrarily small with at most a logarithmic overhead in the number of gates. The implementation of such

error-correction codes on large devices would herald the era of Fault-Tolerant Quantum Computing (FTQC). Predictions for the duration of this transition vary from as short as 5 years to 15 years and beyond.

Despite the current lack of fault tolerance, quantum computers can still make meaningful contributions to computational science. Drawing inspiration from classical computing, we see a significant surge in interest in artificial intelligence and machine learning. The success of these fields rests not on rigorous proofs but on heuristic algorithms, empirical performance and resource-efficient optimisations. Similarly, quantum algorithm developers can focus on heuristic methods in key application domains to establish the near-term utility of quantum computing. The limited circuit width and depth achievable on current hardware are characteristic of the so-called Noisy Intermediate Scale Quantum (NISQ) era. A significant research direction, therefore, is to develop algorithms compatible with NISQ-era devices and identify applications that operate at industry-relevant scales. Key targets include machine learning [11–13], quantum chemistry [14–16] and life sciences [17–20].

This goal poses a fundamental challenge: how can heuristic quantum algorithms be developed without suitable hardware for testing? Moreover, what role should theoretical research play in bridging the gap between algorithm design and experimental implementation? This thesis addresses these challenges through three complementary approaches.

First, it examines simulation and learning tasks that are essential for verifying the performance of emerging quantum computers and for probing the boundary between classical and quantum computational power in chapter 3.

Second, it develops resource-efficient quantum algorithms that exploit problem structure to minimise circuit requirements. We explore such algorithms in chapter 4 for specific string-processing problems relevant in both bioinformatics and text analysis.

Finally, it adopts a holistic perspective on NISQ-era algorithm design, grounding theoretical development in real-world computational needs. In chapter 5, using pangenome-guided sequence assembly as a case study, we collaborate with bioinformatics researchers to provide the first formal treatment of this emerging problem and benchmark both classical and quantum approaches to assess the potential for genuine quantum utility.

1.1 Thesis summary

In the remainder of this chapter, we give a brief overview of the concepts in quantum mechanics and quantum computing used in the rest of this thesis. In chapter 2, we introduce the genomic and pangenomic concepts that are used in chapters 4 and 5. Chapters 3 to 5 contain the main results of this work. In chapter 3, we discuss simulation and learning of matchgate-based quantum computers. Chapter 4 introduces a suite of algorithms for closest string problems, fundamental string processing tasks with essential applications in bioinformatics. Chapter 5 discusses new classical and quantum optimisation approaches to pangenome-guided sequence alignment, a bleeding-edge bioinformatics task. Finally, we conclude and discuss future research avenues in chapter 6.

1.2 Quantum mechanics

Quantum mechanics is the physical theory that explains the behaviour of particles at subatomic scales where classical physics no longer applies. Our understanding of this theory provides the foundation for key applications such as quantum chemistry, condensed-matter physics and quantum information science.

1.2.1 Fundamental particles

Fundamental particles are indistinguishable from each other. Consider a two-particle system initially in the state $|x_1, x_2\rangle$ and an exchange operator $\hat{P}$ that swaps the two particles. Applying $\hat{P}$ twice gives

$$\hat{P}^2 |x_1, x_2\rangle = \hat{P} |x_2, x_1\rangle = |x_1, x_2\rangle . \tag{1.1}$$

Hence $\hat{P}^2 = I$ and so $\hat{P}$ has eigenvalues ± 1. Particles in the $+1$ eigenspace are *bosons* and particles in the -1 eigenspace are *fermions*. The Pauli exclusion principle states that two or more fermions cannot occupy the same quantum state, which is a consequence of their anti-symmetry [21].

1.2.2 Fermionic operators

We now restrict attention to systems containing up to n fermions. The basis states of such a system are called *modes*. The *fermionic creation operator* $a_j^\dagger$, for $j \in [n] :=$

$\{1, \ldots, n\}$, creates a fermion in the j^{th} mode, and its Hermitian conjugate a_j is the *fermionic annihilation operator* that removes a fermion from that j^{th} mode. Due to the Pauli exclusion principle, $a_j^\dagger$ annihilates any state containing a fermion in mode j, while a_j annihilates any state without one. These operators obey the Canonical Anti-commutation Relations (CARs):

$$\{a_i,\, a_j^\dagger\} = \delta_{ij} I, \quad \{a_i, a_j\} = \{a_i^\dagger, a_j^\dagger\} = 0, \tag{1.2}$$

where $\{A,\, B\} = AB + BA$ is the anti-commutator.

1.2.3 Majorana operators

It is often convenient to work with *Majorana operators*, which are the fermionic analogue of position and momentum operators. For a system of n fermions we define $2n$ Hermitian operators:

$$\gamma_{2l-1} \coloneqq a_l + a_l^\dagger, \quad \gamma_{2l} \coloneqq -i(a_l - a_l^\dagger), \quad l \in [n]. \tag{1.3}$$

These operators satisfy the anti-commutation relations

$$\{\gamma_\mu,\, \gamma_\nu\} = 2\delta_{\mu\nu} I, \quad \mu, \nu \in [2n], \tag{1.4}$$

forming a *Clifford algebra* whose elements are given by linear combinations of products of the γ's. Since $\gamma_\mu^2 = I \;\forall\, \mu$, a general element in the algebra is a polynomial of degree at most $2n$, linear in each γ_μ.

For an index set $S \subseteq [2n]$, γ_S is the *Majorana monomial* formed by the ordered product of the Majorana operators with indices in S. The 2^{2n} Majorana monomials form a basis of linear operators on n qubits.

When multiplying Majorana monomials together, we will use the notation

$$\gamma_S \gamma_{S'} = (\pm)\gamma_{S \Delta S'}, \tag{1.5}$$

where $\cdot\Delta\cdot$ denotes the symmetric difference of two sets and the sign accounts for the ordering of anti-commuting operators.

We may choose the Jordan-Wigner representation [22], in which

$$\gamma_{2l-1} = \left(\prod_{i=1}^{l-1} Z_i\right) X_l \quad \text{and} \quad \gamma_{2l} = \left(\prod_{i=1}^{l-1} Z_i\right) Y_l, \tag{1.6}$$

where Z_i acts as the single-qubit Pauli Z on the i^{th} qubit and the identity elsewhere, and similarly for X_i and Y_i (see section 1.3.2).

1.3 Quantum computing

The fundamental unit of quantum information is the *qubit*, a two-level quantum system with Hilbert space $\mathcal{H}_2 \coloneqq \{a\,|0\rangle + b\,|1\rangle : a,\, b \in \mathbb{C},\ |a|^2 + |b|^2 = 1\}$. Quantum computers operate on *registers* of n qubits with state space $\mathcal{H}_2^{\otimes n} = \text{span}_{\mathbb{C}}\,\{|\mathbf{x}\rangle : \mathbf{x} \in \mathbb{Z}_2^n\}$. The set $\{|\mathbf{x}\rangle : \mathbf{x} \in \mathbb{Z}_2^n\}$ is known as the *computational basis*.

1.3.1 Quantum circuits

The most common framework for discussing quantum algorithms is the circuit model, analogous to classical logic circuits. A circuit comprises a sequence of quantum gates, which are unitary operations acting on one or more qubits, and measurements, which irreversibly project quantum states onto eigenstates of an observable and yield classical outcomes.

In practice, quantum circuits are built from small sets of gates that still densely generate the set of quantum operations, known as *universal gate sets*. These may be the gates native to a specific hardware setup, or simply a convenient choice, such as the "Clifford + T" set discussed in 1.3.2. The Solovay-Kitaev Theorem [23] implies that any quantum circuit consisting of arbitrary few-qubit gates can be efficiently approximated by any universal gate set with only logarithmic overhead in the number of gates.

Three metrics usually capture circuit complexity. The *width* is the number of qubits; the *size* is the number of gates; and the *depth* is the number of sequential layers in the circuit when gates acting on disjoint qubits are run in parallel.

1.3.2 Pauli and Clifford operators and the Clifford hierarchy

The *Pauli gates* X, Y, Z are single qubit unitaries with eigenvalues ± 1.

- Z acts as a *phase-flip*, leaving $|0\rangle$ unchanged and mapping $|1\rangle \mapsto -|1\rangle$.
- X acts as a *bit-flip*, interchanging $|0\rangle$ and $|1\rangle$.
- $Y = iXZ$.

The *Pauli group* on n qubits, $\mathcal{P}$, is the set of gates given by

$$\mathcal{P} \coloneqq \{sP_1 \otimes P_2 \otimes \ldots \otimes P_n : s \in \{\pm 1, \pm i\},\, P_i \in \{I, X, Y, Z\}\}\,. \tag{1.7}$$

The *Clifford group* on n qubits, $\mathcal{C}$, is the normaliser of the Pauli group:

$$\mathcal{C} \coloneqq \left\{C \in U(2^n) : C\mathcal{P}C^\dagger \subseteq \mathcal{P}\right\} / \, U(1). \tag{1.8}$$

It is finite, generated by the set $\{H,\, S,\, CX\}$ and is not a universal gate set. Indeed, the celebrated Gottesman-Knill theorem [24] states that any quantum circuit consisting of only Clifford gates and computational basis measurements can be efficiently classically simulated. Thus, informally, Clifford circuits possess only "classical" computing power. States produced by Clifford circuits acting on $|0\rangle^n$ are known as *stabilizer states*.

Extending the gate set with the non-Clifford T gate, $T = \mathrm{diag}(1,\, e^{i\pi/4})$ yields the universal "Clifford + T" set. Due to the relative difficulty of implementing Clifford and non-Clifford gates, especially on certain error-correcting codes [25], Clifford gates are usually treated as free operations, whereas T gates are an expensive resource.

More generally, the *Clifford hierarchy* [26] defines an infinite family of nested sets of gates $\mathcal{C}_k$:

$$\mathcal{C}_1 \coloneqq \mathcal{P}, \quad \mathcal{C}_k \coloneqq \left\{C \in U(2^n) : C\mathcal{P}C^\dagger \subseteq \mathcal{C}_{k-1}\right\} / \, U(1). \tag{1.9}$$

The first two levels of the hierarchy correspond to the Pauli and Clifford groups, respectively, and the universality-enabling T gate lies in $\mathcal{C}_3$. The hierarchy provides a natural framework for studying fault-tolerance [27–29], magic state distillation [30] and learning [31].

1.3.3 Magic states, stabilizer rank and classical simulation

Instead of adding new gates, universal computation can also be achieved by supplementing Clifford circuits with special input states known as *magic states* [32]. For example,

$$|T\rangle \coloneqq \frac{|0\rangle + e^{i\pi/4}\,|1\rangle}{\sqrt{2}} \tag{1.10}$$

enables the implementation of a T gate using a small, adaptive Clifford circuit. This circuit is known as a *T-gadget* and is given in fig. 1.1. Replacing non-Clifford gates with Clifford operations and magic-state ancillas forms the basis of many scalable quantum-computing architectures [33, 34]. Since Clifford gates can be performed fault-

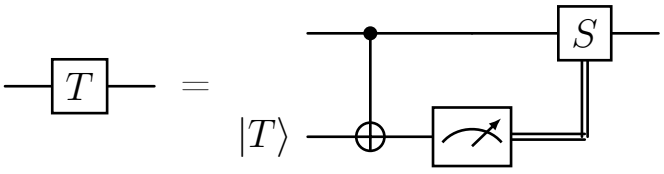


Fig. 1.1 The T-gadget, which implements a T gate using an ancilla in the state $|T\rangle$ and an adaptive Clifford circuit.

tolerantly on logical qubits encoded in specific quantum error-correcting codes [25], such magic state approaches have the potential to scale into the FTQC era.

The combination of the Solovay-Kitaev and the Gottesman-Knill theorems suggests a method for simulating arbitrary quantum processes whose complexity scales with a measure of "non-Cliffordness" first introduced by Bravyi, Smith & Smolin [35]. First, any quantum circuit on n qubits may be re-expressed in the "Clifford + T" gate set. This incurs only logarithmic (in n) overhead by the Solovay-Kitaev theorem, assuming at most a constant number of qubits were initially involved in each gate. Suppose that there are t T gates in the resulting circuit. This circuit can be "gadgetised," where each T gate is replaced by a T-gadget with its corresponding ancilla. The resulting circuit is an adaptive Clifford circuit acting on $|0\rangle^n |T\rangle^t$. The Gottesman-Knill theorem can be applied after expressing the state $|T\rangle^t$ as a sum of stabilizer states. For each term in this sum, the process under consideration is an adaptive Clifford circuit acting on a stabilizer state input, which is classically efficiently simulable. A sketch of this procedure is given in fig. 1.2. The cost of the simulation is dominated by the number of terms in the decomposition of $|T\rangle^t$, a quantity known as the *stabilizer rank*. A naive decomposition of each $|T\rangle$ into two terms gives a cost of 2^t. Decompositions with fewer terms reduce the cost to $2^{\alpha t}$ for $\alpha \lesssim 0.396$ [36].

1.3.4 Matchgates, Gaussian operators and Gaussian states

In addition to the Clifford group, other gate families also permit efficient classical simulation. One crucial such class of gates is the group of *matchgates*, introduced and proven to be classically simulable by Valiant [37]. These gates were shown to correspond to a physical model of non-interacting fermions [38], a classically solvable physical system, giving an alternative proof of their simulability. A matchgate is a two-qubit, nearest-neighbour gate $G = G(A, B)$ that acts on the even-parity subspace as A and on the odd-parity subspace as B, with $\det(A) = \det(B)$. Its matrix representation

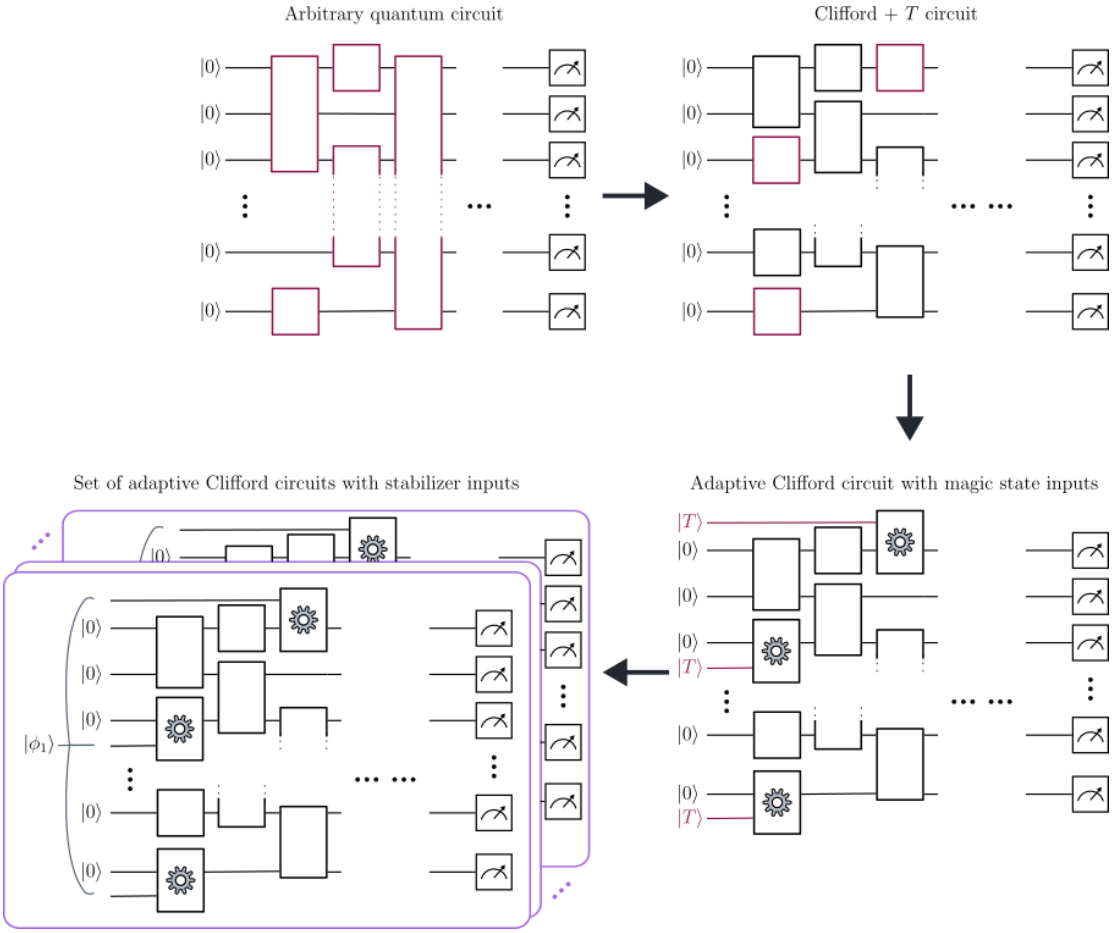


Fig. 1.2 A sketch of a stabilizer-rank-based classical simulation procedure. Throughout, the colour red denotes a non-free resource, namely non-Clifford gates or magic state inputs. (Upper-left) The starting point is a generic quantum circuit containing arbitrary gates. (Upper-right) The circuit is expressed in the "Clifford + T" gate set. This generally increases the circuit depth, but the increase is at most logarithmic if the original gates act on at most a constant number of qubits. (Lower-right) Each T gate is replaced by a T-gadget, a small, adaptive, Clifford circuit that acts on the original qubit and consumes an ancilla qubit in the initial state $|T\rangle$. Notably, all gates in the circuit are now Clifford. (Lower-left) The state $|T\rangle^t$ is decomposed into an exponentially large sum of stabilizer states. The (adaptive) Clifford circuit acting on any term in this sum can be classically simulated.

takes the form

$$G(A,\, B) \coloneqq \begin{pmatrix} p & 0 & 0 & q \\ 0 & w & x & 0 \\ 0 & y & z & 0 \\ r & 0 & 0 & s \end{pmatrix}, \tag{1.11}$$

$$\text{where } A = \begin{pmatrix} p & q \\ r & s \end{pmatrix},\ B = \begin{pmatrix} w & x \\ y & z \end{pmatrix},\ \text{and } \det(A) = \det(B). \tag{1.12}$$

Notably, if a state has definite even (respectively, odd) parity – that is, when expressed in the computational basis, only states with an even (odd) number of 1s appear – the action of a matchgate preserves this property. Hence, matchgates are said to be *parity preserving*.

The group of unitary operations implementable by matchgates is the *matchgate group* $\mathcal{G}$. Members M of the matchgate group are called *(fermionic) Gaussian operators*. Gaussian operators correspond to exponentials of fermionic quadratic Hamiltonians H_f:

$$M = e^{iH_f}, \quad H_f \coloneqq i \sum_{\mu,\,\nu} h_{\mu\nu} \gamma_\mu \gamma_\nu, \tag{1.13}$$

where $h = (h_{\mu\nu})$ is, without loss of generality, a real anti-symmetric matrix.

It can be shown [39] that Gaussian operators map Majorana modes linearly under conjugation:

$$M \gamma_\mu M^\dagger = \sum_\nu Q_{\mu\nu} \gamma_\nu, \tag{1.14}$$

for a special orthogonal matrix $Q = (Q_{\mu\nu}) \in SO(2n)$ given by $Q = \exp(4h)$.

As a direct consequence of eq. (1.14), we can write the behaviour of Majorana monomials under conjugation as:

$$M \gamma_S M^\dagger = \sum_{|S'|=|S|} \det(Q|_{S,\,S'}) \gamma_{S'}, \tag{1.15}$$

where the sum is over subsets of $[2n]$ of equal size to S, and $Q|_{S,\,S'}$ denotes the restriction of Q to rows indexed by S and columns indexed by S'. In particular, the number of Majorana operators is preserved in each term.

Since the Majorana monomials form a basis of linear operators on n qubits, M is fully determined by eq. (1.15) up to a global phase. We therefore label Gaussian operations by their corresponding orthogonal matrix, writing $M = M_Q$.

The closure of Gibbs states of quadratic Hamiltonians gives the set of (possibly mixed) *Gaussian states*:

$$\rho(H_f) \coloneqq \frac{e^{-H_f}}{\mathrm{Tr}(e^{-H_f})}. \tag{1.16}$$

For any density matrix ρ, define the *correlation matrix* $\Gamma(\rho) = (\Gamma(\rho)_{jk})$ by

$$\Gamma(\rho)_{jk} \coloneqq \frac{i}{2}\mathrm{Tr}([\gamma_j, \gamma_k]\rho), \tag{1.17}$$

where $[A, B] \coloneqq AB - BA$ is the commutator.

$\Gamma(\rho)$ is easily seen to be real and anti-symmetric. For a Gaussian state $\rho(H_f)$, it can be checked that $\Gamma(\rho(H_f)) = \tan(2h)$ where h is as defined in eq. (1.13). Since tan is injective for anti-symmetric matrices, Gaussian states are uniquely defined by their correlation matrices, thus explaining the nomenclature.

Pure Gaussian states correspond to the set of states produced by applying some Gaussian operation (or, equivalently, some matchgate circuit) to the initial state $|0\rangle^n$. One can also show an alternative characterisation of Gaussian states [40], where an n-qubit state $|\psi\rangle$ is Gaussian if and only if

$$\Lambda\,|\psi\rangle\,|\psi\rangle = 0, \quad \text{where } \Lambda \coloneqq \sum_{\mu=1}^{2n} \gamma_\mu \otimes \gamma_\mu. \tag{1.18}$$

Equation (1.18) has historically been the primary operational definition of Gaussian states.

The simulable nature of the matchgate group is closely tied to the restriction that matchgates act on adjacent qubits. Indeed, the same set of gates acting on either nearest- or *next*-nearest-neighbours generates a universal gate set [39]. Thus, in some sense, *non-locality* of gates is a resource for matchgate-based computing. Equivalently, the "Matchgate + SWAP" gate set is universal for quantum computing.

The notion of magic states also arises in the theory of matchgate circuits. It was shown by Hebenstreit et al. [41] that any state with definite even or odd parity that is non-Gaussian is a magic state for matchgate circuits. Notably, all states with three or fewer qubits are Gaussian, so the smallest magic state has four qubits; a canonical example is the four-qubit GHZ state

$$|M\rangle \coloneqq \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle). \tag{1.19}$$

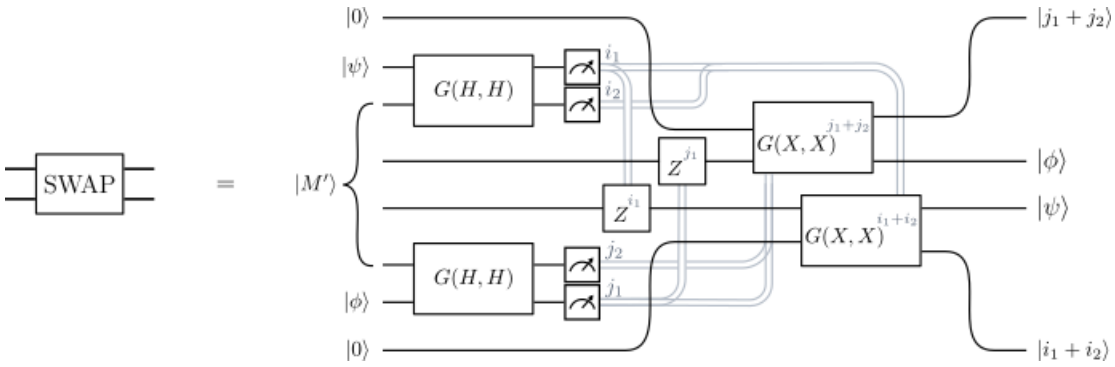


Fig. 1.3 The SWAP-gadget, which implements a SWAP gate using a magic state $|M'\rangle$ and an adaptive matchgate circuit. The additional ancilla qubits, initialised in state $|0\rangle$, are swapped in and out using $G(\pm Z, X)$ gates (not shown).

A gadget for implementing the SWAP gate is given in fig. 1.3. It consumes a Gaussian magic state $|M'\rangle := \frac{1}{2}(|0000\rangle + |0101\rangle + |1010\rangle + |1111\rangle)$ that differs from $|M\rangle$ by a small matchgate circuit. It also requires two ancillas initialised in state $|0\rangle$. They are swapped in and out as needed using $G(\pm Z, X)$ gates, which act as SWAP when one of the qubits is known to be in state $|0\rangle$ or $|1\rangle$, respectively.

1.3.5 Cost and complexity in (quantum) computing

Classical complexity theory

In classical computation, problems are classified by the time and space resources required to solve them as the input size n grows. Such classification provides a broad concept of hardness: the existence of a polynomial-time algorithm is often used to classify a problem as "easy"; conversely, a proof that exponential time is required, or simply a lack of an efficient algorithm, classifies a problem as "hard." Complexity theory formalises these ideas by splitting problems into *complexity classes* such as P, the set of decision problems that can be solved classically in polynomial time, and NP, problems whose solution can be *verified* classically in polynomial time. NP includes decision versions of many classic problems in computer science, such as `3SAT`, `Graph Colouring` and `Travelling Salesman`.

However, such broad classes often fail to convey the practical difficulty of solving problems. There are subsets of problems that have some structure which can be exploited to make them easier to solve. The field of *parameterized complexity* provides a framework to describe algorithms for problems when some feature k other than input size is assumed to be small. Problems solvable in time $\mathcal{O}(\mathsf{poly}(n) \cdot f(k))$ belong to the class FPT (for fixed-parameter tractable). If, instead, the best algorithm has runtime

$\mathcal{O}\big(n^{g(k)} \cdot f(k)\big)$, the problem lies in XP (for slicewise polynomial). Clearly FPT $\subseteq$ XP, and the inclusion is strict by diagonalisation.

Quantum complexity theory

For quantum complexity theory, the situation is even more subtle. First, quantum computers are inherently probabilistic machines whose outputs result from random measurements. While deterministic quantum algorithms do exist – the Deutsch-Jozsa algorithm [42] being a notable example – the vast majority of quantum algorithms succeed only with some probability less than unity. This prompts the definition of the most commonly used quantum complexity class, BQP, which refers to bounded-error quantum polynomial time. Specifically, BQP is the set of decision problems that can be solved in polynomial time on a quantum computer with an error probability of at most 1/3 on each instance. The choice of 1/3 is arbitrary: a constant number of repetitions of the algorithm with a majority vote suffices to achieve any desired error probability greater than zero by a standard Chernoff bound.

Implicit in the definition of BQP is a requirement that the quantum algorithm is described by gates acting on at most a constant number of qubits. If arbitrarily large gates were allowed, then all quantum algorithms could trivially be described by a single n-qubit unitary. According to the Solovay-Kitaev theorem, switching between different choices incurs only logarithmic overhead and has no complexity-theoretic implications for polynomial-time problems.

Such asymptotic classes provide limited guidance for current NISQ devices. The limitations of classical complexity theory still hold, and BQP does not provide a nuanced view of the relative difficulty of its members. More pertinent, however, are the difficulties involved in actually running algorithms on current quantum hardware. Error rates are high, and two-qubit gates dominate noise budgets. Therefore, in the NISQ era, it is reasonable to adopt a more pragmatic approach to algorithm complexity, emphasising two-qubit gate depth and count, as well as empirical performance on actual problem instances.

1.3.6 Quantum algorithms and subroutines

Many quantum algorithms employ standard subroutines. These are often formulated within the quantum-query model [43, 44], where an *oracle* O_f for a function $f : [M] \to [N]$ implements the transformation $|x\rangle\,|b\rangle \mapsto |x\rangle\,|b + f(x) \mod N\rangle$ for $x \in [M]$, $b \in [N]$. We now discuss the quantum subroutines we use in this thesis.

Unstructured search

A key quantum subroutine is the Grover search algorithm, which finds a marked element from a set of size n in $\tilde{\mathcal{O}}(\sqrt{n})$ queries:

Theorem 1.3.1 (Grover search [45])**.** *There is a quantum algorithm that, given a function $f : [n] \to \{0, 1\}$, finds an index i with $f(i) = 1$ or reports that no such index exists. The algorithm uses $\tilde{\mathcal{O}}(\sqrt{n})$ queries to an oracle for f.*

A use of this algorithm to find the minimum (equivalently, maximum) of a set of n values in $\tilde{\mathcal{O}}(\sqrt{n})$ queries was proposed by Dürr and Høyer:

Theorem 1.3.2 (Quantum minimum/maximum finding [46])**.** *There is a quantum algorithm that, given a function $f : [n] \to \mathbb{R}$, finds the index i such that $f(i) \leq f(j)\ \forall j$ (equivalently, $f(i) \geq f(j)\ \forall j$). The algorithm uses $\tilde{\mathcal{O}}(\sqrt{n})$ queries to an oracle for f.*

Search via random walk

A second quantum search routine, due to Magniez et al. [47], proceeds via a random walk. It is a stronger version of the algorithms provided by Ambainis [48] and Szegedy [49]. A complete introduction to this result, including the requisite background on Markov chains, is provided in appendix C. We briefly state the main results here.

Suppose we have a set of states Ω and a subset of $M \subset \Omega$ of *marked* states, and we wish to find some $x \in M$. Classically, we might consider a search algorithm as described by the pseudocode in algorithm 1. Assuming that the algorithm maintains a data structure D that associates data $D(x)$ to each state $x \in \Omega$, then the complexity of the algorithm rests on three quantities:

- **Setup** S: The cost of sampling $x \in \Omega$ according to the distribution X and constructing $D(x)$ from x.
- **Update** U: The cost of simulating a transition $x \to y$ according to P and of updating $D(x)$ to $D(y)$.
- **Checking** C: The cost of checking whether $x \in M$ using $D(x)$.

Then there is a classical search algorithm as described by theorem 1.3.3:

Theorem 1.3.3 (Classical Search via Random Walk [47])**.** *Let $\delta > 0$ be the eigenvalue gap of an ergodic, symmetric Markov chain. Let $|M|/|\Omega| \geq \epsilon > 0$ whenever M is non-empty. For a uniform initial distribution $X \sim U[\Omega]$, algorithm 1 succeeds with high probability if $t_1 = \mathcal{O}\left(\frac{1}{\delta}\right)$ and $t_2 = \mathcal{O}\left(\frac{1}{\epsilon}\right)$. The incurred cost is of the order*

$$S + \frac{1}{\epsilon}\left(\frac{1}{\delta}U + C\right). \tag{1.20}$$

Algorithm 1 Classical Search via Random Walk [47]

Require: Integers t_1, t_2, probability distribution X over Ω, Markov chain P on Ω.

```
x ← sample from X
for i = 1, ..., t_2 do
    if x ∈ M then
        return x
    else
        x ← t_1^th step of a simulation of P starting from x
return "No marked state exists"
```

Intuitively, t_1 is large enough that the state of the Markov chain is close to the stationary distribution after t_1 steps. Algorithm 1 samples from this distribution t_2 times, which is sufficient to succeed with high probability by the promise on ϵ.

For the quantum case, the data structure is captured by states belonging to an expanded Hilbert space $\mathbb{C}^{\Omega_D}$ where $\Omega_D = \{(x, D(x)) : x \in \Omega\}$. For conciseness, we write $|x\rangle_D \equiv |x, D(x)\rangle$. For technical reasons, the search is conducted on walks along *edges* of the Markov chain, which have a Hilbert space of $\mathbb{C}^{\Omega_D \times \Omega_D}$.

Suppose we have a Markov chain P with stationary distribution π and transition probabilities $\{p_{xy} : x, y \in \Omega\}$. Let P^* be the time-reversed Markov chain with transition probabilities p^*_{xy}. The costs of the quantum search are given by:

- **(Quantum) Setup** S: The cost of constructing the state $\sum_x \sqrt{\pi_x} |x\rangle_D |0\rangle_D$.
- **(Quantum) Update** U: The cost of realising any of the unitary transformations

$$|x\rangle_D |0\rangle_D \mapsto |x\rangle_D \sum_y \sqrt{p_{xy}} |y\rangle_D ,$$
$$|0\rangle_D |y\rangle_D \mapsto \sum_x \sqrt{p^*_{yx}} |x\rangle_D |y\rangle_D ,$$

  or their inverses.

- **(Quantum) Checking** C: The cost of realising the conditional phase flip

$$|x\rangle_D |y\rangle_D \mapsto \begin{cases} -|x\rangle_D |y\rangle_D & \text{if } x \in M, \\ |x\rangle_D |y\rangle_D & \text{otherwise.} \end{cases}$$

Then there exists a quantum search algorithm as described by theorem 1.3.4.

Theorem 1.3.4 (Quantum Search via Random Walk [47])**.** *Let S, U and C respectively be the costs of setting up the required data structure, updating the data structure at*

each step of the walk, and applying a phase flip to elements $x \in M$ that are marked for the search.

Let $\delta > 0$ be the eigenvalue gap of a reversible, ergodic Markov chain P, and let $\epsilon > 0$ be a lower bound on the probability that an element chosen from the stationary distribution of P is marked whenever M is non-empty. Then, there is a quantum algorithm that, with high probability, determines if M is empty or finds an element of M, with a cost of order

$$S + \frac{1}{\sqrt{\epsilon}} \left(\frac{1}{\sqrt{\delta}} U + C \right). \tag{1.21}$$

This is a quadratic improvement in the dependence on δ and ϵ compared to theorem 1.3.3.

Quantum dynamic programming

First introduced by Ambainis et al. [50] on the Boolean hypercube, quantum dynamic programming was extended by Glos et al. [51] to the k-dimensional lattice graph with vertices in $\{1, 2, \ldots, n\}^k$. As with classical dynamic programming, the algorithm applies to problems that have the *optimal substructure* property: optimal solutions can be constructed from optimal solutions of sub-problems. The algorithm is presented for the "path in the hyperlattice" problem. Still, it applies equally to any classical dynamic programming problem, replacing the Grover search with quantum minimum/maximum finding as necessary. The quantum algorithm is recursive, using nested quantum searches at each recursive level. While the classical query complexity is $\tilde{\Theta}\left(n^k\right)$, the quantum algorithm has complexity $\tilde{\mathcal{O}}\big(T(n)^k\big)$ for some function $T(n)$ satisfying $n/e \leq T(n) < n$ for every n. Although this is not a polynomial speedup with respect to problem size, it still represents a non-trivial improvement in the algorithm's complexity. We state this result in theorem 1.3.5:

Theorem 1.3.5 (Quantum dynamic programming [51])**.** *There is a quantum algorithm that solves a problem with optimal substructure on a k-dimensional lattice with vertices in $\{1, 2, \ldots, n\}^k$ in time $\tilde{\mathcal{O}}\big(T(n)^k\big)$ for some $T(n) < n + 1$.*

Classical computing

We will sometimes rely on performing classical computing as a subroutine. In particular, we will want to implement oracles O_f for non-trivial classical functions f. It was shown by Bennett [52] that deterministic classical, irreversible algorithms may be efficiently converted to reversible ones. This was extended by Zuliani [53] to include nondeterministic and probabilistic computations. This general-purpose result allows for the efficient simulation of classical computation on a quantum computer.

Optimisation

Perhaps the most well-known NISQ-era algorithm is the Quantum Approximate Optimisation Algorithm (QAOA) [54], which outputs approximate solutions to combinatorial optimisation problems. QAOA is a variational algorithm wherein a quantum circuit containing parametrised gates is sampled and the parameters are updated as part of a classical optimisation loop – see fig. 1.4 for a schematic representation. When the optimisation converges, a final round of sampling is performed, and the best sample is returned as a solution. While substantial effort has been made to analytically prove the performance of QAOA [54–56], it is fundamentally an empirical algorithm, and its performance is best understood in the context of experimental results. Successful experiments have been run with up to 127 qubits [57–59]. When applied to problems with a local optimisation function, QAOA enjoys equal locality in its circuits and linear depth scaling with the number of objective terms. Combinatorial optimisation, therefore, represents a promising direction for quantum utility.

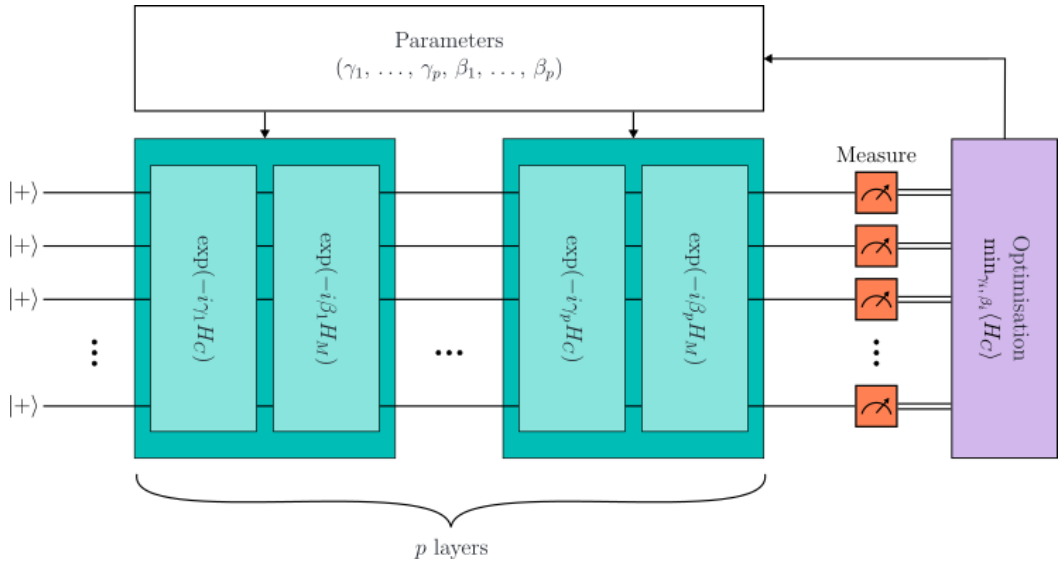


Fig. 1.4 A schematic representation of a QAOA procedure. The hyperparameter p determines the number of layers in the circuit. Each layer comprises two parts. The first part corresponds to a *cost* Hamiltonian H_C, defined by an underlying combinatorial optimisation problem, and the second to a *mixer* Hamiltonian H_M, usually chosen as the sum of a single-qubit X gate on each qubit. In layer i, the system's state evolves under H_C for a duration of γ_i and then under H_M for β_i. The initial state is an equal superposition over all computational basis states. After p layers, each qubit is measured, and the sample is fed into a classical optimiser that minimises the energy $\langle H_C \rangle$. The optimiser proposes new parameters for the next iteration until the energy converges. Measurements of the state prepared by the circuit with optimised parameters are then low-energy solutions to the optimisation problem with high probability.

Chapter 2

A brief review of computational genomics and pangenomics

2.1 Core concepts in genomics

Genomics is the study of an organism's entire genetic material, the genome. In contrast to genetics, which examines individual genes, genomics is a more holistic approach that encompasses the structure, evolution, and interplay of all genetic information. The genome consists of deoxyribonucleic acid (DNA), a complex molecule composed of four nucleotide bases – adenine (A), cytosine (C), guanine (G), and thymine (T) – in the famous double-helix structure. A linear sequence of these nucleotides, commonly represented as a string of letters from the alphabet $\{A, C, G, T\}$, encodes the biological instructions necessary for life.

Although nearly all cells within an organism share an identical genome, *genetic variation* arises between different individuals. These variations, or polymorphisms, can range from small-scale Single-Nucleotide Polymorphisms (SNPs), which involve the change of a single base, to large-scale Structural Variants (SVs), where large DNA segments are copied, deleted, moved, or reversed. Human DNA is fundamentally highly conserved, with each person sharing approximately 99.5% of their DNA with any other [60]. However, even the remaining fraction allows a multitude of possible genetic variations, many of which have critical biological or clinical implications.

For example, the Human Leukocyte Antigen (HLA) genes play a crucial role in the immune system. Specific variants of this region have been associated with increased susceptibility to autoimmune conditions such as type 1 diabetes, as well as infectious diseases, including malaria and tuberculosis. Understanding such variation requires

access to complete genomic sequences from many individuals, which can be obtained through DNA sequencing and subsequent sequence assembly.

2.2 DNA sequencing and sequence assembly

DNA sequencing is the process of determining the order of bases in an organism's genome. The first full-genome sequencing method was developed by Frederick Sanger in 1977 [61], producing random fragments of DNA, or *reads*, each approximately 500 bases long, with 99.99% accuracy. Modern next-generation sequencing (NGS) methods, first developed in the 1990s, are largely automated, yielding vast quantities of DNA data. The financial cost of sequencing has fallen dramatically – roughly six orders of magnitude lower over the past two decades [62].

To ensure full genome coverage, the sequencing procedure is repeated so that each section of DNA is, on average, read multiple times (e.g. 30x coverage). These fragments are then computationally stitched together to obtain a whole sequence in a process known as *sequence assembly*.

Sequence assembly approaches broadly fall into two categories:

- *De novo* assembly, which reconstructs genomes solely from read data; and
- *Reference-guided* assembly, which aligns reads to an existing pre-assembled genome.

De novo assemblers typically represent the relationships between reads as graphs. In the *De Bruijn* graph approach [63], nodes correspond to $k-1$-length substrings of reads, and edges connect reads that form consecutive substrings of length k. Assembly is then recast as a graph-theoretic problem that is NP-hard in general.

In contrast, reference-guided assemblers use knowledge of a reference genome, usually an individual from the same species. Since the reference is expected to be very similar to the new sequence, most reads can be mapped perfectly to the reference, allowing the genome to be reconstructed by following the path of matches along the reference.

Modern sequencing introduces formidable computational challenges. Large-scale NGS projects produce billions of fragments, vastly increasing the problem sizes that assemblers face. The 5,400-base $\Phi X174$ genome is trivial compared to the 3-billion-base-pair human genome. On such massive scales, *de novo* assemblies are expensive, requiring months of bespoke research effort and many hours of computer time. Reference-guided assembly is more tractable, but often suffers from *reference bias*, being likely to over-fit

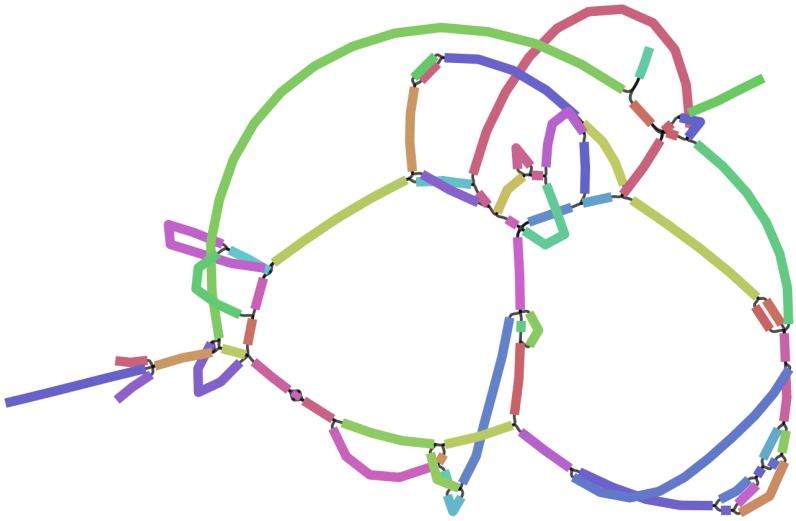


Fig. 2.1 An example of a pangenome graph. Each node contains DNA sequence data, with the node size proportional to the length of the corresponding sequence. Colours for visualisation only.

to the reference and to omit novel variations in the sample. This can be particularly detrimental in highly variable regions, such as the HLA genes, where a single reference sequence cannot capture all possible gene variants.

One approach to tackling reference bias while retaining the speed benefits of reference-guided assembly is to use a reference that accounts for variation within a population. The study of these data structures is *pangenomics*.

2.3 Pangenomes and pangenomics

Pangenomics aims to describe the complete genetic variation within a population, typically a species. Definitions of a *pangenome* vary; we will focus on *sequence-oriented* pangenomes that describe the SNPs and SVs across different individuals [64]. A standard representation is the *pangenome graph*, where graph nodes correspond to DNA segments and edges encode some notion of co-location. An example of a pangenome is given in fig. 2.1.

Pangenome graphs mitigate reference bias by incorporating known variation. Short reads aligned to a pangenome graph can categorise SVs with greater accuracy than traditional single-reference methods [65]. They have also been used for *variant genotyping*, which refers to identifying whether a variant exists on zero, one or both chromosomes in

diploid organisms – organisms that possess two complete sets of chromosomes, one from each parent. Algorithms such as those introduced by Ebler et al. [66] enable accurate detection of SVs in repetitive and duplicated regions where traditional short-read methods struggle.

In chapter 5, we develop a new application of pangenomes: given a set of short reads and a precomputed pangenome built from related individuals, we aim to reconstruct the genome of a new individual. We call this task *pangenome-guided sequence assembly*.

2.4 Sequence alignment and string selection problems

Once genomes are assembled, several possible next steps are available.

A key task is *sequence alignment*, in which sequences are arranged to reveal the regions of DNA shared between two or more samples. These shared regions may reflect functional or evolutionary relationships.

Alignment typically involves inserting gaps into one or more sequences to maximise the number of symbol matches. The number of gaps and mismatches in an optimal alignment between two strings is the *Levenshtein* or *edit distance* [67], so-called because it corresponds to the number of symbols that must be inserted, deleted or substituted from one sequence to transform it into the other. The original, formally correct, algorithm for pairwise alignment is due to Needleman and Wunsch [68] and runs in time $\mathcal{O}(n^2)$ for two strings of length n, using dynamic programming. In real-world applications, empirical average runtime is heavily favoured over worst-case performance, and modern heuristic algorithms achieve observed runtime of $\approx n^{1.1}$ [69]. The near-linear algorithm means that pairwise alignment is essentially "free" for modern computers, and there is no utility in quantum algorithms for this task.

In this thesis, we instead focus on the computational bottlenecks in classical bioinformatics. We therefore study string selection problems [70], a collection of string-processing tasks that take a set of strings as input and output a single string that captures some desired property of the set. Applications include drug design [71], motif finding [72] and primer design in polymerase chain reactions [73]. We discuss quantum algorithms for these problems in chapter 4.

2.5 Quantum computing in genomics

The integration of quantum computing into genomics is a rapidly developing field. Many tasks in computational biology – such as seed-and-extend alignment [74], *de novo*

assembly [20], bounded edit distance [19] and many others [17, 18, 75–81] – exhibit structure amenable to quantum algorithms.

This thesis explores such connections in detail. Chapters 4 and 5 investigate how quantum resources may accelerate sequence-analysis tasks, focusing on the closest string problem and pangenome-guided assembly. These case studies demonstrate how quantum methods can address real-world biological challenges, offering a pragmatic pathway toward near-term quantum utility in the life sciences.

Chapter 3

Simulation and learning of matchgate circuits

Classical simulation of quantum computers not only delineates the limits of classical computation but also enables verification of quantum devices. It allows us to study and probe the properties of quantum systems without relying on a full-scale quantum computer. Matching simulated and experimental results validates device behaviour and sets a benchmark that any demonstration of quantum advantage must exceed [82, 83].

Rank-based methods are particularly effective when circuits are dominated by Clifford operations, as discussed in section 1.3.3. Their runtime is dominated by the *stabilizer rank* of tensor products of the magic state $|T\rangle$, denoted $\chi\big(|T\rangle^t\big)$, the minimum number of stabilizer states needed to decompose $|T\rangle^t$. This quantity is functional and non-convex, and therefore analytical bounds are hard to obtain. It is believed to scale exponentially with t (else $\mathsf{BQP} = \mathsf{P}$), but the best lower bound is only $\Omega(t^2)$ [84].

Bravyi et al. [85] introduced related quantities: the *approximate stabilizer rank* χ^δ, the *stabilizer extent* ξ and the *stabilizer fidelity* F_C of an n-qubit state $|\Psi\rangle$. These are given by

$$\chi^\delta(|\Psi\rangle) \coloneqq \min k \qquad \text{s.t.} \quad \left\| \sum_{i=1}^{k} c_i |\psi_i\rangle - |\Psi\rangle \right\| \leq \delta, \tag{3.1}$$

$$\xi(|\Psi\rangle) \coloneqq \min \|c\|_1^2 \qquad \text{s.t.} \quad \sum_i c_i |\psi_i\rangle = |\Psi\rangle\,, \tag{3.2}$$

$$F_C(|\Psi\rangle) \coloneqq \max_i |\langle\psi_i|\Psi\rangle|^2\,, \tag{3.3}$$

where $\{|\psi_i\rangle\}$ is the set of n-qubit stabilizer states.

Notably, finding the stabilizer extent is a convex optimisation problem. Moreover, these quantities are operationally relevant for near-Clifford simulation due to the following bounds from [85]:

$$\chi^{\delta}(|\Psi\rangle) \leq 1 + \xi(|\Psi\rangle)/\delta^2, \tag{3.4}$$

$$\xi(|\Psi\rangle) \geq 1/F_C(|\Psi\rangle). \tag{3.5}$$

When $|\Psi\rangle$ is a Clifford magic state, eq. (3.5) is in fact an equality, making $1/F_C$ an upper bound for χ^{δ}, which, in turn, controls the cost of approximate rank-based simulation.

While classical simulation is vital for validation, one may also infer properties of quantum devices directly from their outputs. This approach is *quantum process tomography*: the task of determining an unknown quantum operator from its action on certain known states. In general, tomography requires an exponentially large number of queries [86, 87], but an assumed structure can reduce this cost. For example, restricting to the set of Pauli matrices reduces the query complexity to 1 for any number of qubits, for essentially the same reason as superdense coding [88] is possible. Low [31] generalised this to learning arbitrary Clifford hierarchy operations C in $\mathcal{C}_k$ using $\mathcal{O}((2n)^k)$ queries to black boxes for C and $C^\dagger$, based on the observation that knowledge of $U\sigma_p U^\dagger$ for a set $\{\sigma_p\}$ that *generates* $\mathcal{P}$ suffices to learn U.

In this chapter, we extend the Clifford simulation and learning ideas to matchgate circuits. Section 3.1 derives explicit constraints characterising Gaussian states and applies them to classify dependent triples of states, bound the Gaussian fidelity and study Gaussian decompositions of magic states. In section 3.2, we present an efficient algorithm for learning unknown Gaussian operations from oracle access. Section 3.3 generalises this to an infinite family of operations that we define as the *matchgate hierarchy*, analogous to the Clifford hierarchy.

3.1 Simulating matchgate circuits

3.1.1 An explicit set of constraints for Gaussian states

Recall the operational definition of a Gaussian state $|\psi\rangle$:

$$\sum_{\mu=1}^{2n} \gamma_\mu \otimes \gamma_\mu |\psi\rangle |\psi\rangle = 0, \tag{3.6}$$

where the γ_μ are given in the Jordan-Wigner representation by

$$\gamma_{2l-1} = \left(\prod_{i=1}^{l-1} Z_i\right) X_l \quad \gamma_{2l} = \left(\prod_{i=1}^{l-1} Z_i\right) Y_l, \quad l = 1, \ldots, n. \tag{3.7}$$

While eq. (3.6) completely characterises Gaussian states, it is not convenient for analysing how the amplitudes are related. We instead derive an explicit set of independent algebraic constraints imposed by eq. (3.6).

We first need to introduce some notation. Let $d_H(\cdot, \cdot)$ be the Hamming distance between two bit strings of equal length. Let $\mathcal{A}_n$ be the set of n-bit strings with an even number of 1s, and $\mathcal{A}_n^{\mathsf{c}}$ the set with an odd number of 1s. Let $x[i, j]$ denote the substring of x from position i to j. We use an overline to denote changing bits of a binary string: $\overline{x}^i = x \oplus e_i$ and $\overline{x}^{i,j} = x \oplus e_i \oplus e_j$ for elementary vectors e_i.

For any $x \in \{0, 1\}^n$, we have

$$\gamma_{2l-1} |x\rangle = (-1)^{|x[1,l-1]|} \left|\overline{x}^l\right\rangle, \tag{3.8}$$

$$\gamma_{2l} |x\rangle = i(-1)^{|x[1,l-1]|}(-1)^{x_l} \left|\overline{x}^l\right\rangle. \tag{3.9}$$

In both cases, we get a sign factor from the string of Z_i and we flip the l^{th} bit. For γ_{2l}, we also pick up a factor of $i(-1)^{x_l}$.

Let $|\psi\rangle = \sum_{x \in \mathcal{A}_n} a_x |x\rangle$ be an even-parity state. From eq. (3.6), $|\psi\rangle \in G$ if and only if:

$$\sum_{\mu=1}^{2n} (\gamma_\mu \otimes \gamma_\mu) |\psi\rangle |\psi\rangle \tag{3.10}$$

$$= \sum_{\mu=1}^{2n} (\gamma_\mu \otimes \gamma_\mu) \sum_{x,y \in \mathcal{A}_n} a_x a_y |x\rangle |y\rangle \tag{3.11}$$

$$= \sum_{x,y \in \mathcal{A}_n} a_x a_y \sum_{\mu=1}^{2n} (\gamma_\mu \otimes \gamma_\mu) |x\rangle |y\rangle \tag{3.12}$$

$$= \sum_{x,y \in \mathcal{A}_n} \sum_{l=1}^{n} \left((-1)^{|x[1,l-1]|+|y[1,l-1]|} \left(1 + i^2 (-1)^{x_l + y_l}\right) a_x a_y \left|\overline{x}^l\right\rangle \left|\overline{y}^l\right\rangle \right) \tag{3.13}$$

$$= \sum_{x,y \in \mathcal{A}_n^{\mathsf{c}}} \sum_{l=1}^{n} \left((-1)^{|x[1,l-1]|+|y[1,l-1]|} \left(1 - (-1)^{x_l + y_l}\right) a_{\overline{x}^l} a_{\overline{y}^l} \right) |x\rangle |y\rangle = 0, \tag{3.14}$$

where the last line follows by redefining $x \leftarrow \overline{x}^l$ and $y \leftarrow \overline{y}^l$.

Consider some fixed $x,\, y \in \mathcal{A}_n^{\mathsf{c}}$. The inner sum of eq. (3.14) then reads

$$\sum_{l=1}^{n} (-1)^{|x[1,\,l-1]|+|y[1,\,l-1]|} \big(1-(-1)^{x_l+y_l}\big) a_{\overline{x}^l} a_{\overline{y}^l}. \tag{3.15}$$

If $x_l = y_l$, eq. (3.15) vanishes. So we may restrict attention to the set of indices where $x \neq y$. Let this set, written in increasing order, be $D(x,\, y)$.

We now show that the non-vanishing terms in eq. (3.15) alternate sign. Let $l,\, l'$ be a consecutive pair of indices in $D(x,\, y)$. The difference in the power of -1 for the corresponding terms is

$$|x[1,\,l'-1]| + |y[1,\,l'-1]| - |x[1,\,l-1]| - |y[y,\,1]| = |x[l,\,l'-1]| + |y[l,\,l'-1]|. \tag{3.16}$$

By construction, x and y agree on all bits $l+1, \ldots, l'-1$, but disagree on bit l. Therefore, eq. (3.16) resolves to an odd integer *i.e.* there is a relative sign of -1 between the terms.

We may therefore simplify the form of eq. (3.15). For any $x,\, y \in \mathcal{A}_n^{\mathsf{c}}$, we have

$$\sum_{i=1}^{|D(x,\,y)|} (-1)^{i+1} a_{\overline{x}^{l_i}} a_{\overline{y}^{l_i}}. \tag{3.17}$$

It will be helpful to write these equations in terms of labels with even parity rather than odd. To do this, we treat the first term of the sum in eq. (3.17) as preferential. Let $u = \overline{x}^{l_1}$ and $v = \overline{y}^{l_1}$, which have even weight. Then we may re-write eq. (3.17) as:

$$f_{u,\,v}(\{a_x\}) \coloneqq a_u a_v - \sum_{i=2}^{|D(u,\,v)|} (-1)^i a_{\overline{u}^{k_1,\,k_i}} a_{\overline{v}^{k_1,\,k_i}}. \tag{3.18}$$

We think of $f_{u,\,v} = 0$ as the constraint associated with strings $u,\, v \in \mathcal{A}_n$.

Note that if $d_H(u,\, v) = 2$ then the two terms on the RHS of eq. (3.18) cancel and the constraint $f_{u,\,v} = 0$ is trivial. $f_{u,\,u} = 0$ is trivial also, since the corresponding sum in eq. (3.17) has no terms.

Therefore, Gaussianity is equivalent to the satisfaction of the constraint set

$$\mathcal{F}(|\psi\rangle) \coloneqq \left\{ f_{u,\,v}(\{a_x\}) = 0 : (u,\, v) \in \mathcal{A}_n^{\otimes 2};\, d_H(u,\, v) \geq 4 \right\}. \tag{3.19}$$

To simplify notation, we will freely switch between binary and decimal representations of strings. For example, for $n = 4$, one constraint in $\mathcal{F}$ is

$$f_{0000,\,1111} \equiv f_{0,\,15} = a_0 a_{15} - a_3 a_{12} + a_5 a_{10} - a_6 a_9 = 0. \tag{3.20}$$

The other constraints – $f_{0011,1100}$, $f_{0101,1010}$, $f_{1001,0110}$ – are all equivalent to eq. (3.20) up to permutation of terms.

For an n-qubit state, ignoring duplication, $\mathcal{F}$ consists of $2^{n-1}\left(2^{n-1} - \frac{n(n-1)}{2} - 1\right)$ constraints on the 2^{n-1} coefficients $\{a_x\}$; clearly, not all of them are independent. To find an independent set of equations, we will use a complexified form of the *Implicit Function Theorem*:

Theorem 3.1.1 (Implicit Function Theorem, see, e.g., [89])**.** *Let*

$$\{g_j(\mathbf{w}, \mathbf{z}) : j = 1, 2, \ldots, m\} \tag{3.21}$$

be analytic functions of $(\mathbf{w},\, \mathbf{z}) = (w_1, \ldots, w_m, z_1, \ldots, z_n)$ *in a neighbourhood of* $(\mathbf{w_0},\, \mathbf{z_0}) \in \mathbb{C}^m \times \mathbb{C}^n$*. Suppose* $g_j(\mathbf{w_0},\, \mathbf{z_0}) = 0\ \forall\, j$ *and also that the Jacobian* $J \coloneqq \left(\frac{\partial g_j}{\partial w_k}\right)_{j,k=1}^{m}$ *satisfies* $\det(J) \neq 0$ *at* $(\mathbf{w_0},\, \mathbf{z_0})$*. Then, in a neighbourhood of* $\mathbf{z_0}$*,* $\{g_j(\mathbf{w},\, \mathbf{z}) = 0\}$ *has a unique analytic solution such that* $\mathbf{w}(\mathbf{z_0}) = \mathbf{w_0}$*, given implicitly by* $(\mathbf{w},\, \mathbf{z}) = (\mathbf{w}(\mathbf{z}),\, \mathbf{z})$*.*

Intuitively, theorem 3.1.1 states conditions under which m constraints remove m degrees of freedom on $m + n$ variables. Using theorem 3.1.1, we may find a maximal set of independent constraints from $\mathcal{F}(|\psi\rangle)$, which therefore define Gaussian states.

Proposition 3.1.1. *An even-parity Gaussian state* $|\psi\rangle = \sum_{x \in \mathcal{A}_n} a_x |x\rangle$ *on* n *qubits, with* $a_y \neq 0$*, is defined by a set of* $2^n - \frac{n(n-1)}{2}$ *algebraic constraints. These may be chosen as* $\mathcal{F}^y_{indep} \coloneqq \{f_{y,\,w}(\{a_x\}) = 0 : d_H(y,\, w) \geq 4\}$*, where the* $f_{y,\,w}$ *are defined by eq.* (3.18)*, along with a normalisation and global phase equivalence condition.*

Proposition 3.1.1 restricts the higher-distance amplitudes to analytic functions of those with Hamming distance to $y \leq 2$.

Proof. Assume without loss of generality that $a_0 \neq 0$; the proof easily generalises to any other $y \in \mathcal{A}_n$ such that $a_y \neq 0$. We show that

$$\mathcal{F}^0_{\text{indep}} \coloneqq \{f_{0,\,w} : w \in \mathcal{A}_n;\ |w| \geq 4\}. \tag{3.22}$$

satisfy the conditions of theorem 3.1.1. First, define the following partition of $\mathcal{A}_n$:

$$Z \coloneqq \{z \in \mathcal{A}_n : |z| \leq 2\}, \quad W \coloneqq \{w \in \mathcal{A}_n : |w| \geq 4\}. \tag{3.23}$$

Define a further partition of W based on the first "set" bit:

$$W_p \coloneqq \{w \in W : \; w_1 = \ldots = w_{p-1} = 0; \; w_p = 1\}, \quad p = 1, 2, \ldots n-3. \tag{3.24}$$

For some $w \in W_p$, let $\{k_1 = p, \ldots, k_{|w|}\}$ be the set of indices where w has a 1. Then the form of $f_{0,w}$, as given by eq. (3.18), is:

$$f_{0,w}(\{a_x\}) \coloneqq a_0 a_w - \sum_{i=2}^{|w|} (-1)^i a_{\overline{0}^{p,k_i}} a_{\overline{w}^{p,k_i}}. \tag{3.25}$$

Crucially, every label x that appears in eq. (3.25) only has 1s on indices greater than or equal to p. Also, the strings $\overline{0}^{p,k_i}$ have weight two so are not members of any W_q.

With these facts in hand, we can compute the Jacobian elements. For $w \in W_p$,

$$J_{wv} = \frac{\partial f_{0,w}}{\partial a_v} = \begin{cases} a_0 \delta_{wv} & v \in W_p, \\ 0 & v \in W_q, \; q < p, \\ \bullet & v \in W_q, \; q > p. \end{cases} \tag{3.26}$$

Equation (3.26) implies the Jacobian is of block-upper-triangular form with multiples of the identity on the diagonal. We need not consider the form of J_{wv} for $v \in W_q$, $q > p$ since these entries do not contribute to the determinant of J. Letting I_r represent the $r \times r$ identity matrix, we have

$$J = \left(\begin{array}{c|c|c|c} a_0 I_{|W_1|} & \bullet & \bullet & \cdots \\ \hline 0 & a_0 I_{|W_2|} & \bullet & \cdots \\ \hline 0 & 0 & a_0 I_{|W_3|} & \cdots \\ \hline \vdots & \vdots & \vdots & \ddots \end{array}\right). \tag{3.27}$$

The determinant of J is the product of the determinants of the diagonal blocks. Since $a_0 \neq 0$ by assumption, we have $\det(J) \neq 0$. We now have a set of functions, a partition of variables and a non-zero determinant, as required by theorem 3.1.1.

Therefore, there is a unique analytic solution to the set of constraints $\mathcal{F}^0_{\text{indep}} \coloneqq \{f_{0,w} = 0 : |w| \geq 4\}$. For this solution, for any $w \in W$, we have an implicit expression $a_w = a_w(\{a_z : |z| \leq 2\})$. In words, the coefficients of $|\psi\rangle$ labelled by strings w with $|w| \geq 4$ are functions of the coefficients with labels z where $|z| \leq 2$. Since $|Z| = \binom{n}{2} + 1$, we have an independent set of $2^n - \frac{n(n-1)}{2}$ constraints as claimed.

The proof generalises to any favoured $y \in \mathcal{A}_n$ with $a_y \neq 0$ by replacing the bit string 0 by y and the Hamming weight $|\cdot|$ by the Hamming distance $d_H(y, \cdot)$ throughout. The corresponding independent set of constraints which define Gaussian states is then:

$$\mathcal{F}^{y}_{\text{indep}}(|\psi\rangle) = \Big\{ f_{y,w}(\{a_x\}) \coloneqq a_y a_w - \sum_{i=2}^{|D(w,y)|} (-1)^i a_{\overline{y}^{k_1,k_i}} a_{\overline{w}^{k_1,k_i}} = 0 : d(y,\, w) \geq 4 \Big\}. \tag{3.28}$$

□

3.1.2 Triples of Gaussian states

The explicit representation above allows analysis of linear dependencies among Gaussian states.

Proposition 3.1.2. *If $|\psi_0\rangle = \alpha\,|\psi_1\rangle + \beta\,|\psi_2\rangle$ for normalised Gaussian states $|\psi_i\rangle$ and (without loss of generality) real α, β, then the triple can be written as*

$$\{|\psi_0\rangle\,,\, |\psi_1\rangle\,, |\psi_2\rangle\} = \Big\{ U\,|0\rangle\,,\, \frac{1}{\alpha} U \sum_{\substack{y\in\mathcal{A}_n:\\ |y|\leq 2}} a_y\,|y\rangle\,,\, \frac{1}{\beta} U \Big((1-a_0)\,|0\rangle - \sum_{\substack{y\in\mathcal{A}_n:\\ |y|=2}} a_y\,|y\rangle \Big) \Big\}. \tag{3.29}$$

for some Gaussian operation $U \in \mathcal{G}$ and some complex coefficients $\{a_y : |y| \leq 2\}$ such that

$$\sum_{\substack{y\in\mathcal{A}_n:\\ |y|\leq 2}} a_y\,|y\rangle \quad \text{and} \quad (1-a_0)\,|0\rangle - \sum_{\substack{y\in\mathcal{A}_n:\\ |y|=2}} a_y\,|y\rangle\,. \tag{3.30}$$

are Gaussian states.

Moreover, the manifold of Gaussians for which $\alpha\,|\psi\rangle + \beta\,|\psi'\rangle \in G$ has dimension is $2n-3$.

Proof. Fix some $|\psi_0\rangle \in G_n$, and suppose $|\psi_0\rangle = \alpha\,|\psi_1\rangle + \beta\,|\psi_2\rangle$ for real coefficients α, β. Let $U \in \mathcal{G}$ be the Gaussian operator s.t. $|\psi_0\rangle = U\,|0\rangle$. Also define $|\tilde{\psi}_1\rangle = U^\dagger\,|\psi_1\rangle\,,\ |\tilde{\psi}_2\rangle = U^\dagger\,|\psi_2\rangle$. Then we have $|0\rangle = \alpha\,|\tilde{\psi}_1\rangle + \beta\,|\tilde{\psi}_2\rangle$.

Write $\alpha\,|\tilde{\psi}_1\rangle = \sum_{x\in\mathcal{A}_n} a_x\,|x\rangle$, where, without loss of generality, we can set $0 < a_0 \leq 1$. Then $\beta\,|\tilde{\psi}_2\rangle = (1-a_0)\,|0\rangle - \sum_{x\in\mathcal{A}_n:|x|\geq 2} a_x\,|x\rangle$. We require these states to be Gaussian.

For each $x \in \mathcal{A}_n$ with $|x| \geq 4$, we have a constraint of the form

$$f_{0,\,x} = c_0 c_x - \sum_{i=2}^{|x|} (-1)^i c_{\overline{0}^{k_1,k_i}} c_{\overline{x}^{k_1,k_i}} = 0. \tag{3.31}$$

Imposing this constraint for $\alpha\,|\tilde{\psi}_1\rangle$ and $\beta\,|\tilde{\psi}_2\rangle$ respectively gives:

$$a_0 a_x - \sum_{i=2}^{|x|} (-1)^i a_{\overline{0}^{k_1,k_i}} a_{\overline{x}^{k_1,k_i}} = 0, \tag{3.32}$$

$$(1-a_0)(-a_x) - \sum_{i=2}^{|x|} (-1)^i (-a_{\overline{0}^{k_1,k_i}})(-a_{\overline{x}^{k_1,k_i}}) = 0. \tag{3.33}$$

Comparing eqs. (3.32) and (3.33) we see that $a_x = 0$ for all $|x| \geq 4$. We therefore have:

$$\alpha\,|\tilde{\psi}_1\rangle = a_0\,|0\rangle + \sum_{y\in\mathcal{A}_n:|y|=2} a_y\,|y\rangle\,, \tag{3.34}$$

$$\beta\,|\tilde{\psi}_2\rangle = (1-a_0)\,|0\rangle - \sum_{y\in\mathcal{A}_n:|y|=2} a_y\,|y\rangle\,, \tag{3.35}$$

which is the first part of our result.

It remains to show the dimension of the manifold of coefficients $\{a_y : |y| \leq 2\}$ such that eqs. (3.34) and (3.35) hold.

We first note that $a_x = 0$ for all $|x| = 4$ is sufficient to enforce $a_x = 0$ for all $|x| \geq 4$. Consider some x with $|x| = 6$. Each term in the sum of eq. (3.31) contains some coefficient labelled by $\overline{x}^{k_1,\,k_i}$. Since $\left|\overline{x}^{k_1,\,k_i}\right| = 4$, each summand vanishes. Thus eq. (3.31) reduces to $a_0 a_x = 0$. Since $a_0 \neq 0$ by assumption, we must have $a_x = 0$. Proceeding iteratively for $|x| = 8,\, 10,\, \ldots$, we see that $a_x = 0$ for all $|x| \geq 4$.

We therefore only need to enforce $a_x = 0$ for all $|x| = 4$. The relevant constraints on the a_y with $|y| \leq 2$ are then

$$\tilde{\mathcal{F}} = \left\{ \sum_{i=2}^{4} (-1)^i a_{\overline{0}^{k_1,k_i}} a_{\overline{x}^{k_1,k_i}} = 0 : |x| = 4,\, D(0,\,x) = \{k_i : i = 1,\,\ldots,\,4\} \right\}. \tag{3.36}$$

For each y, $\exists\, \binom{n-2}{2}$ choices of $x \in \mathcal{A}_n$ with $|x| = 4$ such that $d(x,\,y) = 2$. In other words, each a_y appears in exactly $\binom{n-2}{2}$ equations. We may assume $a_{y^*} \neq 0$ for some $|y^*| = 2$, else the vectors in the linearly dependent triple are all parallel. In appendix A.1 we show that satisfying the constraints explicitly containing a_{y^*} is sufficient to meet all the constraints. This leaves $\binom{n}{2} - \binom{n-2}{2} = 2n-3$ of the variables unconstrained. □

3.1.3 An upper bound on the fermionic Gaussian fidelity of Haar-random states

Analogous to the stabilizer fidelity F_C [85], we introduce the *Gaussian fidelity* F_G:

Definition 3.1.1. *The Gaussian fidelity $F_G(|\phi\rangle)$ of an n-qubit state $|\phi\rangle$ is given by*

$$F_G(|\phi\rangle) \coloneqq \sup_{|\psi\rangle \in G_n} |\langle\phi|\psi\rangle|^2. \tag{3.37}$$

Using an ϵ-net construction [90], we prove that for a Haar-random state the Gaussian fidelity is exponentially small with overwhelming probability. Intuitively, an ϵ-net is a finite set of states that tile a Hilbert space such that all states in some other set are close to a state in the net.

Definition 3.1.2. *An ϵ-net $\mathcal{N} \subset G$ is a set of Gaussian states such that for every $|\psi\rangle \in G_n$ there exists $|\tilde{\psi}\rangle \in \mathcal{N}$ with $\left\| |\psi\rangle - |\tilde{\psi}\rangle \right\|_1 < \epsilon$.*

We first bound the cardinality of our ϵ-net:

Proposition 3.1.3. *Let $\epsilon = 2^{-l}$. Then there is an ϵ-net $\mathcal{N}$ with cardinality $\log_2 |\mathcal{N}| \leq n^4 + 3n^2 + n + ln^2$.*

We defer the proof to appendix A.2.

By a standard union-bound argument over the net, we obtain a bound for the overlap of random states with this discrete set. This directly leads to a bound on the overlap with any Gaussian state, i.e., a bound on the fidelity:

Proposition 3.1.4. *For a generic Haar-random state $|\Psi\rangle$, the Gaussian fidelity is exponentially small with probability exponentially close to unity. In particular, for any $\delta > 0$, $F_G(|\Psi\rangle) \leq 2^{-n+2}(1+\delta)\log(2)n^4$ with probability $\approx 1 - e^{-\delta n^4}$.*

Proof. Following the proof of Claim 2 in [85], for any n-qubit state $|\tilde{\psi}\rangle$ and a Haar-random state $|\Psi\rangle$, we have

$$\mathbb{P}\left(|\langle\tilde{\psi}|\Psi\rangle|^2 \geq x \right) = (1-x)^{2^n - 1} \leq e^{-x(2^n-1)}. \tag{3.38}$$

Taking a union bound over $\mathcal{N}$ with $x = 2^{-l}$,

$$\mathbb{P}\left(\max_{|\tilde{\psi}\rangle \in \mathcal{N}} |\langle\tilde{\psi}|\Psi\rangle|^2 \geq 2^{-l} \right) \leq |\mathcal{N}| \cdot \exp\left(-2^{-l}(2^n - 1)\right) \tag{3.39}$$

$$\leq \exp\left(-2^{n-l} + (n^4 + 3n^2 + n + ln^2)\log(2)\right). \tag{3.40}$$

Take $l = n - \log_2((1+\delta)\log(2)n^4)$ for any $\delta > 0$. Then

$$\mathbb{P}\left(\max_{|\tilde{\psi}\rangle\in\mathcal{N}} |\langle\tilde{\psi}|\Psi\rangle|^2 \geq 2^{-l}\right) \leq \exp\left(-\delta n^4 + \mathcal{O}(n^3)\right). \tag{3.41}$$

By definition of a 2^{-l}-net, we can write $|\psi\rangle = |\tilde{\psi}\rangle + 2^{-l}|\zeta\rangle$, where $\||\zeta\rangle\|_1 < 1$, $|\psi\rangle \in G$ and $|\tilde{\psi}\rangle \in \mathcal{N}$. Then

$$\begin{aligned} F_G(|\Psi\rangle) &= \sup_{|\psi\rangle\in G} |\langle\psi|\Psi\rangle|^2 && (3.42)\\ &= |\langle\tilde{\psi}|\Psi\rangle + 2^{-l}\langle\zeta|\Psi\rangle|^2 && (3.43)\\ &\leq (2^{-\frac{l}{2}} + 2^{-l}\cdot 1)^2 && (3.44)\\ &\leq 2^{-l+2} && (3.45)\\ &= 2^{-n+2}(1+\delta)\log(2)n^4, && (3.46) \end{aligned}$$

with probability $\approx 1 - e^{-\delta n^4}$. □

3.1.4 Gaussian decompositions of magic states

Rank-based simulation methods from the Clifford setting can be applied to matchgate circuits, since they are also efficiently classically simulable. Arbitrary quantum circuits are first written in terms of the universal "Matchgate + SWAP" gateset. Each SWAP gate is replaced by a SWAP-gadget, consuming an ancilla register prepared in the magic state $|M\rangle$. The tensor product of all these magic states, $|M\rangle^k$, is then decomposed into a sum of Gaussian states. The resulting circuit acting on any term in the decomposition is classically simulable in polynomial time, since it consists of a matchgate circuit acting on Gaussian inputs. A sketch of this procedure is provided in fig. 3.1.

The dominant cost of such simulation methods is the number of terms in the Gaussian decomposition of $|M\rangle^k$. In analogy with stabilizer methods, we call this quantity the *Gaussian rank*:

Definition 3.1.3. *The Gaussian rank $\chi_G(|\Psi\rangle)$ of an n-qubit state $|\Psi\rangle$ is the smallest integer k such that $|\Psi\rangle$ can be written as*

$$|\Psi\rangle = \sum_{i=1}^{k} c_i |\psi_i\rangle, \tag{3.47}$$

for some Gaussian states $|\psi_i\rangle \in G$ and some complex coefficients $c_i \in \mathbb{C}$.

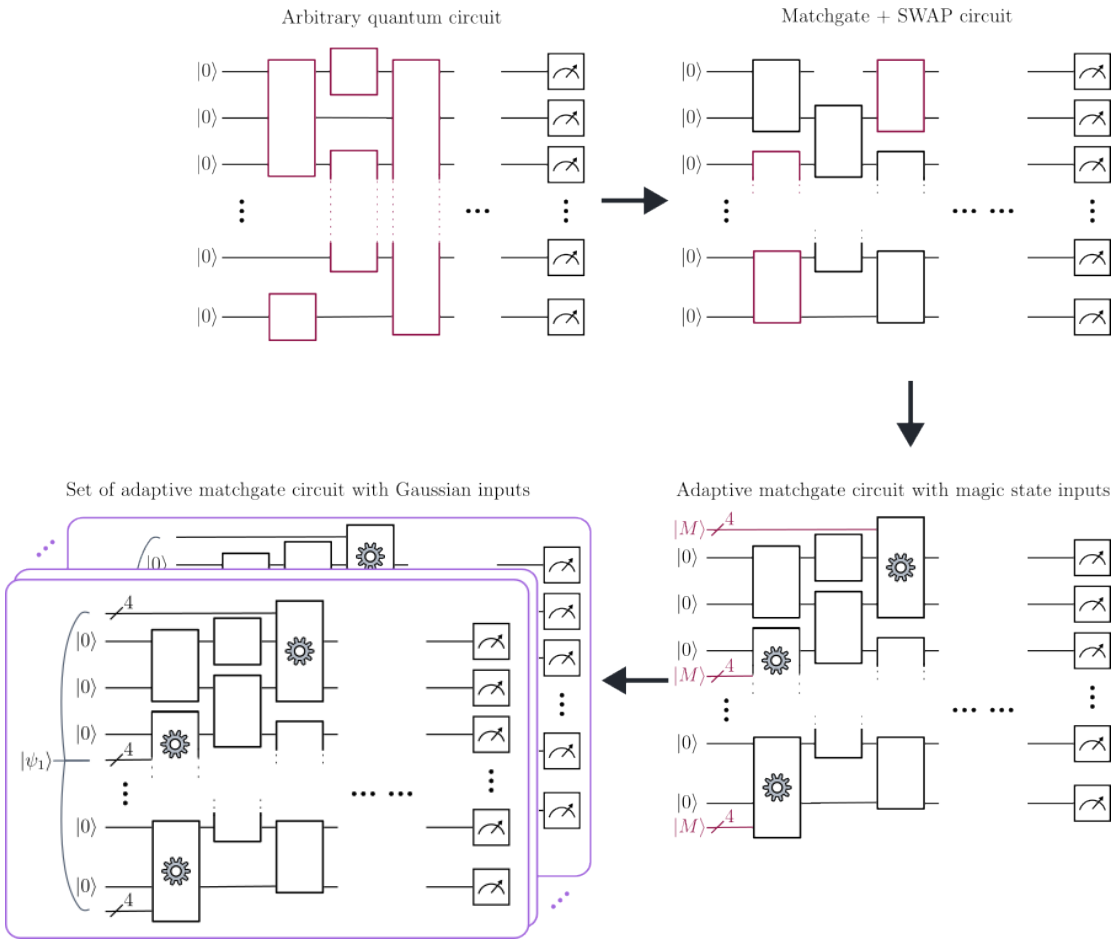


Fig. 3.1 A sketch of a Gaussian-rank-based classical simulation procedure. Throughout, the colour red denotes a non-free resource, namely non-matchgate gates or magic state inputs. (Upper-left) The starting point is a generic quantum circuit containing arbitrary gates. (Upper-right) The circuit is expressed in the "matchgate + SWAP" gate set. This generally increases the circuit depth, but the increase is at most logarithmic if the original gates act on at most a constant number of qubits. (Lower-right) Each SWAP gate is replaced by a SWAP-gadget, a small, adaptive, matchgate circuit that acts on the original qubit and consumes an ancilla register in the initial state $|M\rangle$. Notably, all gates in the circuit are now matchgates. (Lower-left) The state $|M\rangle^k$ is decomposed into an exponentially large sum of Gaussian states. The (adaptive) matchgate circuit acting on any term in this sum can be classically simulated.

It is often sufficient to consider the *approximate* rank, in which some error is allowed in the decomposition:

Definition 3.1.4. *The approximate Gaussian rank $\chi_G^\delta(|\Psi\rangle)$ of an n-qubit state $|\Psi\rangle$ is the smallest integer k such that $\||\Psi\rangle - |\Psi'\rangle\| \leq \delta$ for some state $|\Psi'\rangle$ with exact Gaussian rank k.*

For simulation algorithms, we are interested in the Gaussian rank of several copies of the magic state $|M\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle)$. An easily checked upper bound $\chi_G(|\Psi\rangle^k) \leq \big(\chi_G(|\Psi\rangle^{k'})\big)^{k'/k}$ for any k' that divides k means that low-rank decompositions of only two or three copies of $|M\rangle$ are sufficient to speed up simulation of large near-matchgate circuits.

Note that there is a subtlety in the choice of magic state for matchgate circuits that is not present for Clifford circuits. Hebenstreit et al. [41] showed that all non-Gaussian states with definite parity are magic. For example, we could consider a parameterized family of magic states

$$|M(\alpha)\rangle \coloneqq (1-\alpha)\,|0000\rangle + \alpha\,|1111\rangle\,. \tag{3.48}$$

For any $0 < \alpha < 1$, this state violates the constraint $f_{0,15} = 0$ given in eq. (3.20). But as $\alpha \to 0$ (or 1), $|M(\alpha)\rangle$ approaches a Gaussian state. Therefore, while $|M(\alpha)\rangle$ has an exact Gaussian rank of $\chi_G(|M(\alpha)\rangle) = 2$, it has an approximate Gaussian rank of $\chi_G^\delta(|M(\alpha)\rangle) = 1$ for all $\delta \leq \alpha$. Indeed, $\chi_G^\delta(|M(\alpha)\rangle^k) = 1$ for all $\delta \leq 1 - \big(\max(\alpha, 1-\alpha)\big)^k$. Such trivial decompositions do not lead to interesting simulation properties; we will focus only on magic states with sufficiently low Gaussian fidelity.

Our canonical magic state $|M\rangle$ has the minimal Gaussian fidelity of 1/2 and is therefore in some sense "maximally magic". Formally quantifying measures of Gaussian magic remains an interesting research direction that we do not consider here.

We used numerical optimisation tools from the *SciPy* [91] package to seek out low-rank decompositions. Specifically, we chose simulated annealing with cost function $L_{|M\rangle}(|\Psi\rangle) = \Big\| |M\rangle^k - |\Psi\rangle \Big\|_2^2$. Sample code is available on GitHub here.

Our program failed to find any exact low-rank decompositions for either $k = 2$ or $k = 3$ after hundreds of hours of machine time using global optimisation techniques, suggesting that exact low-rank decompositions are unlikely. Indeed, the code never found an approximate decomposition better than a trivial one for $|M\rangle\,|M\rangle = \frac{1}{2}(|00000000\rangle + |00001111\rangle + |11110000\rangle + |11111111\rangle)$. For example, one computed solution was $|\Psi'\rangle = \frac{1}{2}(|00000000\rangle + |00001111\rangle + |11110000\rangle)$, using the 3 Gaussian

states to get 3 out of 4 non-zero amplitudes of $|M\rangle^2$ correct. Similarly, no good decomposition was found for $k = 3$.

We conjecture that good decompositions, even approximate ones, do not exist for only a few copies of this state due to the low Gaussian fidelity of $|M\rangle$. In particular, we conjecture that $\chi_G(|M\rangle |M\rangle) = 4$. A proof of this remains an open question.

However, we can make progress in a simplified setting. In particular, $|M\rangle$ is invariant under the (Gaussian) operation $Z_3Z_4 = G_{34}(I, -I)$. This motivates considering a decomposition into only Gaussian states that are invariant under the same operation on each block of 4 qubits.

Definition 3.1.5. *Let $|\Psi\rangle$ be a $4n$-qubit state. For $j = 1, \ldots, n$, let $O_j \coloneqq Z_{4j-1}Z_{4j}$. The* symmetry-restricted Gaussian rank $\tilde{\chi}_G(|\Psi\rangle)$ *is the smallest integer k such that $|\Psi\rangle$ can be written as*

$$|\Psi\rangle = \sum_{i=1}^{k} c_i |\psi_i\rangle \tag{3.49}$$

for some complex coefficients c_i and some n-qubit Gaussian states $|\psi_i\rangle$ which satisfy $O_j |\psi_i\rangle = |\psi_i\rangle$ for $j = 1, \ldots, n$.

Proposition 3.1.5. *The symmetry-restricted Gaussian rank of $|M\rangle^2$ is $\tilde{\chi}_G(|M\rangle^2) = 4$.*

We defer a full proof to appendix A.3, but give some discussion here.

The symmetry is chosen to simplify the constraints on each Gaussian state significantly. Let $c_i |\psi\rangle = \sum_x a_x |x\rangle$ be a non-normalised Gaussian state with non-zero amplitudes only for computational-basis states $|x\rangle$ which are invariant under O_j. Almost all the constraints on $\{a_x\}$ have only two terms, and are much easier to work with than the general form in eq. (3.18). We can visualise these constraints in a grid, wherein amplitudes with labels in the body of the grid are defined in terms of the amplitudes labelled by leading entries of their row and column. Table 3.1 represents all these constraints present on a restricted Gaussian state with eight qubits.

The only exceptions are a pair of constraints that have four terms. These can be viewed as fixing the values of the last entry in the first row and column, respectively:

$$a_0a_{195} - a_3a_{192} + a_{65}a_{130} - a_{66}a_{129} = 0 \tag{3.50}$$

$$a_0a_{60} - a_{12}a_{48} + a_{20}a_{40} - a_{24}a_{36} = 0 \tag{3.51}$$

If $a_0 = 0$, then we must also set all the amplitudes labelled by either the first row or the first column to 0. This follows since, reading off each constraint from the second

Table 3.1 Constraints on an 8-qubit, symmetry-restricted Gaussian state, assuming $a_0 \neq 0$, are determined by examining a component within the body and multiplying it by the upper-leftmost entry, which equals the product of its row and column headers. For example, $a_{15}a_0 = a_3a_{12}$, $a_{90}a_0 = a_{66}a_{24}$, etc.

0	3	65	66	129	130	192	195
12	15	77	78	141	142	204	207
20	23	85	86	149	150	212	215
24	27	89	90	153	154	216	219
36	39	101	102	165	166	228	231
40	43	105	106	169	170	232	235
48	51	113	114	177	178	240	243
60	63	125	126	189	190	252	255

column of the grid, we have

$$a_3a_{12} = a_3a_{20} = a_3a_{24} = a_3a_{36} = a_3a_{40} = a_3a_{48} = a_3a_{60} = 0. \tag{3.52}$$

Thus, either every term in the first column vanishes, or $a_3 = 0$. Proceeding similarly along the columns, we see that either the first column or the first row must all vanish. This corresponds to deleting the entire first column or first row of the grid. The table remains valid as long as the value in the upper-leftmost entry is non-zero. For example, we might have $a_{12} = a_{20} = a_{24} = a_{36} = a_{40} = a_{48} = a_{60} = 0$, $a_3 \neq 0$. Then constraints are read off like $a_3a_{77} = a_{15}a_{65}$.

The proof of proposition 3.1.5 proceeds by contradiction. Suppose that $|M\rangle^2 = c_1 |\psi_1\rangle + c_2 |\psi_2\rangle + c_3 |\psi_3\rangle$. For $|\psi_3\rangle$, each amplitude labelled by a term in the body of table 3.1 is defined twice: once by the requirement that $|\psi_3\rangle$ is Gaussian; and a second time by the requirement that the $c_i |\psi_i\rangle$ sum to $|M\rangle^2$. This gives a system of equations which we prove is insoluble. Edge cases are spawned when one or more of the states have $a_0 = 0$, but these can each be resolved by iteratively considering deleting rows or columns of table 3.1.

We note some obvious limitations of this argument. Firstly, there is little justification for the symmetry reduction beyond some appeal to the symmetry of the problem. Without the reduction, the algebraic constraints are difficult to work with, as there is no straightforward distinction between independent and dependent components. It is unlikely that our proof techniques will extend to the unrestricted case. Additionally, the method is highly tailored to the case of two copies of the magic state. With any more copies, the number of terms becomes too unwieldy, and it is impractical to unwind all the definitions, even for the reduced case. We therefore expect that further

results on lower bounds for the Gaussian rank will require new tools beyond symmetry reductions.

3.2 Learning Gaussian operations

3.2.1 Problem setup and result

Since the group of Gaussian operations is continuous, learning an unknown member of the group can only be done up to some finite precision. Further, physically equivalent unitaries which differ only by a global phase are indistinguishable. We therefore consider the two metrics used in [31], which we refer to as the *phase-sensitive* and *phase-insensitive* distances, respectively:

Definition 3.2.1. *Let U_1 and U_2 be two $d \times d$ unitary matrices, where $d = 2^n$. The phase-sensitive distance $D^+(U_1, U_2)$ is given by*

$$D^+(U_1, U_2) \coloneqq \frac{1}{\sqrt{2d}} \|U_1 - U_2\|_2 \tag{3.53}$$

$$\equiv \sqrt{1 - \operatorname{Re}\left(d^{-1} \operatorname{Tr}\left(U_1^\dagger U_2\right)\right)}. \tag{3.54}$$

The phase-insensitive distance $D(U_1, U_2)$ is given by

$$D(U_1, U_2) \coloneqq \frac{1}{\sqrt{2d^2}} \|U_1 \otimes U_1^* - U_2 \otimes U_2^*\|_2 \tag{3.55}$$

$$\equiv \sqrt{1 - \left|d^{-1} \operatorname{Tr}\left(U_1^\dagger U_2\right)\right|^2}, \tag{3.56}$$

where $\|A\|_2 = \sqrt{\operatorname{Tr}(A^\dagger A)}$.

Although D is not a true distance, we can check that:

1. $D(U_1, U_2) = 0$ implies $U_1 = e^{i\phi} U_2$ for some phase ϕ.
2. D is unitarily invariant.
3. D satisfies the triangle inequality.

Since D is phase-insensitive, it serves as a helpful measure for assessing the quality of our estimates for unknown Gaussian unitaries.

Now suppose that $M = \exp(iH_f)$ is an unknown Gaussian operation. If we had access to copies of the Gibbs state $\rho(H_f)$, we could recover h and therefore M. Since

$\Gamma(\rho(H_f)) = \tan(2h)$, we would only need to compute estimates for the $n(2n-1)$ independent entries of $\Gamma(\rho(H_f))$ to obtain an estimate for h. This process requires $\mathcal{O}(n(2n-1)/\epsilon^2)$ copies of the Gibbs state to obtain a precision of ϵ.

Instead, suppose we only have access to the gates M and $M^\dagger$ as black boxes. Recall that a Gaussian operation M on n qubits is uniquely defined, up to a global phase, by a special orthogonal matrix $Q \in SO(2n)$, and we write $M = M_Q$. We consider a 3-step learning process:

1. **Unsigned entries:** sample each row of Q in turn, obtaining a guess $\tilde{Q}$, which is identical to Q up to additive error and entry-wise incorrect signs.

2. **Row-wise consistency:** compute estimates of two-point correlations $\Gamma(\phi)_{jk}$ for certain pure Gaussian states $|\phi\rangle = MX_l |0\rangle$, refining our guess to $\overline{Q}$, which equals Q up to overall signs for each row and additive error.

3. **Per-row correction:** perform a single distinguishing measurement to correct the signs for each row, leading to an additive error estimate of Q.

The above protocol leads to the following result:

Proposition 3.2.1. *Given oracle access to an unknown operation $M \in \mathcal{G}_n$ and its conjugate $M^\dagger$, an estimate M' of M satisfying $D(M, M') = \mathcal{O}(n^3\eta)$ can be determined with high probability using $\tilde{\mathcal{O}}((n/\eta)^2)$ queries for any $\eta = \mathcal{O}(n^{-6})$.*

We prove the correctness and complexity of the protocol below. Note that similar results on efficient tomography of Gaussian channels [92] and efficient learning of states prepared by circuits with only a small number of non-matchgate gates [93] are available. The former employs a similar technique to learn the corresponding orthogonal matrix. The latter considers a more general setting but learns only the Gaussian state, rather than the entire Gaussian operation, and relies on compressing the "non-Gaussianity" into a small state on which full tomography is performed.

3.2.2 Finding unsigned entries of the orthogonal matrix

Let $|\psi\rangle \coloneqq 2^{-n/2} \sum_i |i\rangle |i\rangle$ be the maximally entangled state on $2n$ qubits. Recall that for any operator A

$$\begin{aligned}
\langle\psi| A \otimes I |\psi\rangle &= \operatorname{Tr}((A \otimes I) |\psi\rangle \langle\psi|) && (3.57)\\
&= 2^{-n} \sum_{i,j} \operatorname{Tr}(A |i\rangle \langle j| \otimes |i\rangle \langle j|) && (3.58)\\
&= 2^{-n} \sum_{i} \operatorname{Tr}(A |i\rangle \langle i|) = 2^{-n} \operatorname{Tr}(A). && (3.59)
\end{aligned}$$

The set $\{(\gamma_S M_Q \otimes I) |\psi\rangle : S \subseteq [2n]\}$ forms an orthonormal basis of $\mathcal{H}_2^{2n}$:

$$\begin{aligned}
\langle\psi| (M_Q^\dagger \gamma_{S'}^\dagger \otimes I)(\gamma_S M_Q \otimes I) |\psi\rangle &= 2^{-n} \operatorname{Tr}\Big(M_Q^\dagger \gamma_{S'}^\dagger \gamma_S M_Q\Big) && (3.60)\\
&= 2^{-n} \sum_{|P|=|S'\Delta S|} (\pm) \det(Q|_{S\Delta S',\,P}) \operatorname{Tr}(\gamma_P) && (3.61)\\
&= \delta_{S,\,S'}, && (3.62)
\end{aligned}$$

where the first equality follows from eq. (3.59), the second from the conjugation behaviour of matchgates given in eq. (1.15), and the final equality holds since $\operatorname{Tr}(\gamma_S) = 0$ for any $S \neq \emptyset$.

A measurement of the state $(M_Q \gamma_\mu \otimes I) |\psi\rangle$ with respect to $\{(\gamma_S M_Q \otimes I) |\psi\rangle : S \subseteq [2n]\}$ has outcome probabilities

$$\begin{aligned}
\mathbb{P}(S) &= \Big|\langle\psi| (M_Q^\dagger \gamma_S^\dagger \otimes I)(M_Q \gamma_\mu \otimes I) |\psi\rangle\Big|^2 && (3.63)\\
&= \Big|2^{-n} \operatorname{Tr}\Big(M_Q^\dagger \gamma_S^\dagger M_Q \gamma_\mu\Big)\Big|^2 && (3.64)\\
&= \Big|2^{-n} \sum_{|S'|=|S|} \det(Q|_{S,\,S'}) \operatorname{Tr}(\gamma_{S'} \gamma_\mu)\Big|^2 && (3.65)\\
&= \begin{cases} Q_{\mu\nu}^2 & S = \{\nu\}, \\ 0 & \text{otherwise.} \end{cases} && (3.66)
\end{aligned}$$

Suppose we take K such measurements and report estimates of the squared entries $Q_{\mu\nu}^2$ based on the observed frequencies. By the Hoeffding inequality and a union bound, we can learn each squared entry within additive error η using $K = \mathcal{O}(\log(n)/\eta^2)$ measurements with high probability. By taking the positive square roots, this is

equivalent to learning the unsigned entries $\pm Q_{\mu\nu}$ up to an additive error $\eta/2$. Repeating for each row requires a total of $\mathcal{O}(n\log(n)/\eta^2)$ measurements.

3.2.3 Fixing signs up to row-wise sign errors

We now have a matrix consisting of the unsigned entries of Q with additive errors. Let the matrix be $\tilde{Q} = \tilde{Q}_{\mu\nu}$. Let the necessary sign corrections be $t_\mu s_{\mu\nu}$, where t_μ, $s_{\mu\nu} \in \{\pm 1\}$ and we have factored out whole-row sign corrections for each row. Explicitly,

$$\overline{Q}_{\mu\nu} \coloneqq s_{\mu\nu}\tilde{Q}_{\mu\nu} + \eta_{\mu\nu}, \tag{3.67}$$

$$Q_{\mu\nu} = t_\mu \overline{Q}_{\mu\nu} = t_\mu s_{\mu\nu}\tilde{Q}_{\mu\nu} + t_\mu\eta_{\mu\nu}, \tag{3.68}$$

where $|\eta_{\mu\nu}| < \eta$ and without loss of generality $s_{\mu,1} = +1$.

We will estimate certain 2-point correlation functions:

$$\Gamma^{(0)}_{jk} \coloneqq \langle 0|\, M^\dagger(i\gamma_j\gamma_k)M\,|0\rangle\,, \tag{3.69}$$

$$\Gamma^{(l)}_{jk} \coloneqq \langle 0|\, X_l M^\dagger(i\gamma_j\gamma_k)MX_l\,|0\rangle\,. \tag{3.70}$$

We may compute an explicit expression for $\Gamma^{(0)}_{jk}$:

$$\Gamma^{(0)}_{jk} = i\sum_{a<b} \det\big(Q|_{\{a,b\},\{j,k\}}\big)\,\langle 0|\,\gamma_a\gamma_b\,|0\rangle \tag{3.71}$$

$$= i\sum_{l=1}^{n} \det\big(Q|_{\{2l-1,2l\},\{j,k\}}\big)\,\langle 0|\,iZ_l\,|0\rangle \tag{3.72}$$

$$= -\sum_{l=1}^{n} \det\big(Q|_{\{2l-1,2l\},\{j,k\}}\big). \tag{3.73}$$

Similarly,

$$\Gamma^{(l)}_{jk} = i\sum_{l'=1}^{n} \det\big(Q|_{\{2l'-1,2l'\},\{jk\}}\big)\,\langle 0|\,iX_lZ'_lX_l\,|0\rangle \tag{3.74}$$

$$= \det\big(Q|_{\{2l-1,2l\},\{jk\}}\big) - \sum_{l'\neq l} \det\big(Q|_{\{2l'-1,2l'\},\{jk\}}\big). \tag{3.75}$$

Therefore

$$\frac{\Gamma^{(l)}_{jk} - \Gamma^{(0)}_{jk}}{2} = \det\big(Q|_{\{2l-1,2l\},\{jk\}}\big). \tag{3.76}$$

Setting $j = 1$ and expanding, we have

$$C_{lk} \coloneqq \frac{\Gamma^{(l)}_{1k} - \Gamma^{(0)}_{1k}}{2} = Q_{2l-1,\,1}Q_{2l,\,k} - Q_{2l,\,1}Q_{2l-1,\,k}. \tag{3.77}$$

For clarity, we first consider the error-free case, where $\eta_{\mu\nu} = 0$. Then our matrix of unsigned entries of Q allows us to compute four guesses for the value of C_{lk} (recalling that we fixed $s_{\mu,\,1} = +1$):

$$D^{\pm\pm}_{lk} \coloneqq t_{2l-1}t_{2l}\Big(Q_{2l-1,\,1}(\pm Q_{2l,\,k}) - Q_{2l,\,1}(\pm Q_{2l-1,\,k})\Big). \tag{3.78}$$

If none of the four $D^{\pm\pm}_{lk}$ values are zero, then they are all distinct, and one of them exactly equals C_{lk}. We would therefore fix the signs of $Q_{2l,\,k}$ and $Q_{2l-1,\,k}$ in terms of $t_{2l-1}t_{2l}$. The set of matrices for which any $D^{\pm\pm}_{lk} = 0$ has measure 0, and can safely be ignored.

In the case *with* errors, we have instead only approximations of $D^{\pm\pm}_{lk}$, given by

$$\tilde{D}^{\pm\pm}_{lk} \coloneqq t_{2l-1}t_{2l}\Big(\tilde{Q}_{2l-1,\,1}(\pm\tilde{Q}_{2l,\,k}) - \tilde{Q}_{2l,\,1}(\pm\tilde{Q}_{2l-1,\,k})\Big). \tag{3.79}$$

Furthermore, we only have an estimate $\tilde{C}_{lk}$ of C_{lk} up to additive error ϵ. Obtaining these for each C_{lk} with high probability requires $\mathcal{O}(n^2\log(n)/\epsilon^2)$ queries by the Hoeffding inequality and a union bound. The true value $\tilde{D}^{s_{2l-1,\,k}s_{2l,\,k}}_{lk}$ satisfies

$$\left|\tilde{D}^{s_{2l-1,\,k}s_{2l,\,k}}_{lk} - \tilde{C}_{lk}\right| \leq 4\eta + \epsilon + \mathcal{O}(\eta^2), \tag{3.80}$$

using $|Q_{\mu\nu}| \leq 1$. Conversely, incorrect choices of sign are such that

$$\left|\tilde{D}^{\pm\pm}_{lk} - \tilde{C}_{lk}\right| \geq |2F - (4\eta + \epsilon)|, \tag{3.81}$$

where $F \in \{|Q_{2l,\,1}Q_{2l-1,\,k}|,\, |Q_{2l-1,\,1}Q_{2l,\,k}|,\, |C_{lk}|\}$ for the 3 incorrect cases. If $4\eta + \epsilon < \min\{|Q_{2l,\,1}Q_{2l-1,\,k}|,\, |Q_{2l-1,\,1}Q_{2l,\,k}|,\, |C_{lk}|\}$, then we are guaranteed that the true choice of $\tilde{D}^{\pm\pm}_{lk}$ is the closest to $\tilde{C}_{lk}$. In appendix B.1, we prove that $\mathbb{P}(F < \kappa) = \mathcal{O}(\kappa^{1/3})$. Taking a union bound over the $2n^2$ such calculations that we perform, we have that $\kappa = \mathcal{O}(n^{-6})$ suffices to ensure that all the quantities are larger than κ with high probability. Therefore, $\epsilon, \eta = \mathcal{O}(n^{-6})$ suffices to successfully learn the matrix s with high probability, fixing the signs up to overall row-wise errors.

3.2.4 Fixing row-wise sign errors

Assuming the above protocol succeeds, we have some matrix $\overline{Q}$, where $\overline{Q}_{\mu\nu} = t_\mu Q_{\mu\nu}$.

For any set of rows R where $t_\mu = -1$, $M_{\overline{Q}}$ is straightforwardly related to M_Q. Let $M_R \coloneqq M_Q \gamma_R$. Then M_R acts as

$$M_R \gamma_\mu M_R^\dagger = M_Q \gamma_R \gamma_\mu \gamma_R^\dagger M_Q^\dagger \tag{3.82}$$

$$= (-1)^{|R| - \mathbb{1}(\mu \in R)} M_Q \gamma_\mu M_Q^\dagger \tag{3.83}$$

$$= (-1)^{|R|} \sum_\nu (-1)^{\mathbb{1}(\mu \in R)} Q_{\mu\nu}. \tag{3.84}$$

If $|R|$ is even, then $M_{\overline{Q}} = M_R$. If $|R|$ is odd, then $M_{\overline{Q}} = M_{R^c}$, where R^c is the complement of R in $[2n]$. Let P be R or R^c corresponding to the two cases.

A single-shot measurement suffices to distinguish P. Note that knowledge of $\overline{Q}$ is sufficient to write down an exact, poly-sized circuit of two-qubit Gaussian operators which implements $M_{\overline{Q}}^\dagger$ [39]. A measurement of $(M_{\overline{Q}}^\dagger \otimes I)\,|\psi\rangle$ with respect to the basis $\{(\gamma_S M_Q^\dagger \otimes I)\,|\psi\rangle\}$ has outcome probabilities

$$\mathbb{P}(S) = \left| \langle\psi| \left(M_Q \gamma_S M_{\overline{Q}}^\dagger \otimes I\right) |\psi\rangle \right|^2 \tag{3.85}$$

$$= \left| 2^{-n} \operatorname{Tr}\left(M_Q \gamma_S \gamma_P^\dagger M_Q^\dagger\right) \right|^2 \tag{3.86}$$

$$= \begin{cases} 1 & S = P, \\ 0 & \text{otherwise.} \end{cases} \tag{3.87}$$

The outcome of this measurement allows the correction of the remaining errors in the sign of each row of $\overline{Q}$, leaving the true matrix Q up to additive error η.

In Appendix appendix B.2, we prove that, with high probability, this is sufficient to obtain an estimate $M_{Q'}$ of M_Q satisfying

$$D(M_Q,\, M_{Q'}) = \mathcal{O}(n^3 \eta). \tag{3.88}$$

3.2.5 Complexity of learning Gaussian operations

The total number of queries is easily seen to be $\mathcal{O}(n \log(n)/\eta^2 + n^2 \log(n)/\epsilon^2 + 1)$. Setting $\eta = \epsilon = \mathcal{O}(n^{-6})$, we obtain an estimate of Q up to additive error η using $\mathcal{O}((n/\eta)^2 \log(n))$ queries. Thus, the learning task may be completed in (high-degree) polynomial query complexity.

3.3 The matchgate hierarchy

3.3.1 Defining the matchgate hierarchy

In the spirit of the Clifford Hierarchy, we introduce an infinite family of gates $\mathcal{M}_k$ satisfying $\mathcal{M}_k \subseteq \mathcal{M}_{k+1}$ for each k, which we call the *Matchgate Hierarchy*.

Let $\Gamma_1 \coloneqq \{\gamma_\mu : \mu \in [2n]\}$. The γ_μ play the role of the Pauli generators $\{X_i, Z_i\}$ in the Clifford hierarchy.

Definition 3.3.1. *The k^{th} level of the Matchgate hierarchy $\mathcal{M}_k$ is defined recursively by*

$$\mathcal{M}_1 \coloneqq \{M \in U(2^n) : M = \sum_\mu a_\mu \gamma_\mu; a_\mu \in \mathbb{R}\}, \tag{3.89}$$

$$\mathcal{M}_k \coloneqq \{M \in U(2^n) : M\Gamma_1 M^\dagger \subseteq \mathcal{M}_{k-1}\}. \tag{3.90}$$

The condition that $\mathcal{M}_1$ gates are unitary immediately gives $\sum_\mu a_\mu^2 = 1$. We note that the set $\mathcal{M}_2$ consists of gates M for which, for each $\mu \in [2n]$,

$$M\gamma_\mu M^\dagger = \sum_{\nu=1}^{2n} Q_{\mu\nu}\gamma_\nu, \tag{3.91}$$

for some $Q = (Q_{\mu\nu}) \in \mathbb{R}^{2n\otimes 2n}$.

We may evaluate the anti-commutator of $M\gamma_\mu M^\dagger$ and $M\gamma_\nu M^\dagger$ in two ways:

$$\begin{aligned} \{M\gamma_\mu M^\dagger, M\gamma_\nu M^\dagger\} &= M\{\gamma_\mu, \gamma_\nu\}M^\dagger \\ &= \delta_{\mu\nu} MIM^\dagger = \delta_{\mu\nu}, \end{aligned} \tag{3.92}$$

$$\begin{aligned} \{M\gamma_\mu M^\dagger, M\gamma_\nu M^\dagger\} &= \sum_{\sigma,\tau} Q_{\mu\sigma}Q_{\nu\tau}\{\gamma_\sigma, \gamma_\tau\} \\ &= \sum_\sigma Q_{\mu\sigma}Q_{\nu\sigma} = (QQ^T)_{\mu\nu}. \end{aligned} \tag{3.93}$$

Equating eqs. (3.92) and (3.93) implies $(QQ^T)_{\mu\nu} = \delta_{\mu\nu}$, *i.e.* $Q \in O(2n)$. Such operations are sometimes referred to as *extended Gaussian operations*. As in the Clifford case, it is not straightforward to characterise the higher levels of the hierarchy. However, there is a link between the Matchgate Hierarchy and the Clifford hierarchy:

Lemma 3.3.1. *For any integer $k \geq 1$, $\mathcal{C}_k \subseteq \mathcal{M}_{k+1}$.*

Proof. We proceed by induction.

For the base case, note that $\mathcal{C}_1 \equiv \mathcal{P}$ is the set of Pauli strings, which exactly coincides with the set of Majorana monomials $\{\gamma_S : S \subseteq [2n]\}$ up to factors of ± 1 or $\pm i$. Since γ_S is easily seen to be a Gaussian operation, we have $\mathcal{C}_1 \subseteq \mathcal{M}_2$.

For the inductive step, note that $\gamma_\mu \in \mathcal{C}_1$ for any $\mu \in [2n]$. Therefore, for any $C \in \mathcal{C}_k$, we have $C\gamma_\mu C^\dagger \in \mathcal{C}_{k-1}$ by definition of the Clifford hierarchy. By the inductive hypothesis, $\mathcal{C}_{k-1} \subseteq \mathcal{M}_k$. Since this is true for all $\mu \in [2n]$, we have $C \in \mathcal{M}_{k+1}$. □

A noteworthy implication of lemma 3.3.1 is that SWAP $\in \mathcal{M}_3$. SWAP is a universality-enabling gate for matchgates and, in some sense, the maximally non-matchgate two-qubit gate. This is of potential interest for the fault-tolerant implementation of quantum protocols based on a "Matchgate + SWAP" universal gate set.

3.3.2 Learning matchgate hierarchy operations

The learning algorithm for Gaussian operations can be generalised to learning any operation from the $\mathcal{M}_k$ hierarchy. The proof requires two lemmas, given in [31], which trivially extend to the current setting: the former bounds the phase-sensitive distance between transformed Majorana monomials given bounds on the transformed generators; the latter bounds the phase-insensitive distance between unitaries given bounds on the transformed Majorana monomials.

Lemma 3.3.2 (Lemma 18, [31]). *For unitaries U_1, U_2, if for all $\mu \in [2n]$*

$$D^+(U_1\gamma_\mu U_1^\dagger, U_2\gamma_\mu U_2^\dagger) \leq \delta, \tag{3.94}$$

then for any $S \subset [2n]$

$$D^+(U_1\gamma_S U_1^\dagger, U_2\gamma_S U_2^\dagger) \leq 2n\delta. \tag{3.95}$$

Proof. The proof follows by induction on the number of Majorana operators in γ_S and the triangle inequality. □

Lemma 3.3.3 (Lemma 17, [31]). *For unitaries U_1, U_2, if for all $S \subset [2n]$*

$$D^+(U_1\gamma_S U_1^\dagger, U_2\gamma_S U_2^\dagger) \leq 2n\delta, \tag{3.96}$$

then

$$D(U_1, U_2) \leq 2n\delta. \tag{3.97}$$

Proof. The proof follows from the fact that the group of Paulis is a 1-design and that, under the Jordan-Wigner representation, each γ_S is simply a Pauli string up to a factor of i. □

Proposition 3.3.1. *Given oracle access to an unknown operation $M \in \mathcal{M}_k$ for $k \geq 2$ and its conjugate $M^\dagger$, an estimate M' of M satisfying $D(M, M') = \mathcal{O}(n^3\eta)$ can be determined with high probability with query complexity $\tilde{\mathcal{O}}(2^{5(k-2)} \cdot n^{2(k-2)} \cdot (n/\eta)^2)$ for any $\eta = \mathcal{O}(n^{-(6-k)}), o(1)$.*

Proof. The proof is by induction, where the base case for $k = 2$ is given by proposition 3.2.1.

For the inductive step, suppose we have a learning algorithm for members of $\mathcal{M}_k$, which outputs estimates with phase-insensitive precision δ. Let $M \in \mathcal{M}_{k+1}$, and for any $\mu \in [2n]$, let $M\gamma_\mu M^\dagger = \alpha_\mu M_\mu$ for some $\alpha_\mu = \pm 1$. We may express M_μ in the Hermitian basis of Majorana monomials $\overline{\gamma}_S \coloneqq i^{f(|S|)}\gamma_S$, where the factors of i are chosen to make $\overline{\gamma}_S^\dagger = \overline{\gamma}_S$. Let $M_\mu = \sum_S c_S \overline{\gamma}_S$. Since M_μ is Hermitian, $c_S \in \mathbb{R}$ for each S, and we choose α_μ so that $c_\emptyset \in \mathbb{R}_+$.

The M_μ are elements of $\mathcal{M}_k$, so we may (phase-insensitively) learn them, obtaining estimates M''_μ satisfying $D(M_\mu, M''_\mu) \leq \delta$. We aim to obtain a phase-sensitive estimate by leveraging the Hermiticity of M_μ.

Let $M''_\mu = \sum c''_S \overline{\gamma}_S$. The vector of coefficients c'' may be written $c'' = \lambda c + \chi c^\perp$, for some $c^\perp$ satisfying $\langle c, c^\perp \rangle = 0$ for the usual complex inner product $\langle \cdot, \cdot \rangle$. It is easy to check that $D(M_\mu, M''_\mu) \leq \delta$ implies $|\langle c, c'' \rangle|^2 \geq 1 - \delta^2$, which in turn implies $|\lambda|^2 \geq 1 - \delta^2$. We need to modify the phase of M''_μ to obtain a phase-sensitive estimate. The optimal phase to multiply by is $-\arg(\lambda)$, but this is unknown. We approximate λ by $\theta \coloneqq -\arg(\lambda c_S + \chi c_S^\perp)$ for any S. Some elementary trigonometry gives

$$\left|\arg(\lambda c_S + \chi c_S^\perp) - \arg(\lambda)\right| \leq \frac{\delta}{\sqrt{1-\delta^2}}. \tag{3.98}$$

Then we may compute the relevant real trace for $D^+(M_\mu,\, e^{-i\theta}M''_\mu)$:

$$\begin{aligned}
\mathrm{Re}\left(\mathrm{Tr}\big(M_\mu^\dagger e^{-i\theta}M''_\mu\big)\right) &= \mathrm{Re}\left(\sum_{S,\,S'} e^{-i\theta}c_S c''_{S'}\,\mathrm{Tr}(\overline{\gamma}_S\overline{\gamma}_{S'})\right) && (3.99)\\
&= d\,\mathrm{Re}\left(e^{-i\theta}\langle c,\, c''\rangle\right) && (3.100)\\
&= d\,\mathrm{Re}\left(|\lambda|e^{i(\arg(\lambda)-\theta)}\right) && (3.101)\\
&= d|\lambda|\cos(\arg(\lambda)-\theta) && (3.102)\\
&\geq d\sqrt{1-\delta^2}\cdot\left(1-\frac{\delta^2}{1-\delta^2}\right) && (3.103)\\
&= d\frac{1-2\delta^2}{\sqrt{1-\delta^2}} \geq d(1-2\delta^2). && (3.104)
\end{aligned}$$

Substituting into the definition of D^+, we see that

$$D^+(M_\mu,\, e^{-i\theta}M''_\mu) \leq \sqrt{2}\delta \quad \forall\ \mu \in [2n]. \tag{3.105}$$

Now let M'' be the operator such that $M''\gamma_\mu M''^\dagger = M''_\mu$ for all μ. Also, let M' be such that $M'\gamma_\mu M'^\dagger = M_\mu$, i.e. all the phases α_μ are set to $+1$. Similarly to the level-2 case, $M = M'\gamma_R$ for some set $R \subseteq [2n]$. By lemma 3.3.2 and eq. (3.105), we see that

$$D^+(M'\gamma_S M'^\dagger,\, e^{-i\theta}M''\gamma_S M''^\dagger) \leq 2\sqrt{2}n\delta \quad \forall\ S \subseteq [2n]. \tag{3.106}$$

By lemma 3.3.3, we then have

$$D(M',\, e^{-i\theta}M'') \leq 2\sqrt{2}n\delta. \tag{3.107}$$

We may correct our estimate from M' to M by preparing $(M^\dagger M''\otimes I)\,|\psi\rangle$ and measuring with respect to the basis $\{(\gamma_S^\dagger \otimes I)\,|\psi\rangle\}$. Then the probability of obtaining outcome R is:

$$\begin{aligned}
\mathbb{P}(R) &= \left|\langle\psi|\,(\gamma_R\otimes I)(\gamma_R^\dagger M'^\dagger M'')\,|\psi\rangle\right|^2 && (3.108)\\
&= \left|d^{-1}\,\mathrm{Tr}\big(M'^\dagger M''\big)\right|^2 && (3.109)\\
&\geq 1-8n^2\delta^2, && (3.110)
\end{aligned}$$

using (3.107). The outcome of the measurement allows us to correct our estimate to $e^{-i\theta}M''\gamma_R$, which, with high probability, satisfies

$$D(M,\, e^{-i\theta}M''\gamma_R) \leq 2\sqrt{2}n\delta. \tag{3.111}$$

At the k^{th} level, the $2\sqrt{2}n$ factor accumulates $k-2$ times. We therefore need to learn the operators to precision $\delta = \eta/(2\sqrt{2}n)^{k-2}$. For each additional level in the hierarchy, we must learn $2n$ operators from the previous level, requiring $4n$ queries to prepare them. Therefore, the query complexity at level k is given by

$$\tilde{\mathcal{O}}\left((4n)^{k-2}\cdot\left(\frac{n}{\eta/(2\sqrt{2}n)^{k-2}}\right)^2\right) \tag{3.112}$$

$$= \tilde{\mathcal{O}}\big(2^{5(k-2)}\cdot n^{2(k-2)}\cdot(n/\eta)^2\big), \tag{3.113}$$

as claimed. □

Chapter 4

Parameterized quantum algorithms for closest string problems

String processing is one of the oldest branches of theoretical computer science, yet it remains an active research field, particularly with the rapid growth of available data. Bioinformatics is a key motivating area: rapid advances in sequencing technology and global-scale viral monitoring have produced vast amounts of genomic data [94]. Extracting meaningful insights from this information requires fast algorithms.

Given a set of strings – often representing DNA, mRNA, or protein sequences – important tasks are to find a new string that lies within a bounded distance to each input, or to some substring of each input. These are the Closest String Problem (CSP) and the Closest Substring Problem (CSSP) [70], where distance is measured by metrics such as the Hamming or Levenshtein distances [67]. Formal definitions are given in section 4.1.

Solutions to closest string problems can act as *consensus strings*, serving as compact representations of the input set and proxies for a full multiple sequence alignment (MSA). A complete MSA requires $\mathcal{O}(n^k)$ time to align k strings of length n, which is prohibitive for large data sets. Computing a consensus string is cheaper while still informative for tasks such as DNA binding site analysis [95]. Indeed, the widely used centre star heuristic iteratively aligns each sequence against a tentative closest string, and the final alignment is computed from the results [96]. The quality of the final alignment directly depends on the centre string used; using the CSP solution guarantees a good alignment.

Both the CSP and the CSSP are NP-hard, even for the simplest case of binary alphabets under Hamming distance. The lens of parameterized complexity has enabled more nuanced algorithmic approaches [97–102], revealing structure-dependent regimes

of efficiency such as few input strings k or small distance thresholds d. The choice of metric also affects hardness: for example, the CSP with fixed k is polynomial-time soluble under Hamming distance [97] but not under Levenshtein distance [98]. A detailed parameterized study across the various distance metrics and parameters is therefore warranted.

Moreover, recent progress in quantum algorithms and hardware motivates the exploration of quantum speedups for such string problems. Previous work includes quantum algorithms for sequence comparison [75], pattern matching [78, 80], seed-and-extend alignment [74] and *de novo* assembly [20]; see also [18, 19, 76, 77, 79, 103–107].

In this chapter, we develop parameterized quantum algorithms for the CSP and CSSP, analysing their runtime with respect to the alphabet size σ, the number of inputs k, and the distance threshold d. Their runtimes are given in propositions 4.0.1 to 4.0.4:

Proposition 4.0.1 (CSP, any distance)**.** *There exists a quantum algorithm for solving the CSP under any desired distance metric, with runtime* $\tilde{\mathcal{O}}\Big(\sigma^{n/2} \cdot \sqrt{k} \cdot D(n)\Big)$*, where* $D(n)$ *is the* classical *time required to compute the distance between 2 strings of length* n*. In particular,* $D(n) = \mathcal{O}(n)$ *for the Hamming distance and* $\mathcal{O}(n^2)$ *for the Levenshtein distance.*

Proposition 4.0.2 (CSP, Hamming distance)**.** *There exists a quantum algorithm for solving the CSP under the Hamming distance, with runtime* $\tilde{\mathcal{O}}\big(kn + k^2 \cdot d^{(d+5)/2}\big)$*.*

Proposition 4.0.3 (CSP, Levenshtein distance)**.** *There exists a quantum algorithm for solving the CSP under the Levenshtein distance, with runtime* $\tilde{\mathcal{O}}\big(\sigma \cdot 2^k \cdot T(n)^{2k}\big)$*, where* $T(n)$ *satisfies* $n/e \leq T(n) < n$ *for every* n*.*

Proposition 4.0.4 (CSSP, Hamming distance)**.** *There exists a quantum algorithm for solving the CSSP under the Hamming distance, with runtime* $\tilde{\mathcal{O}}\Big(\sigma^{d(\log(d)+2)/2} \cdot (kn)^{\log(d)/2} \cdot k^2 n^3\Big)$*.*

The best parameterized classical algorithms for the CSP run in time $\mathcal{O}\big(\sigma \cdot 2^k \cdot n^{2k}\big)$ [98], $\mathcal{O}\big(kn + k \cdot d^{d+1}\big)$ [97], or $\mathcal{O}\big(kn + kd \cdot M^d\big)$ for $M = \Theta(\sigma)$ [102, 108, 109], with each suited to different parameter regimes. Proposition 4.0.2 therefore represents a quadratic improvement in the dominant scaling factor for the CSP under Hamming distance when the alphabet size is unbounded. Proposition 4.0.3 is a non-trivial

speedup, but the improvement is not polynomial in the size of the input. Finally, proposition 4.0.1 is a quadratic speedup over trivial classical searches, which are the best methods when k and d are both large. We also show that for the special case of binary alphabets, proposition 4.0.1 is tight under the quantum analogue of the Strong Exponential Time Hypothesis (SETH) [110] formulated by Buhrman et al. [111].

The best classical algorithm for the CSSP runs in time $\mathcal{O}\big(\sigma^{d(\log(d)+2)}(kn)^{\log(d)}k^3n^4\big)$. Proposition 4.0.4 gives a quadratic improvement in the dominant scaling factors.

4.1 String processing problems

A string s is a finite sequence over a finite alphabet Σ. For bioinformatics applications, this will often be the DNA-base alphabet $\{A, C, G, T\}$. Let $\sigma = |\Sigma|$. The length of s is $|s|$, and for $0 \leq i < |s|$, its i^{th} character is $s[i]$. For $0 \leq i < j < |s|$, $s[i]s[i+1]\dots s[j]$ is a *substring* of s and is denoted by $s[i, j]$. The set of strings over Σ of any length is Σ^*; those of length exactly n is Σ^n; those of length at most n is $\Sigma^{\leq n}$.

We use $d(\cdot, \cdot)$ to denote the distance between two strings. The Hamming distance $d_H(\cdot, \cdot)$ between two strings s and s' of equal length is the number of positions i for which $s[i] \neq s'[i]$. The Levenshtein distance [67] $d_L(\cdot, \cdot)$ between s and s' is the minimum number of characters that must be inserted, deleted, or substituted to transform s into s'. For a formal definition based on the notion of alignments, see, e.g., Chapter 6 of [112].

We formally state the CSP as follows:

Problem 4.1.1 (Closest String Problem)**.** *Given k strings $s_1, \dots, s_k \in \Sigma^{\leq n}$ and a threshold distance d, find a string $s \in \Sigma^*$ such that $\max_i\{d(s, s_i)\} \leq d$.*

When $d(\cdot, \cdot)$ is the Hamming distance, we require all strings to be of equal length for the problem to be well-defined. Several parameterized algorithms exist for different parameter combinations. Nicolas and Rivals [98] achieve a runtime of $\mathcal{O}\big(\sigma \cdot 2^k \cdot n^{2k}\big)$ via dynamic programming, placing the CSP in XP with respect to k. Note that d is allowed to be unbounded in this case. Gramm et al. [97] give an $\mathcal{O}\big(kn + k \cdot d^{d+1}\big)$ algorithm under Hamming distance, placing the CSP in FPT with respect to d. Variants with bounded alphabet size achieve $\mathcal{O}\big(kn + kd \cdot M^d\big)$ for $M = \Theta(\sigma)$ [102, 108, 109].

The related CSSP generalises the CSP, being equivalent when $L = n$:

Problem 4.1.2 (Closest Substring Problem)**.** *Given k strings $s_1, \dots, s_k \in \Sigma^{\leq n}$ and a threshold distance d, find a length-L string $s \in \Sigma^L$ such that for all $i = 1, \dots, k$, there exists a length-L substring w_i of s_i with $d(s, w_i) \leq d$.*

Marx [113] provides an XP algorithm for the CSSP under Hamming distance with runtime $\mathcal{O}\big(\sigma^{d(2+\log(d))}(kn)^{4+\log(d)}\big)$.

4.2 Quantum algorithms for the closest string problem

We present three quantum algorithms for the CSP:

A generic nested search, applicable for any desired distance metric. It is suitable for cases with many input strings, each not too long. This may be the case, for example, when seeking a conserved fragment of a gene across different strains of pathogens such as SARS-CoV-2.

A random-walk search, based on a modified version of Gramm et al.'s algorithm [97]. It is best suited to closely related input strings, and, for a fixed distance threshold, the runtime scales only linearly with both the number and the length of the input strings.

A dynamic-programming algorithm, relevant for Levenshtein distance. It is suited to cases where there are few input strings and no bound on the threshold distance is known – at present, this is most likely of theoretical interest only.

4.2.1 A trivial quantum search algorithm

A simple baseline is obtained by applying quantum minimum finding twice. The outer search minimises over candidate strings $s \in \Sigma^{\leq(d+n)}$ for the Levenshtein distance or $s \in \Sigma^n$ for the Hamming distance. The function to be minimised is the maximum distance between s and any of the input strings $s_1, \ldots, s_k$. Instead of checking each of these distances sequentially, we use a quantum maximum finding.

Formally, we may write the search functions:

$$\begin{aligned} f : S &\to \mathbb{Z}_+ & g_s : [k] &\to \mathbb{Z}_+, \\ s &\to \max_{i\in[k]} d(s,\, s_i) & i &\to d(s,\, s_i), \end{aligned} \tag{4.1}$$

where S is either $\Sigma^{\leq(d+n)}$ or Σ^n.

The inner search of g_s runs in time $T_I = \tilde{\mathcal{O}}\Big(\sqrt{k}\cdot T_d\Big)$, where T_d is cost of computing $d(\cdot,\,\cdot)$. $T_d = \mathcal{O}(n)$ for Hamming distance and $\mathcal{O}(n^2)$ for Levenshtein distance. The

outer search of f runs in time $\tilde{\mathcal{O}}\Big(\sqrt{|S|}\cdot T_I\Big) = \tilde{\mathcal{O}}\Big(\sigma^{n/2}\cdot\sqrt{k}\cdot T_d\Big)$. We therefore obtain our first result, proposition 4.0.1:

Proposition 4.0.1 (CSP, any distance)**.** *There exists a quantum algorithm for solving the CSP under any desired distance metric, with runtime* $\tilde{\mathcal{O}}\Big(\sigma^{n/2}\cdot\sqrt{k}\cdot D(n)\Big)$*, where* $D(n)$ *is the* classical *time required to compute the distance between 2 strings of length* n*. In particular,* $D(n) = \mathcal{O}(n)$ *for the Hamming distance and* $\mathcal{O}(n^2)$ *for the Levenshtein distance.*

The search over all length-n strings will usually dominate the runtime. However, it scales very favourably with respect to k and could therefore be feasible in very high-volume data scenarios.

In the classical setting, it can be shown that exhaustive search is essentially optimal for the CSP under Hamming distance in the binary case [114], assuming SETH holds. This widely held computational hardness assumption implies the optimality of many known algorithms. The proof follows via a reduction from (q, k)-CNF SAT, i.e. the problem of finding satisfying assignments to q-ary CNF formulae, to the *Remotest String Problem* (RSP), a natural analogue of the CSP where the goal is to find a string far from all inputs. For binary alphabets, the CSP and the RSP are equivalent by flipping all bit strings in the inputs. We restate the result in Theorem 4.2.1:

Theorem 4.2.1 (Based on Theorems 1.1 & 1.2 from [114])**.** *An algorithm that solves the CSP for binary alphabets under Hamming distance in time* $\mathcal{O}\big(2^{(1-\epsilon)n}\mathsf{poly}(k)\big)$ *implies the existence of an algorithm for* (q, k)*-SAT on* N *variables with* M *clauses which runs in time* $\mathcal{O}\big(2^{(1-\epsilon/2)N}\mathsf{poly}(M)\big)$*.*

The framework of *Quantum Strong Exponential Time Hypotheses (QSETH)* of Buhrman et al. [111] allows us to make the *Basic* (q,k)*-CNF SAT QSETH* conjecture:

Conjecture 4.2.1 (Basic (q,k)-CNF SAT QSETH)**.** *A quantum algorithm cannot, given an input* C *from the set of* (q,k)*-CNF formulae on* N *variables and* M *clauses, decide whether there exists* $x \in [q]^N$ *such that* $C(x) = 1$ *in time* $\mathcal{O}\Big(q^{\frac{N}{2}(1-\delta)}\mathsf{poly}(M)\Big)$*, for any* $\delta > 0$*.*

Using the same reduction that leads to Theorem 4.2.1, we see that Proposition 4.0.1 is tight for binary alphabets unless Basic (q,k)-CNF SAT QSETH is false:

Proposition 4.2.1. *No quantum algorithm solves the binary alphabet CSP in time* $\mathcal{O}\big(2^{(1-\epsilon)\frac{n}{2}}\mathsf{poly}(k)\big)$ *unless conjecture 4.2.1 is false.*

Hence, only parameterized approaches can yield further asymptotic gains.

4.2.2 A quantum random walk algorithm for the CSP under Hamming distance

We next quantise the algorithm of Gramm et al. [97], which targets the setting where threshold distance d is small. Pre-processing removes any position where all k strings agree; if more than kd remain, no solution exists. This takes time $\mathcal{O}(kn)$, which is usually negligible.

The algorithm recursively explores candidate strings within Hamming distance d of s_1. At each level, if the candidate is not a solution, then there is a minimal index i^* for which $d_H(s, s_{i^*}) > d$. Then the $d+1$ candidates for the next level are formed by replacing a character in s by the corresponding character in s_{i^*} for $d+1$ positions where $s \neq s_{i^*}$. Since, by hypothesis, $d_H(\hat{s}, s_{i^*}) \leq d$, there are at most d positions where both $s \neq s_{i^*}$ and $\hat{s} \neq s_{i^*}$. Therefore, at least one of the modifications must decrease $d_H(s, \hat{s})$, moving the candidate closer to the proper solution. After d such rounds, a solution must be found if one exists.

The algorithm, in the worst case, searches over a *perfect* $(d+1)$*-ary tree* $\mathcal{T}$: a tree where all interior nodes have $d+1$ children and all leaves are at the same depth $d+1$. We therefore call the algorithm `TreeSearch`.

As a further optimisation, we note that if, at any layer, the candidate solution's distance to any input is strictly greater than d plus the number of rounds remaining, then no solution is possible on that branch of the recursion. The algorithm is described by the pseudocode in algorithm 2. The inputs at the first level of recursion are $s = s_1$ and $\Delta d = d$.

Gramm et al. compute the algorithm's runtime, which we restate in theorem 4.2.2.

Theorem 4.2.2 ([97])**.** *`TreeSearch`, described in algorithm 2, with (pre-processed) global variables $s_1, \ldots, s_k$, threshold distance d and inputs $s = s_1$ and $\Delta d = d$, solves the CSP under the Hamming distance in time $\mathcal{O}\big(k \cdot d^{d+1}\big)$.*

Let Ω be the set of nodes of $\mathcal{T}$. We may denote some node of Ω by a d-dimensional, integer-valued vector $\mathbf{j} \in \{0, \ldots, d+1\}^d$. The root is $(0, \ldots, 0)$; for $\mathbf{j}$ in the l^{th} layer, $1 \leq l \leq d$, the first l coordinates are non-zero, corresponding to which branch was taken at layer l. Then we can write

$$\Omega = \Big\{\mathbf{j} = (j_1, \ldots, j_l, 0, \ldots, 0) \in \{0, \ldots, d+1\}^d : 0 \leq l \leq d; 1 \leq j_i \leq d\Big\}. \tag{4.2}$$

Algorithm 2 `TreeSearch`$(s,\, \Delta d)$ [97]

Require: (Global Variables) $s_1, \ldots, s_k \in \Sigma^{kd}$, threshold distance d.

Require: Candidate string s and integer Δd.
Output: String $\hat{s}$ with $\max_i d_H(\hat{s},\, s_i) \leq d$ and $d_H(\hat{s},\, s) \leq \Delta d$, if one exists.

if $\Delta d < 0$ **then**
  return "Not found"
if $d_H(s,\, s_i) > d + \Delta d$ for some $i \in [1\,..\,k]$ **then**
  return "Not found"
if $d_H(s,\, s_i) \leq d$ for all $i \in [1\,..\,k]$ **then**
  return s
$i^* \leftarrow \min\{i : d_H(s,\, s_i) > d\}$
$P \leftarrow \{p : s[p] \neq s_{i^*}[p]\}$
$P' \leftarrow P[1\,..\,d+1]$
for all $p \in P'$ **do**
  $s' \leftarrow s$
  $s'[p] \leftarrow s_{i^*}[p]$
  $s_{\text{new}} \leftarrow$ `TreeSearch`$(s',\, \Delta d - 1)$
  if $s_{\text{new}} \neq$ "Not found" **then**
    return s_{new}
return "Not found"

To use quantum search via random walks [47], we first define a lazy, reversible Markov chain P on Ω as follows:

$$\text{If } \mathbf{j} \text{ is neither root nor leaf,} \quad p_{\mathbf{j}\mathbf{j}'} = \begin{cases} \frac{1}{2} & \text{if } \mathbf{j}' = \mathbf{j}, \\ \frac{1}{4(d+1)} & \text{if } \mathbf{j}' \text{ is a child of } \mathbf{j}, \\ \frac{1}{4} & \text{if } \mathbf{j}' \text{ is the parent of } \mathbf{j}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

$$\text{If } \mathbf{j} \text{ is a leaf,} \quad p_{\mathbf{j}\mathbf{j}'} = \begin{cases} \frac{1}{2} & \text{if } \mathbf{j}' = \mathbf{j}, \\ \frac{1}{2} & \text{if } \mathbf{j}' \text{ is the parent of } \mathbf{j}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

$$\text{If } \mathbf{j} \text{ is the root,} \quad p_{\mathbf{j}\mathbf{j}'} = \begin{cases} \frac{1}{2} & \text{if } \mathbf{j}' = \mathbf{j}, \\ \frac{1}{2(d+1)} & \text{if } \mathbf{j}' \text{ is a child of } \mathbf{j}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

The set of edges of P is

$$E = \{(\mathbf{j}, \mathbf{j}') \in \Omega \times \Omega : \mathbf{j} \text{ in layer } l,\ \mathbf{j}' \text{ in layer } l+1,\ j_i = j'_i \text{ for } i = 1, \ldots, l\}. \quad (4.6)$$

We may bound the eigenvalue gap of P:

Proposition 4.2.2. *The eigenvalue gap $\delta(P)$ of P satisfies $\delta(P) > 2^{-9}d^{-2}$.*

Proof. See appendix C.3. □

We now define the data structure and compute the costs of setup, update and checking required by the quantum search routine.

Data Structure. By analysing algorithm 2, we see that the data we need to track is the history of modifications made to the candidate string. Given a list of indices $i_1, \ldots, i_l \in \{1, \ldots, k\}$ representing the donor strings, and positions $k_1, \ldots, k_l \in \{1, \ldots, kd\}$ representing the character position that was substituted, we can build the candidate string s from s_1 in time $\mathcal{O}(l) = \mathcal{O}(d)$. For $\mathbf{j} = (j_1, \ldots, j_l, 0, \ldots, 0)$, we therefore take $D(\mathbf{j}) = \{(i_1, \ldots, i_l, 0, \ldots, 0),\ (k_1, \ldots, k_l, 0, \ldots, 0)\}$. This structure requires $\mathcal{O}(d \log(kd))$ qubits.

Setup. First, we prepare the state $|\Phi\rangle = \sum_{\mathbf{j}\in\Omega} \sqrt{\pi_{\mathbf{j}}} |\mathbf{j}\rangle$, which is a state-preparation operation on $\mathcal{O}(d \log(d))$ qubits. By, e.g., the results of Zhang et al. [115], this can be done in time $\mathcal{O}(d \log(d))$. We then need to implement the transformation

$$U_D : |\mathbf{j}\rangle |\mathbf{0}\rangle \mapsto |\mathbf{j}\rangle |D(\mathbf{j})\rangle . \quad (4.7)$$

In the worst case, this requires computing d modifications to the candidate string s. For each modification, we need to compare s to the k input strings, which takes time $\mathcal{O}(k \cdot kd)$. So the worst case complexity of U_D is $\tilde{\mathcal{O}}(k^2 d^2)$. We have then prepared $\sum_{\mathbf{j}\in\Omega} \sqrt{\pi_{\mathbf{j}}} |\mathbf{j}\rangle_D$. Finally, we append the state $|0\rangle_D \equiv |0\rangle |0\rangle$ at no cost. So $S = \tilde{\mathcal{O}}(k^2 d^2)$.

Update. We wish to implement the transformation:

$$|\mathbf{j}\rangle_D |\mathbf{0}\rangle_D \mapsto |\mathbf{j}\rangle_D \sum_{\mathbf{j}'} \sqrt{p_{\mathbf{j}\mathbf{j}'}} |\mathbf{j}'\rangle_D . \quad (4.8)$$

We first implement a diffusion-like operator that realises

$$|\mathbf{j}\rangle_D |\mathbf{0}\rangle_D \mapsto |\mathbf{j}\rangle_D \sum_{\mathbf{j}'} \sqrt{p_{\mathbf{j}\mathbf{j}'}} |\mathbf{j}'\rangle |\mathbf{0}\rangle . \quad (4.9)$$

This controlled-state-preparation on $\mathcal{O}(d \log(d))$ qubits can again be performed in time $\tilde{\mathcal{O}}(d \log(d))$. To update the data structure, we must compute $|\mathbf{j}'\rangle_D$ from $|\mathbf{j}\rangle_D$, where $\mathbf{j}'$

is either the parent or child of $\mathbf{j}$. To go to the parent, we set the last entries i_l, k_l to zero in $\mathcal{O}(1)$ depth. To go to a child, we must compute one further modification to the candidate string, in time $\tilde{\mathcal{O}}(k^2 d)$. So $U = \tilde{\mathcal{O}}(k^2 d)$.

Checking. It takes $\mathcal{O}(d)$ time to compute the candidate string and $\mathcal{O}(k^2 d)$ time to check it against all the input strings. So $C = \tilde{\mathcal{O}}(k^2 d)$.

Overall time complexity. Finally, we need to bound the probability of sampling a marked element under the stationary distribution. The worst case is when a single leaf is marked. In appendix C.3 we derive an expression for the stationary distribution. From eq. (C.13), we may bound

$$\epsilon \coloneqq \mathbb{P}_{\mathbf{J}\sim\pi}(\mathbf{J} \in M) \geq \frac{\tilde{\pi}_d}{(d+1)^d} = \frac{1}{2d(d+1)^d}. \tag{4.10}$$

Substituting the quantum costs as well as the computed values of δ and ϵ into the equation provided by theorem 1.3.4 gives a runtime of

$$\tilde{\mathcal{O}}\Big(k^2 d^2 + \sqrt{2d(d+1)^d}\left(\sqrt{2^9 d^2} k^2 d + k^2 d\right)\Big) = \tilde{\mathcal{O}}\big(k^2 \cdot d^{(d+5)/2}\big). \tag{4.11}$$

We therefore obtain our second result, proposition 4.0.2.

Proposition 4.0.2 (CSP, Hamming distance)**.** *There exists a quantum algorithm for solving the CSP under the Hamming distance, with runtime* $\tilde{\mathcal{O}}\big(kn + k^2 \cdot d^{(d+5)/2}\big)$.

Compared with the classical $\mathcal{O}\big(kn + k \cdot d^{d+1}\big)$ runtime, this represents a quadratic improvement in the dominant factor $d^d \to d^{d/2}$.

4.2.3 A Quantum dynamic programming algorithm for the CSP under Levenshtein distance

Nicolas and Rivals [98] give a dynamic programming algorithm for the CSP under general weighted edit distances using a $(n+1)^{2k}$ size Boolean array. This is precisely the setting considered by Glos et al. [51]. The only change required is to replace the $\mathcal{O}(1)$-cost membership queries of Glos et al. with $\mathcal{O}\big(\sigma \cdot 2^k\big)$-cost logical ORs required by Nicolas and Rivals. By applying theorem 1.3.5, we obtain a quantum algorithm with runtime $\tilde{\mathcal{O}}\big(\sigma \cdot 2^k \cdot T(n)^{2k}\big)$, and thus our third result, proposition 4.0.3.

Proposition 4.0.3 (CSP, Levenshtein distance)**.** *There exists a quantum algorithm for solving the CSP under the Levenshtein distance, with runtime* $\tilde{\mathcal{O}}\big(\sigma \cdot 2^k \cdot T(n)^{2k}\big)$*, where* $T(n)$ *satisfies* $n/e \leq T(n) < n$ *for every* n.

4.3 A quantum algorithm for the closest substring problem

For the CSSP under Hamming distance, we quantise Marx's XP algorithm [113] to achieve a quadratic speedup.

The algorithm is based on searching through particular sets of strings called *generators*. A set $G = \{g_1, \ldots, g_l\}$ of length-L strings is a generator of the length-L string s if, whenever all g_i have the same character at some position p, then s also has that character at position p. The *conflict* of G is the number of positions where the g_i do *not* all agree. Marx proves that if an instance of the CSSP is solvable, then there exists a solution with a generator that is not too large and has low conflict. We restate the result in theorem 4.3.1.

Theorem 4.3.1 ([113])**.** *If an instance of CSSP is solvable, then there exists a solution that has a generator $G = \{g_1, \ldots, g_l\}$ satisfying:*

- *Each g_i is a length-L substring of some input s_j.*
- $|G| \leq \log(d) + 2$.
- *The conflict of G is at most $d(\log d + 2)$.*

The corresponding classical CSSP algorithm is as follows. A set T containing all length-L substrings of inputs is produced. All candidate generators $G \subseteq T$ with $|G| = \log(d) + 2$ are enumerated, giving $\binom{|T|}{\log(d)+2} = \mathcal{O}\big((kn)^{\log(d)+2}\big)$ possibilities. Each generator with conflict at most $d(\log(d) + 2)$ generates at most $\sigma^{d(\log(d)+2)}$ candidate strings. Each is checked in time $\mathcal{O}(k \cdot (n - L) \cdot L) = \mathcal{O}(kn^2)$, yielding total cost of $\mathcal{O}\big(\sigma^{d(\log(d)+2)}(kn)^{\log(d)}k^3n^4\big)$.

Crucially, the algorithm is simply an exhaustive search over subsets $G \subseteq T$ and then over strings s generated by G. Applying a nested Grover search subroutine to both of these searches leads to a quantum algorithm with a runtime $\tilde{\mathcal{O}}\Big(\sqrt{(kn)^{\log(d)+2}} \cdot\sqrt{\sigma^{d(\log(d)+2)}} \cdot kn^2\Big)$. We therefore obtain proposition 4.0.4.

Proposition 4.0.4 (CSSP, Hamming distance)**.** *There exists a quantum algorithm for solving the CSSP under the Hamming distance, with runtime* $\tilde{\mathcal{O}}\Big(\sigma^{d(\log(d)+2)/2} \cdot(kn)^{\log(d)/2} \cdot k^2n^3\Big)$.

Chapter 5

Pangenome-guided sequence assembly via binary optimisation

As discussed in chapter 2, a single linear reference genome cannot capture the full range of intra-species variation, whether it is local Single Nucleotide Variants (SNVs) or large-scale Copy Number Variations (CNVs) [116]. However, assembling DNA reads from scratch with *de novo* assemblers is expensive and often yields incomplete results. With the rapid decline in sequencing costs, whole-genome shotgun short-read data now exist for millions of human genomes. Reconstructing these using a pangenome rather than a standard reference, such as GRCh38 [117], offers a more accurate and efficient approach.

Existing pangenomics tools consider only a limited range of tasks. *DRAGEN* [118] performs variant detection by mapping reads against a pangenome before mapping back to a comparison against a single reference genome. *Pasa* [119] uses pangenomes as a source of global information to resolve assemblies with a large number of contigs into a single sequence. No dedicated pipeline currently reconstructs genomes from short reads and pangenomes.

This chapter introduces a methodology for *pangenome-guided sequence assembly* that maps reads onto a pangenome and reformulates the assembly as a graph-traversal optimisation. Most genomic regions are easily resolved; we focus on the highly complex, repetitive regions of the pangenome graph. The pipeline, summarised in fig. 5.1, consists of five stages:

1. **Problem creation:** Generate a synthetic pangenome and a new individual genome. Simulate short-read sequencing (typically 30× coverage).

2. **Read mapping:** Align short reads to the pangenome using *GraphAligner*, *kmer2node* or *minigraph*, and annotate nodes with kmer counts.

3. **Copy number estimation:** Estimate node copy numbers from the annotated graph using *pathfinder*.

4. **Path finding:** Identify a path through the pangenome graph that best fits the observed copy numbers using either the *pathfinder* tool or a novel binary optimisation approach.

5. **Solution processing:** Extract and optionally re-align the resulting sequence to improve quality before evaluating against the ground truth.

The core computational task is the path finding in step 4. This problem is similar to those tackled by certain *de novo* genome assemblers like *Flye* [120] and *Oatk* [121], which aim to resolve repeat-induced non-linear structures in assembly graphs. Their approach typically also involves estimating the copy numbers of repeat sequences from sequence coverage, then identifying a graph traversal that satisfies these constraints.

In the context of pangenome graphs, however, additional complexities arise. First, due to genomic divergence, a pangenome graph does not fully represent the target genome, necessitating imputation of missing or divergent regions. Second, mapping reads to a pangenome graph is inherently noisier due to its repetitive and multi-genome nature, resulting in greater uncertainty in sequence coverage and, consequently, in copy number estimates. To accurately resolve the graph paths through repetitive regions, pangenome-based methods must be more robust to such noise.

We formalise path finding as an NP-hard optimisation task that generalises the Hamiltonian path problem. We cast it as a Quadratic Unconstrained Binary Optimisation (QUBO), a framework commonly used in the optimisation literature for its ability to balance expressivity and tractability. Moreover, QUBO is, in some sense, "native" to quantum optimisation, particularly the Quantum Adiabatic Optimisation Algorithm (QAOA). We also test Higher-order Unconstrained Binary Optimisation (HUBO) approaches and their potential to alleviate the quantum resource usage. We compare both classical and quantum approaches to the optimisation and discuss the potential utility of quantum methods.

In section 5.1, we formalise the optimisation task that underpins our approach and discuss how it can be cast to various forms of unconstrained binary optimisation. Sections 5.2 to 5.4 discuss classical, quantum annealing, and QAOA approaches to solving these binary optimisation tasks, respectively. Finally, section 5.5 concludes with a discussion and the myriad future research directions.

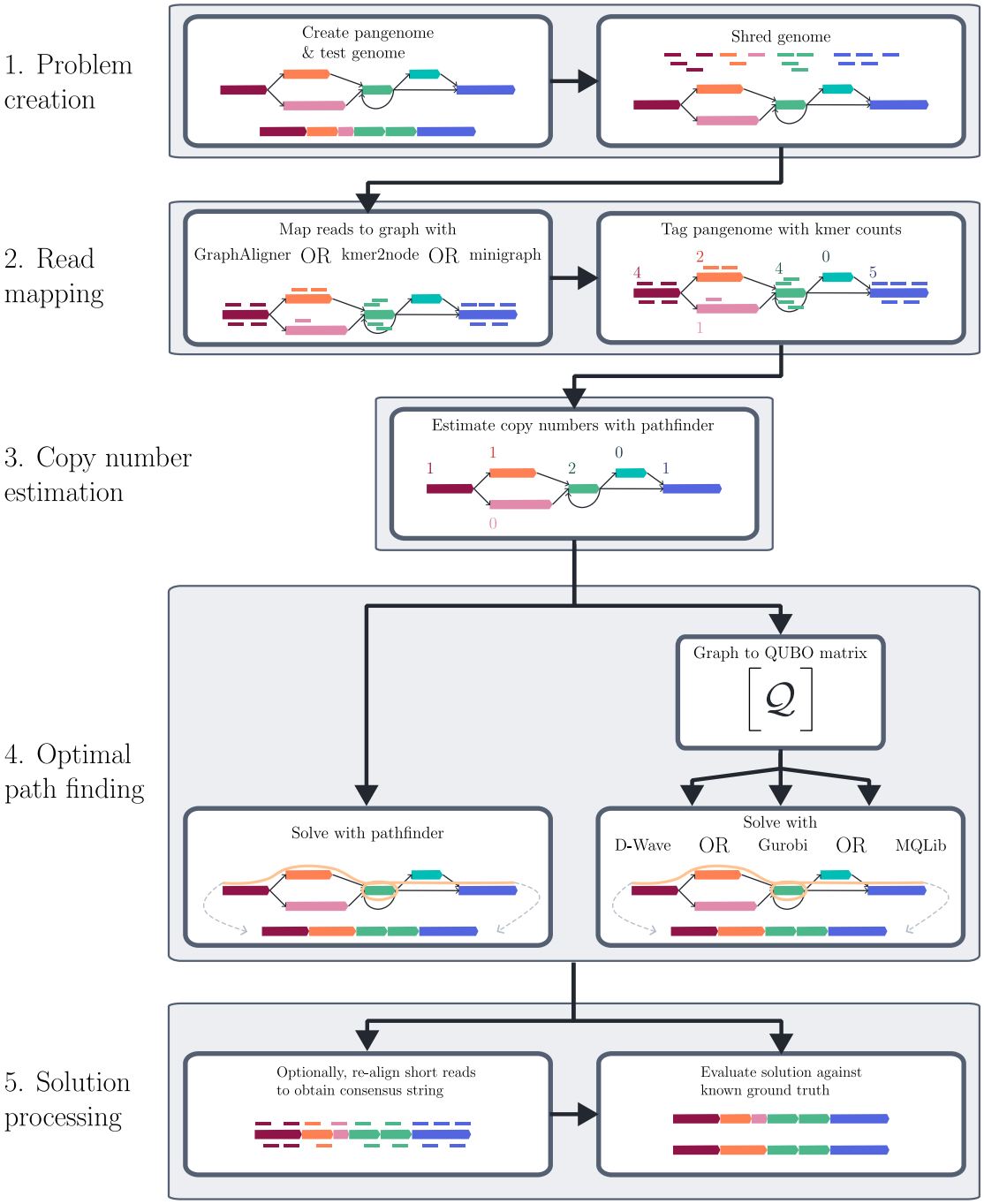


Fig. 5.1 A sketch of the pangenome-guided sequence assembly procedure. (1) *Problem creation* consists of pangenome and short-read data synthesis. (2) *Read mapping* involves aligning the short reads to the pangenome and tagging the nodes with the observed kmer counts. (3) *Copy number estimation* is performed using *pathfinder*. (4) *Optimal path finding* either uses *pathfinder* or constructs a QUBO matrix and inputs that into a chosen solver. (5) *Solution processing* begins with an optional realignment step to improve solution quality, followed by evaluation with a variety of metrics.

JB wrote the classical pre- and post-processing code; I developed all code involving mapping to or solving optimisation problems.

5.1 Pangenome-guided sequence assembly as a binary optimisation problem

The computational core of pangenome-guided assembly is finding graph paths consistent with observed copy-number data. We formalise three optimisation problems of increasing biological realism in section 5.1.1 and map them to QUBO forms in section 5.1.2 and HUBO forms in section 5.1.3. The QUBO format is natively accepted as input by classical optimisation suites and quantum annealing optimisers, and it is easy to map to the Ising Hamiltonians used to define QAOA problem instances. HUBO forms also map easily onto Ising Hamiltonians and require almost quadratically fewer variables; however, they are not natively solved by classical optimisers.

5.1.1 Pangenome-guided assembly as an optimisation

Our problem data is a pangenome graph with non-negative node weights. We formally define the input as a triplet $G = (V, E, w)$, where V is the vertex set, $E \subseteq V \times V$ the edges, and $w : V \to \mathbb{R}_+$ gives the copy numbers.

We seek a walk W on G that best explains the copy number data. We define the following optimisation problem, which we refer to as *Tangle Resolution* due to the often complex, knot-like structures present in pangenomes. Note the implicit requirement in the problem that each step in W traverses an edge in G.

Problem 5.1.1 (Tangle Resolution)**.** *Given a vertex-weighted graph $G = (V, E, w)$, find a walk W on G that minimises*

$$C_G(W) = \sum_{v \in V} \Big(\#W(v) - w(v) \Big)^2, \tag{5.1}$$

where $\#W(v)$ is the number of times W visits v.

This graph traversal problem is similar to the Travelling Salesman Problem (TSP), but with a node-visit matching criterion rather than distance minimisation.

While this model serves as a good starting point, it does not encompass all the biological features. Most notably, it overlooks the double-stranded nature of genomic data, in which both a sequence and its reverse complement are present in the DNA.

To better reflect the biology, we must consider a *directed* graph. The vertex set is now formed of two halves, V_+ and V_-. The sequence data corresponding to $v_- \in V_-$ is the reverse complement of the sequence data corresponding to $v_+ \in V_+$. Edges of the graph are directed and come in reciprocal pairs, e.g. an edge $u_+ \to v_+$ implies the existence of an edge $v_- \to u_-$, and vice versa. We call such a graph $G = (V_+ \cup V_-, E, w)$ *oriented*.

The corresponding optimisation task, *Oriented Tangle Resolution*, is similar, but we now sum the visits over both the positive and negative orientations:

Problem 5.1.2 (Oriented Tangle Resolution)**.** *Given a vertex-weighted, oriented graph $G = (V_+ \cup V_-, E, w)$, find a walk W on G that minimises*

$$C_G(W) = \sum_{v \in V} \Big(\#W(v_+) + \#W(v_-) - w(v) \Big)^2, \tag{5.2}$$

where $\#W(v_+)$ and $\#W(v_-)$ are the number of times W visits v_+ and v_-, respectively.

Finally, when the pangenomes under consideration are human, we need to account for individuals having two copies of each chromosome. DNA of this type is known as *diploid*. We therefore need to find a *pair* of walks W_1 and W_2 through an oriented graph that combine to explain the data. We thus have *Diploid Tangle Resolution*:

Problem 5.1.3 (Diploid Tangle Resolution)**.** *Given a vertex-weighted, oriented graph $G = (V_+ \cup V_-, E, w)$, find a pair of walks W_1, W_2 on G that minimise*

$$C_G(W_1, W_2) = \sum_{v \in V} \left(\sum_{i=1}^{2} \Big(\#W_i(v_+) + \#W_i(v_-) \Big) - w(v) \right)^2, \tag{5.3}$$

where $\#W_i(v+)$ and $\#W_i(v_-)$ are the number of times walk W_i visits v_+ and v_-, respectively.

This formulation naturally extends to polyploid data by increasing the number of walks; we will not consider that setting here.

5.1.2 Mapping the optimisation task to quadratic unconstrained binary optimisation

The QUBO formulation defines a class of optimisation problems characterised by the minimisation of a quadratic polynomial over binary variables x_i. Since $x_i^2 = x_i$ for $x_i \in \{0, 1\}$, any linear term can be absorbed into the quadratic form; constant offsets

merely shift the energy landscape without altering the optimal assignment. Therefore, the objective can be formally defined as

$$\min_{x \in \{0,1\}^n} x^T M x, \tag{5.4}$$

where $M \in \mathbb{R}^{n \times n}$ is a symmetric, real-valued square matrix.

QUBO problems exhibit a high degree of expressivity, enabling them to represent classic combinatorial optimisation problems such as MAX-CUT and the TSP, while also remaining tractable to modern solvers at moderate sizes. Constraints are incorporated into the objective function through penalty terms. Solution techniques include generic branch-and-bound methods, as employed by commercial solvers such as *Gurobi*, as well as domain-specific heuristics, such as those implemented by the open-source library *MQLib*.

QUBO is equivalent, under a linear change of variables, to the Ising model in statistical physics, a formulation extensively used in quantum annealing devices such as those developed by *D-Wave* Systems. As such, QUBO is widely employed in quantum computing contexts, where its unconstrained nature and binary domain align with the hardware capabilities of current quantum and neuromorphic processors.

Notably, however, the QUBO formulations of many classic graph-traversal problems, such as the TSP, require quadratically many variables, traditionally using binary variables x with two indices t and v, where $x_{t,v} = 1$ if the solution path visits vertex v at time t.

We propose a QUBO formulation for problems 5.1.1, 5.1.2 and 5.1.3. For concreteness, we describe the simplest, problem 5.1.1.

Similar to QUBO forms of the TSP, our formulation uses quadratically many variables. In particular, our formulation uses $(N+1)T$ variables, where $N = |V|$ is the number of vertices in the graph and T is the maximum length of a walk we consider. In practice, we choose $T = \alpha \sum_{v \in V} w(v)$ for some constant $\alpha > 1$, which we usually take equal to 1.2. This allows walks to visit some vertices more than the data suggests if doing so significantly improves the quality of another part of the solution. The extra vertex introduced is a virtual "end" node with no sequence data attached. A walk can spend several consecutive time steps in the end node without incurring any cost, allowing walks to effectively finish early. However, any step that leaves the end node is heavily penalised. A sketch of this process is given in fig. 5.2.

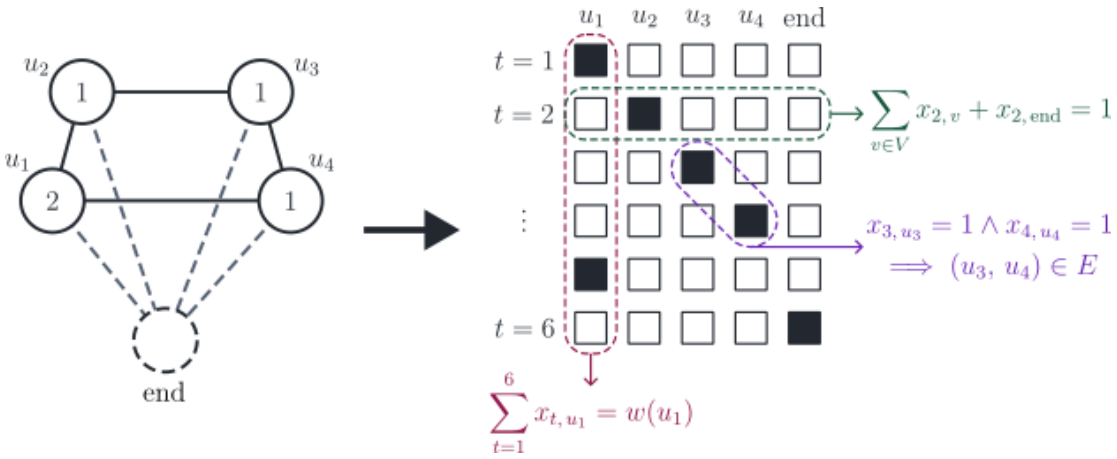


Fig. 5.2 A sketch of the procedure for converting a graph-traversal problem to a QUBO problem. (Left) A graph annotated with weights, corresponding to a pangenome with reads aligned. Also shown is the virtual "end" node used in the QUBO formulation. (Right) A diagrammatic representation of the QUBO, with binary variables $x_{t,v}$ shown as squares that are filled if $x_{t,v} = 1$. Some of the constraints that we impose are highlighted: in green, the constraint that a single variable is "on" at time 2; in purple, that the path traverses a graph step going from time 3 to 4; and in red, that the path visits vertex u_1 twice, according to its weight.

The total cost function consists of three parts: the first is a Lagrange multiplier term enforcing one-vertex-per-time; the second is a multiplier term enforcing valid graph-edge traversal; the final term encodes the cost function $C_G(W)$.

In particular, we write the QUBO cost function as $Q_G(\{x_{t,v}\}) = Q_G^1(\{x_{t,v}\}) + Q_G^2(\{x_{t,v}\}) + Q_G^3(\{x_{t,v}\})$ where

$$Q_G^1(\{x_{t,v}\}) \coloneqq \Lambda_1 \sum_{t=1}^{T} \left(\sum_{v \in V} x_{t,v} + x_{t,\text{end}} - 1 \right)^2, \tag{5.5}$$

$$Q_G^2(\{x_{t,v}\}) \coloneqq \Lambda_2 \sum_{t=1}^{T-1} \left(1 - \sum_{v,v' \in V} x_{t,v} x_{t+1,v'} \mathbb{1}\big((v,v') \in E\big) - \sum_{v \in V} x_{t,v} x_{t+1,\text{end}} \right), \tag{5.6}$$

$$Q_G^3(\{x_{t,v}\}) \coloneqq \sum_{v \in V} \left(\sum_{t=1}^{T} x_{t,v} - w(v) \right)^2. \tag{5.7}$$

The choice of Lagrange multipliers Λ_1, and Λ_2 requires empirical calibration. If they are sufficiently large, then we can guarantee that the assignment that minimises $Q_G(\{x_{t,v}\})$ encodes a walk that minimises $C_G(W)$. However, if they are too large, then the energy landscape will have steep valleys around assignments that satisfy the constraints, and

changes in the cost function will be nearly indistinguishable to the optimiser. We find that intermediate-size values of $\Lambda_1 = 10$ and $\Lambda_2 = 5$ work well in practice.

To extend this method to Oriented Tangle Resolution, we introduce a pair of variables for each vertex and each time, resulting in $(2N + 1)T$ variables. For Diploid Tangle Resolution, we further introduce a new set of variables for the second path, resulting in $2(2N + 1)T$ variables.

5.1.3 Mapping the optimisation task to higher-order unconstrained binary optimisation

The quadratic scaling in the QUBO formulation's number of variables limits the practical problem size on current hardware. A HUBO formulation reduces variable count by encoding each node index in binary rather than enumerating all node–time pairs. HUBO formulations are more expressive than QUBO, with terms containing products of arbitrarily many binary variables; the downside is that (classically) solving them is generally more difficult and less well-explored in the literature. Notably, however, HUBO may not have such a large solution overhead when mapped to QAOA algorithms on quantum computers. In that setting, each layer of the cost Hamiltonian now contains multi-qubit Z rotations, rather than only single- and two-qubit rotations. These gates need to be compiled down to 2-qubit gates, but constructions that are linear in the number of qubits involved exist [122]. Careful circuit compilation can further reduce the depth overhead; see section 5.4.2 and appendix D for details.

For a HUBO formulation of *Tangle Resolution*, we again add a virtual "end" node to the set of vertices, leading to $N + 1$ sites for the walk at each time step. Let $n = \lceil \log_2(N+1) \rceil$ and define binary variables $\{x_{t,k}\}$ for $t = 1, \ldots, T$ and $k = 0, \ldots, n-1$. For each time t, the variables $\{x_{t,0}, \ldots, x_{t,n-1}\}$ encode some integer X_t in binary. When all $X_t \leq N$, $(X_1, \ldots, X_T)$ can be interpreted as a walk on the graph; otherwise, the solution is infeasible.

First, we note the following identity, which holds for binary variables x and y:

$$1 - x - y + 2xy = \begin{cases} 1 & x = y, \\ 0 & x \neq y. \end{cases} \tag{5.8}$$

By taking the product of n terms of the form of the LHS of eq. (5.8), we can construct an indicator function for the encoded integers X_t. For an integer $i \in \{0, \ldots, N\}$, let

$(b_0, b_1, \ldots, b_{n-1})$ be its binary encoding. Then we have

$$\mathbb{1}(X_t = i) \coloneqq \prod_{k=0}^{n-1} \Big(1 - b_k - x_{t,k} + 2b_k \cdot x_{t,k}\Big) = \begin{cases} 1 & X_t = i, \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

The cost function now consists of two terms. The first adds a penalty whenever subsequent X_t, X_{t+1} do not represent a valid edge of the graph. The second is the objective term that encodes the node-frequency matching. The HUBO cost function corresponding to problem 5.1.1 is then $H_G(\{x_{t,k}\}) = H_G^1(\{x_{t,k}\}) + H_G^2(\{x_{t,k}\})$ where:

$$H_G^1(\{x_{t,k}\}) \coloneqq \Lambda_1 \sum_{t=1}^{T-1} \left[1 - \sum_{i=0}^{N} \left(\mathbb{1}(X_t = i) \cdot \Big(\sum_{\substack{j: \\ (i,j)\in E}} \mathbb{1}(X_{t+1} = j) + \mathbb{1}(X_{t+1} = N)\Big)\right)\right], \tag{5.10}$$

$$H_G^2(\{x_{t,k}\}) \coloneqq \sum_{i=0}^{N-1} \left(\sum_{t=1}^{T} \mathbb{1}(X_t = i) - w(i)\right)^2. \tag{5.11}$$

5.1.4 Mapping binary optimisation to adiabatic optimisation

Quantum adiabatic optimisation algorithms take as input a Hamiltonian H chosen so that the ground state of H encodes an optimal assignment to some cost function.

In the case of binary optimisation, the choice of H is simple. Let $C(\{x_i\})$ be some cost function on binary variables x_i. Let H_C be the Hamiltonian obtained by replacing, for each i, every occurrence of x_i in $C(\{x_i\})$ with $\frac{1}{2}(I - Z_i)$. Since $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$, we have $\frac{1}{2}(I - Z)|b\rangle = b|b\rangle$ for $b \in \{0, 1\}$. We can then easily check that

$$H_C|x\rangle = C(x)|x\rangle \quad \forall\ x \in \{0,\ 1\}^n. \tag{5.12}$$

Therefore, the *energy* of any computational basis state $|x\rangle$, $\langle x| H_C |x\rangle$, exactly corresponds to the cost of the assignment x.

A further benefit of converting binary cost functions to Hamiltonians is that the circuits to sample from during the quantum optimisation are relatively simple. Recall that a QAOA circuit consists of p repetitions of a cost layer and a mixer layer, where the cost layer is given by $\exp(-i\gamma H_C)$. Since all Z_i operators commute with each other, this exponential decomposes to a simple product of terms; namely, a product of Z interactions $\exp(-i\gamma Z \otimes \ldots \otimes Z)$. The number of qubits in any interaction is limited

by the degree of the cost function. In particular, only one- and two-qubit gates are required for a QUBO cost function.

5.1.5 Evaluation methods

Both the QUBO cost function Q_G and the HUBO cost function H_G are chosen so that a walk on the graph that perfectly explains the weight data has zero cost. Any discrepancies in the weights incur a slight cost increase; any illegal graph steps, or, for QUBO, any time when not exactly one variable is "on," incur a significant penalty. These costs, however, may not perfectly reflect the quality of the solution to the underlying sequence-assembly task. To evaluate biological accuracy, graph-level solutions are mapped back to sequences and compared with known ground truths from synthetic pangenomes.

Candidate assemblies are aligned against the known truth using standard bioinformatics tools. This produces one or more alignments, each representing co-linear matches between the candidate and true genomes, being broadly equivalent to the contigs produced by a *de novo* assembler. Each alignment may, in turn, have more minor variations that were not deemed significant enough to break the alignment into two. This gives us a unified way to compare pangenome path reconstruction against more traditional *de novo* sequence assembly tools. We report the number of contigs and align them against the true genome to identify the proportion of the true genome that we have sampled, and vice versa. This two-way approach permits us to spot over- and under-calling of copy-number variations.

These provide basic evidence on the contiguity of the assembled data, but not its accuracy. To this end, we also report the number of breaks when aligning these contigs back to the true genome (representing incorrect path forks or large indels that cause a supplementary alignment). These are analogous to false joins from a *de novo* assembler. Generally, more contigs are preferred to incorrectly joined contigs, but the relative weighting of the two is a subjective decision. For finer-grained evaluation, we also report small insertions and deletions (≥ 10 base pairs), regions of substantial base variation (multi-nucleotide polymorphism with 30% variance within a 100 base pair window), and an overall percentage identity.

A further optional refinement step is to realign the short reads back to the generated candidate genome sequence. This corrects for novel mutations in this sample or simplifications made during the pangenome creation step.

5.2 Classical approaches

The primary benchmark for existing methods is *pathfinder* [121]. *Pathfinder* is a submodule from the *Oatk* package, a tool for resolving plant organelles using high-accuracy long reads. While this application is far from our current setting, the methods are related. In particular, *pathfinder* also assigns weights to a graph and then searches for a walk that explains the data. However, it proceeds via an exhaustive search rather than an optimisation. To the best of our knowledge, this is the software package that gives the fairest comparison with our results.

5.2.1 Pangenome-guided assembly versus *de novo* assemblers

We first determine whether pangenome-guided assembly is an appropriate strategy when compared to state-of-the-art *de novo* assemblers. A comparison of pangenome-guided assembly of synthetic data using *pathfinder* with *de novo* sequence assembly is shown in table 5.1. The primary difference lies in the number of contigs returned. Table 5.1 shows that a pangenome-guided approach resolves short reads into only 1–5 long contigs, reconstructing most of the genome with only minor errors; conversely, *de novo* methods leave 15–30 contigs. This simplification dramatically reduces downstream ordering complexity, at a small cost in false joins (alignment breaks). Optimising the path sequence by realigning the short read data against it and producing a new consensus further improves accuracy across a range of measures.

5.2.2 Comparing optimisation approaches to *pathfinder*

We benchmark the performance of our formulation using two classical optimisation suites. *Gurobi* [123] is an industry-standard package that employs branch-and-bound methods to find provably optimal solutions to a range of optimisation problems, including mixed-integer quadratic programming tasks which admit QUBO as a special case. *MQLib* [124] is a package that provides fast implementations of several heuristics for QUBO. We use the multistart tabu search strategy introduced by Palubeckis [125], as it performs most consistently across our instances.

We test the performance of each solver – *pathfinder*, *Gurobi* and *MQLib* – on graphs obtained with each annotation strategy – *GraphAligner*, *minigraph* and the in-house *kmer2node* tool. We use the Oriented Tangle Resolution problem, defined in section 5.1, as it captures the biological nature of the task while being simpler than the Diploid version, making it easier to benchmark across all solution strategies.

Table 5.1 Performance of pangenome-guided mapping of short-read data using *minigraph*, *GraphAligner*, *vg giraffe* and *kmer2node* weight assignments with the *pathfinder* solver, compared against pure *de novo* sequence assembly from *syncasm* and *miniasm*. Averaged statistics from 51 seeds with five alignments each. *Pathfinder* results are presented twice, evaluating the original concatenated path and an optimised path formed by realigning all the short-read data back to the original path to produce a refined consensus sequence. Columns represent the proportion of the true genome covered by the assembly result (%Cov.), and the percentage of this result covered by the true genome (%Used), the number of contigs (Contigs), the number of breaks in alignment of assembly to the actual sequence *i.e.* false joins (Breaks), the number of large indels (Indels) and regions of significant runs of differences (Diffs), and total percent identity between assembly and true genome (% Ident) – See section 5.1.5 for further details.

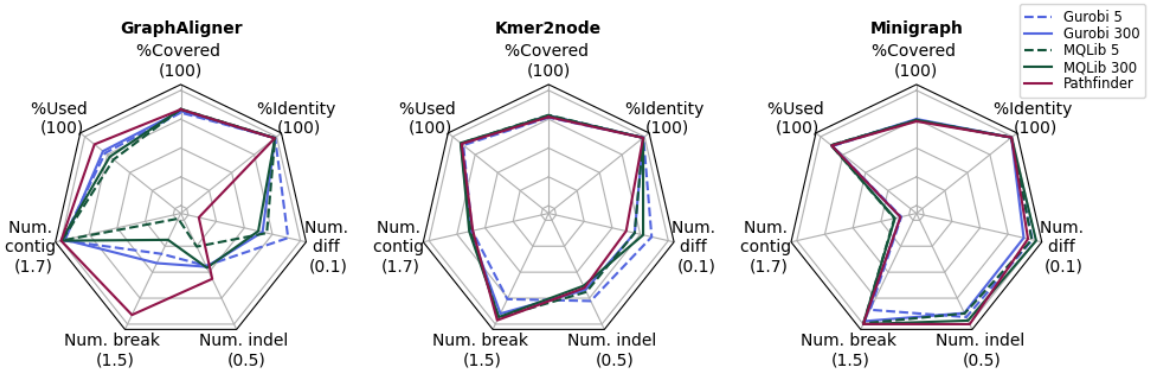
Annotator/ Assembler	%Cov.	%Used	Contigs	Breaks	Indels	Diffs	%Ident.
Kmer2node	86.7	93.2	2.8	1.9	1.3	0.2	96.9
Minigraph	84.0	90.9	4.7	2.4	1.0	**0.0**	97.2
GraphAligner	92.4	91.8	1.7	2.0	1.6	0.2	96.9
Giraffe	91.8	91.5	1.6	2.3	1.5	0.1	96.9
Kmer2node-opt	86.8	**94.1**	2.6	1.5	0.6	0.3	98.7
Minigraph-opt	84.4	92.7	4.3	1.8	0.3	0.1	99.2
GraphAligner-opt	**92.7**	92.5	1.6	1.7	0.8	0.3	98.6
Giraffe-opt	92.1	92.0	**1.5**	1.9	0.8	0.3	98.6
Syncasm	91.6	86.2	31.3	**0.0**	**0.1**	**0.0**	**100.0**
Miniasm	81.5	91.3	14.8	0.1	**0.1**	**0.0**	99.6

Table 5.2 Performance of pangenome guided mapping of short-read data using *minigraph*, *GraphAligner* and *kmer2node* weight assignments with the *MQLib*, *Gurobi* and *pathfinder* solvers. Averaged statistics from 20 seeds with five alignments each. *MQLib* and *Gurobi* were given 300 seconds per run, with three runs per alignment.

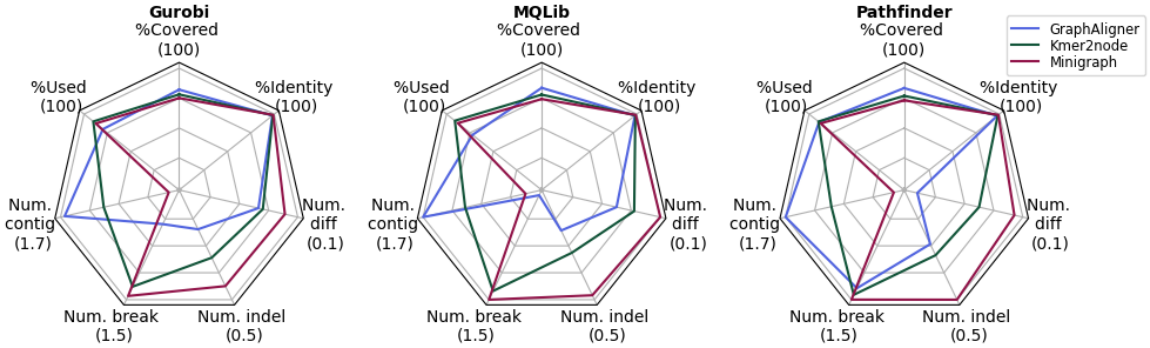
Annotator	Solver	%Cov.	%Used	Contigs	Breaks	Indels	Diffs	%Ident.
Graph Aligner	Gurobi	85.69	83.85	1.92	5.31	1.09	0.18	98.45
	MQLib	**86.90**	77.75	**1.73**	6.76	1.08	0.19	98.42
	Pathfinder	86.74	91.42	**1.73**	2.09	0.97	0.33	98.21
Kmer2node	Gurobi	82.50	92.07	3.73	2.17	0.86	0.17	98.82
	MQLib	82.36	**92.88**	3.68	1.95	0.90	0.15	98.75
	Pathfinder	81.55	92.01	3.85	1.76	0.88	0.19	98.70
Minigraph	Gurobi	80.09	89.92	6.74	1.70	0.63	0.12	99.09
	MQLib	79.40	90.31	6.47	**1.52**	0.56	**0.09**	99.14
	Pathfinder	78.51	90.06	6.74	**1.52**	**0.52**	0.11	**99.17**

We test each combination on 20 synthetic pangenomes with an average of 77.8 nodes, with five sequences aligned to each graph. *Gurobi* and *MQLib* are given time limits of 5 and 300 seconds to test whether higher quality solutions are obtained at long run times. These classical optimisers are allowed three runs per sequence and per time limit to account for variations in solution quality due to their heuristic nature. The averaged results are given in table 5.2. Radar charts showing the average performance across the metrics discussed in section 5.1.5 are given in fig. 5.3. Figure 5.3a shows that, for any choice of annotation strategy, each of the solvers is broadly competitive with the others. With *kmer2node* or *minigraph* annotations, the solvers yield comparable average-case results. The starkest difference comes when *GraphAligner* is used. In that case, *pathfinder* reports more large differences than either *MQLib* or *Gurobi*, but fewer breaks and slightly fewer large indels on average. The large number of breaks in the QUBO solver results may be due to not taking into account edge weights, which can help to resolve inversions: see section 5.5 for a brief discussion.

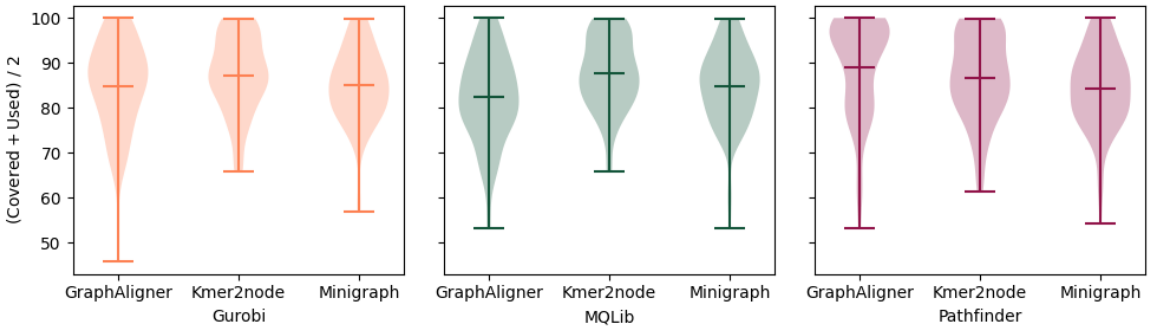
As well as average-case performance, it is illuminating to see how the combinations perform on specific instances, particularly hard ones. Violin plots showing the average of the "Covered" and "Used" statistics across all instances are given in fig. 5.3c. When *GraphAligner* is used to annotate graphs, it is clear that *pathfinder* outputs high-quality solutions on a majority of instances. However, a very low worst-case outcome drags down its average performance. For the other annotators, the QUBO solvers are much



(a) Radar charts comparing the performance of each of the annotation strategies.



(b) Radar charts comparing the performance of each of the classical solvers.



(c) A violin plot comparing the performance of each combination of annotation strategy and classical solver.

Fig. 5.3 (a), (b) Radar charts comparing the performance of each combination of annotation strategy: *GraphAligner*, *kmer2node* and *minigraph*; and classical solver: *Gurobi*, *MQLib* and *pathfinder*. The seven axes plotted are the evaluation criteria discussed in section 5.1.5. For each, the number in brackets corresponds to the outermost point on that axis. For the lower half of each plot, points further out along the axis correspond to fewer contigs, breaks, and so on. (c) Violin plots showing the per-instance performance of *Gurobi*, *MQLib* and *pathfinder*.

Table 5.3 Performance of pangenome guided mapping of short-read data using *minigraph*, *GraphAligner* and *kmer2node* weight assignments with the *D-Wave* and *pathfinder* solvers. Averaged statistics from 10 seeds with three alignments each. *D-Wave* was given 60 seconds per run, and two runs per alignment.

Annotator	Solver	%Cov.	%Used	Contigs	Breaks	Indels	Diffs	%Ident.
Graph Aligner	D-Wave	80.64	86.31	**1.10**	5.33	0.63	0.10	99.03
	pathfinder	**93.24**	92.29	**1.10**	1.77	0.83	0.10	99.22
kmer2node	D-Wave	85.02	92.45	1.97	2.87	0.57	0.20	98.87
	pathfinder	85.69	**94.44**	1.90	**1.27**	0.77	0.13	99.08
minigraph	D-Wave	86.29	94.29	3.50	1.63	**0.23**	**0.03**	**99.59**
	pathfinder	86.03	94.43	3.53	**1.27**	0.33	**0.03**	99.58

more competitive, with plots that are similarly shaped across both the *minigraph* and *kmer2node* settings.

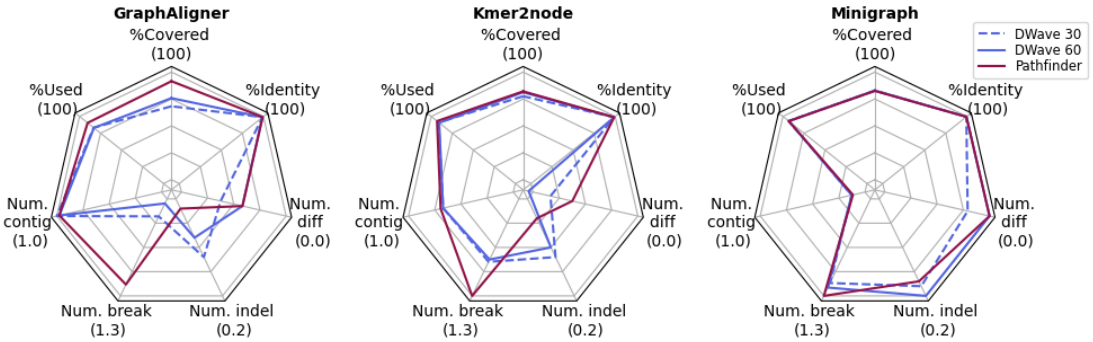
5.3 Quantum annealing

To explore the potential of QA, we perform a range of experiments using *D-Wave*'s *Leap* hybrid solver [126], which combines classical heuristics with quantum processing. The *Advantage2* system has 4,400 qubits arranged in a lattice structure, around 30 times larger than the largest commercially available gate-model machines.

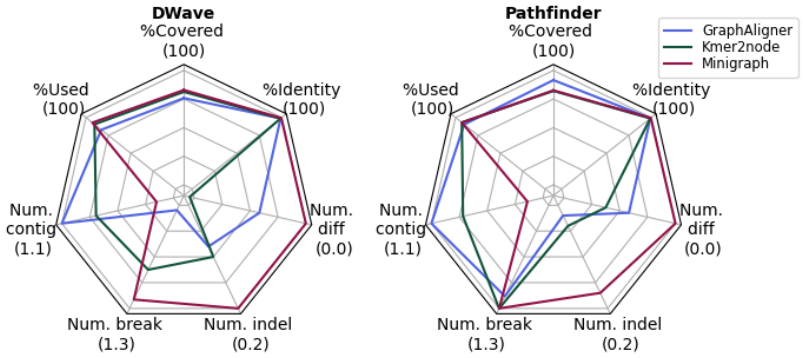
We test *D-Wave* on 10 pangenomes, each with three sequences to be aligned. Due to limited access to *D-Wave* systems, we tested on smaller instances with an average of 44.4 nodes per graph. The annealer was allowed two runs per sequence, with a time limit of either 30 or 60 seconds. Results are given in table 5.3 and visualised in fig. 5.4. For the instances tested, there is a clear benefit to using *minigraph* as an annotator; in that case, *D-Wave* performs almost identically to *pathfinder*.

5.4 Quantum adiabatic optimisation algorithms

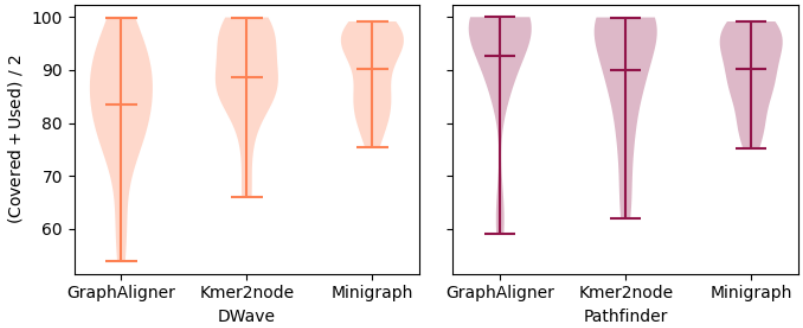
We perform a thorough classical simulation benchmark of the QAOA algorithm for Tangle Resolution. While the exponentially scaling cost limits the size of the problem instance we can explore, our results demonstrate the feasibility of our method for future runs on real hardware.



(a) Radar charts comparing the performance of each of the annotation strategies.



(b) Radar charts comparing the performance of *D-Wave* and *pathfinder*.



(c) A violin plot comparing the performance of *D-Wave* and *pathfinder* for each annotation strategy.

Fig. 5.4 (a), (b) Radar charts comparing the performance of each combination of annotation strategy: *GraphAligner*, *kmer2node* and *minigraph*; with quantum annealing solver *D-Wave* or classical solver *pathfinder*. The seven axes plotted are the evaluation criteria discussed in section 5.1.5. For each, the number in brackets corresponds to the outermost point on that axis. For the lower half of each plot, points further out along the axis correspond to fewer contigs, breaks, and so on. (c) Violin plots showing the per-instance performance of *D-Wave* and *pathfinder*.

Simulations used the GPU-accelerated *Qiskit Aer* package, and were run using up to 4 Nvidia H-100 GPUs and up to 512 GiB of RAM. All of our simulations contain shot noise to emulate hardware measurement statistics.

Convergence was greatly enhanced by using the Conditional Value at Risk (CVaR) objective, which focuses optimisation on the lowest-energy samples. Recent studies have justified CVaR both analytically [127] and empirically [127–129]. We also find that we can significantly increase simulation speed by improving the efficiency of classical sample processing. A naive sample-by-sample evaluation of the incurred cost takes time comparable to that of the circuit simulation. Using bitwise operations reduces the classical processing time to a negligible fraction of the quantum simulation time.

5.4.1 QAOA applied to the QUBO formulation

We use the advanced QAOA circuit compilation techniques described in [59] and [130] and implemented by IBM staff members on Github. These methods reduce QAOA circuit depth by finding an initial qubit layout on the hardware and a shallow network of SWAP gates that enable implementation of each QAOA interaction.

In general, a linear-depth SWAP network is needed to achieve all-to-all connectivity; in practice, a much smaller depth suffices for reasonably sparse Hamiltonians. To find this depth, Matsuo et al [130] propose a SAT-based approach. They first fix a *SWAP strategy* based on the hardware under consideration; for example, a simple brickwork pattern for linear nearest-neighbour connectivity.

For some fixed number of layers d of this strategy, one can efficiently compute all the pairs of qubits that are neighbours at some point during the SWAP network. Let X be the set of the device's physical qubits. For some physical qubit $x \in X$, let $N_d(x)$ be the set of all qubits which neighbour x at some point in the d layers of SWAPs. The aim is to map the qubits from the Hamiltonian onto physical qubits in such a way that every two-qubit interaction is mapped onto initial physical locations that will become neighbours at some point. To achieve this, a Boolean variable (i, x) is assigned to each possible pairing of virtual qubit i and physical qubit x. A standard Conjunctive Normal Form (CNF) formula is created that is satisfied if and only if each virtual qubit is mapped to a unique physical qubit. Then, for each two-qubit interaction, several CNF terms are added that are collectively satisfied if and only if that interaction can be implemented. In particular, an interaction between virtual qubits (i, j) leads to CNF terms

$$\Big\{\neg(i, x) \bigvee_{y \in N_d(x)} (j, y) : x \in X\Big\}. \tag{5.13}$$

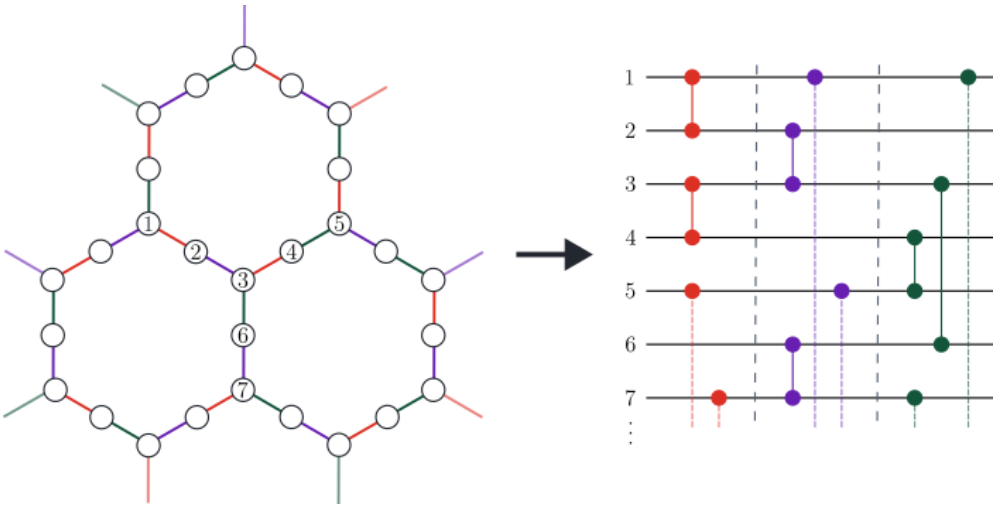


Fig. 5.5 The use of edge colouring to efficiently implement two-qubit gates on the IBM heavy-hex layout. (Left) Three cells of the heavy-hex layout and the corresponding 3-colouring. Some of the qubits are assigned a virtual qubit index. (Right) Part of a quantum circuit that implements all pairwise interactions of a heavy-hex layout in circuit depth 3. Vertical barriers separate operations applied at different circuit layers.

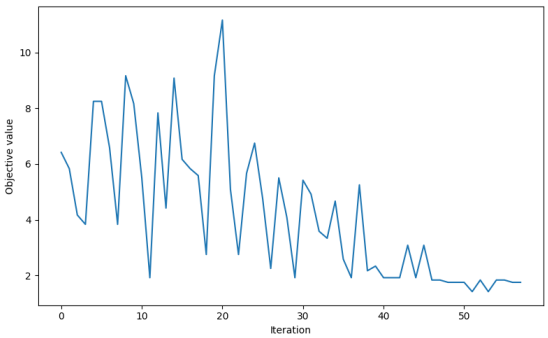
In words, each added term reads: "if i is mapped onto x, then j must be mapped onto some y that neighbours x."

The conjunction of all these terms and the 1-1 mapping CNF leads to a CNF that is satisfiable if and only if every interaction can be implemented in SWAP depth d. A binary search over d ensures that a layout which minimises the required number of SWAPs is found.

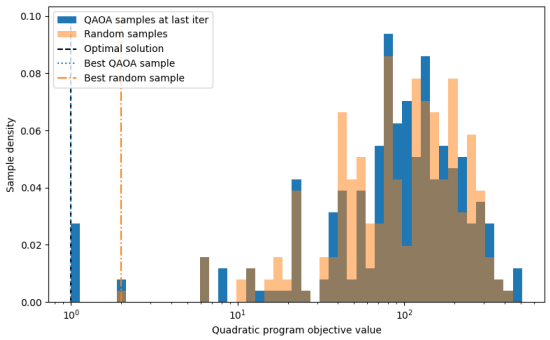
Between each SWAP layer, there is a set of interactions that can be implemented on the current qubit layout. These are ordered by finding a minimal edge colouring of the hardware coupling map and partitioning the interactions according to the colour of their corresponding edge. For example, the *heavy-hex* layout of current-generation IBM hardware permits a 3-colouring. Therefore, the interactions between any pair of SWAP layers can be implemented in depth at most 3. An illustrative example is provided in fig. 5.5.

Compiled circuits were simulated with up to 35 qubits. We also performed small experiments on real quantum devices; a comparison of simulated and actual performance for one such experiment is provided below.

Figure 5.6 shows the results of simulating the QAOA algorithm with $p = 4$ on a small Oriented Tangle Resolution problem. The circuit has 10 qubits, 496 two-qubit gates, and a two-qubit depth of 112 after compilation to the IBM-native gate set. 256 samples were taken at each iteration, and a maximum of 100 iterations was allowed.



(a) The objective value at each iteration of a QAOA simulation.



(b) Histogram of the samples obtained at the final iteration of the QAOA algorithm (blue), compared with random samples (orange). The optimal solution is shown in black dashes.

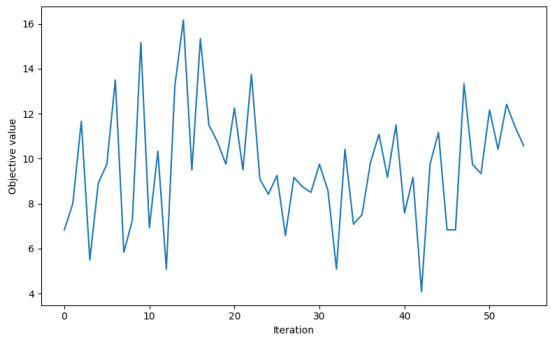
Fig. 5.6 Plots showing the performance of a 10-qubit QAOA simulation for Oriented Tangle Resolution.

The simulation took under 20 seconds to run on a single CPU and GPU, including circuit compilation time. Figure 5.6b demonstrates the successful convergence of the QAOA procedure. The simulated quantum computer sampled the optimum repeatedly and had several other samples at the same energy as the best random samples. The use of CVaR prevents the optimiser from seeing high-energy solutions, so there are many QAOA samples on the RHS of fig. 5.6b.

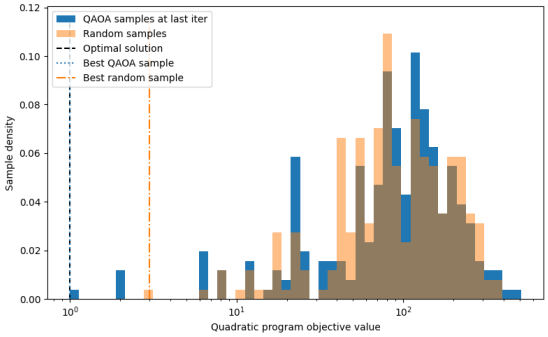
Figure 5.7 shows the results of running the same QAOA algorithm, with the same initial conditions, on an actual quantum device. The device used is the *IBM Strasbourg* computer, an Eagle R3 device with 127 qubits in a heavy-hex layout and an average 2-qubit error rate of approximately 3×10^{-2}. The whole computation took 213 seconds. Each of the 55 iterations of the algorithm used less than 1s of QPU time; the remaining time is comprised of classical computation, communication with the device and preparing the device for the next iteration.

Device noise degraded the optimiser's convergence. Nonetheless, Figure 5.7b still shows a non-trivial shift of probability mass towards lower-energy solutions. Moreover, the optimal solution was again sampled, and several other low-energy solutions were found. Future work will integrate Pauli twirling and measurement-error mitigation to improve stability.

To better understand the performance of hardware devices, we then considered a sampling-only task. In this variant, the QAOA circuit parameters were optimised classically by sweeping over the parameter space using local searches at each iteration.



(a) The objective value at each iteration of a QAOA algorithm on a real device.



(b) Histogram of the samples obtained at the final iteration of the QAOA algorithm (blue), compared with random samples (orange). The optimal solution is shown in black dashes.

Fig. 5.7 Plots showing the performance of a 10-qubit QAOA hardware experiment for Oriented Tangle Resolution.

We consider the same Oriented Tangle Resolution problem on 10 qubits, now with $p = 3$. Figure 5.8 shows the sample density of low-energy solutions when sampling from: the noiseless, classically-computed statevector; the *IBM Aachen* Heron device with 4096 shots; or a uniformly random distribution. The noiseless statevector achieves an 8.8× improvement in the rate of optimal solutions over random samples; the physical device achieves a factor of 3.4×. Since we are considering a sampling task, increasing the sample overhead by a factor of $K > 1$ can be seen as a crude but effective form of error mitigation. For these small circuits, a relatively modest overhead would suffice to recover an equivalent set of high-quality results. As circuit size grows and errors become more common, K must also increase. For QAOA circuits, provable performance bounds result from taking $K = (\mathrm{LF})^p$, where LF is the *layer fidelity* of the cost Hamiltonian layer [131]. As the experiment size increases, a low-cost calibration experiment to compute the LF will enable our algorithms to converge.

5.4.2 QAOA applied to the HUBO formulation

The circuit compilation techniques of [59, 130] are designed explicitly for QUBO-QAOA. First, the SWAP strategies proposed are chosen to ensure that all *pairs* of qubits are neighbouring on the physical hardware after a minimal number of SWAP layers. This suffices to ensure that any QUBO-QAOA cost layer can be implemented. However, there is no guarantee that higher-order interactions become feasible. Moreover, the logic

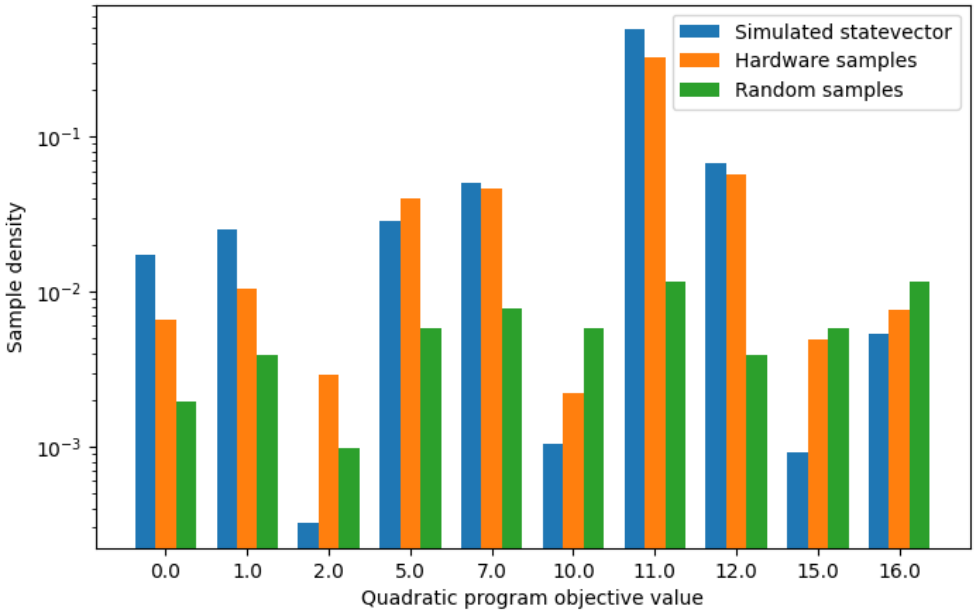


Fig. 5.8 A histogram comparing the low-energy sample density of an optimised QAOA circuit when it is simulated noiselessly and when it is executed on a quantum device. Also shown are the sample densities of a uniformly random distribution.

for ordering interactions between SWAP layers relied on an edge-colouring argument that holds only for degree-2 interactions.

We address each of these limitations, leading to a novel method of HUBO-QAOA circuit compilation that achieves a $30 \sim 60\%$ reduction in circuit depth compared to the state-of-the-art general-purpose *Qiskit* transpiler.

For the former issue, we first modify the SWAP strategies defined in [59] to increase mixing and, consequently, the number of higher-order interactions. We then define a similar CNF formula as introduced by Matsuo et al. [130]. Assume we have fixed some number of swap layers, d. Let $N_d(x_1, \ldots, x_{k-1})$ be the set of qubits y for which $\{x_1, \ldots, x_{k-1}, y\}$ form a contiguous group at some point in the first d layers of SWAPs. For an interaction between virtual qubits $(i_1, \ldots, i_k)$, we now add terms

$$\Big\{ \bigvee_{l=1}^{k-1} \neg(i_l,\, x_l) \vee \bigvee_{y \in N_d(x_1,\ldots,x_{k-1})} (i_k,\, y) : (x_1, \ldots, x_{k-1}) \in X^{\otimes k-1} \Big\}. \tag{5.14}$$

In words, each term reads "if $i_1, \ldots i_{k-1}$ are mapped to $x_1, \ldots, x_{k-1}$, then i_k must be mapped to $y \in N_d(x_1, \ldots, x_{k-1})$."

Due to the large number of CNF terms and the computational cost of calculating $N_d(x_1, \ldots, x_{k-1})$ for large k, we usually limit the maximum size of interactions that

we consider for the CNF formula to six. Moreover, we can no longer guarantee that the CNF formula becomes satisfiable for some sufficiently large d. We therefore take a MAX-SAT approach, aiming to find an initial qubit layout that implements as many interactions as possible. The remaining interactions are applied one by one "manually" at the end, using SWAPs as necessary. Rather than a provably minimum number of layers as in the QUBO case, we now seek a SWAP depth d that effectively balances the cost of SWAP layers with the cost incurred by manually implementing the interactions at the end. We sample a range of d and choose the one which minimises the overall circuit depth.

Once we have chosen a SWAP depth and the corresponding initial qubit layout, the interactions between each SWAP layer must be implemented efficiently. The edge-colouring strategy no longer holds due to the presence of multi-qubit interactions, which must be compiled down to sequences of two-qubit gates. In a naive implementation, the multi-qubit Z-rotations are compiled to a ladder of CX gates, a single-qubit R_z, and the inverse CX ladder. We instead implement a bespoke heuristic strategy that dramatically reduces the number of CXs. We implement two versions: the former performs multi-qubit rotations with an R_z gate, whereas the latter uses an R_{zz} gate. We identify sequences of interactions that can be performed with the R_z or R_{zz} gate in a fixed qubit location, iteratively building information onto and off the target qubit(s) between each interaction. Doing so effectively corresponds to massive cancellations in the CX ladders of a naive implementation. An example of this strategy applied to four interactions on a linear nearest-neighbour topology is shown in fig. 5.9; we have also implemented strategies on heavy-hex and grid layouts. More details on the logic behind the transpilation of interactions at each layer, including a pseudocode implementation, are provided in algorithm 3 of appendix D.

In table 5.4, we compare the results of using our bespoke transpilers with the inbuilt *Qiskit* transpiler at the highest level of optimisation. Across all instances, we see that both our methods achieve reductions in both the two-qubit gate count and depth. Our transpilers are designed with depth in mind, which explains their generally stronger performance in that regard. The R_{zz} version also uniformly outperforms the R_z version, which would generally be expected due to using a single R_{zz} in place of a pair of CZ gates around an R_z gate. We achieve at least a 32% and up to a 61% reduction in circuit depth compared to the *Qiskit* transpiler, which already contains advanced circuit compilation and layout techniques. These optimisations are crucial for near-term devices, where shallow circuits mitigate noise.

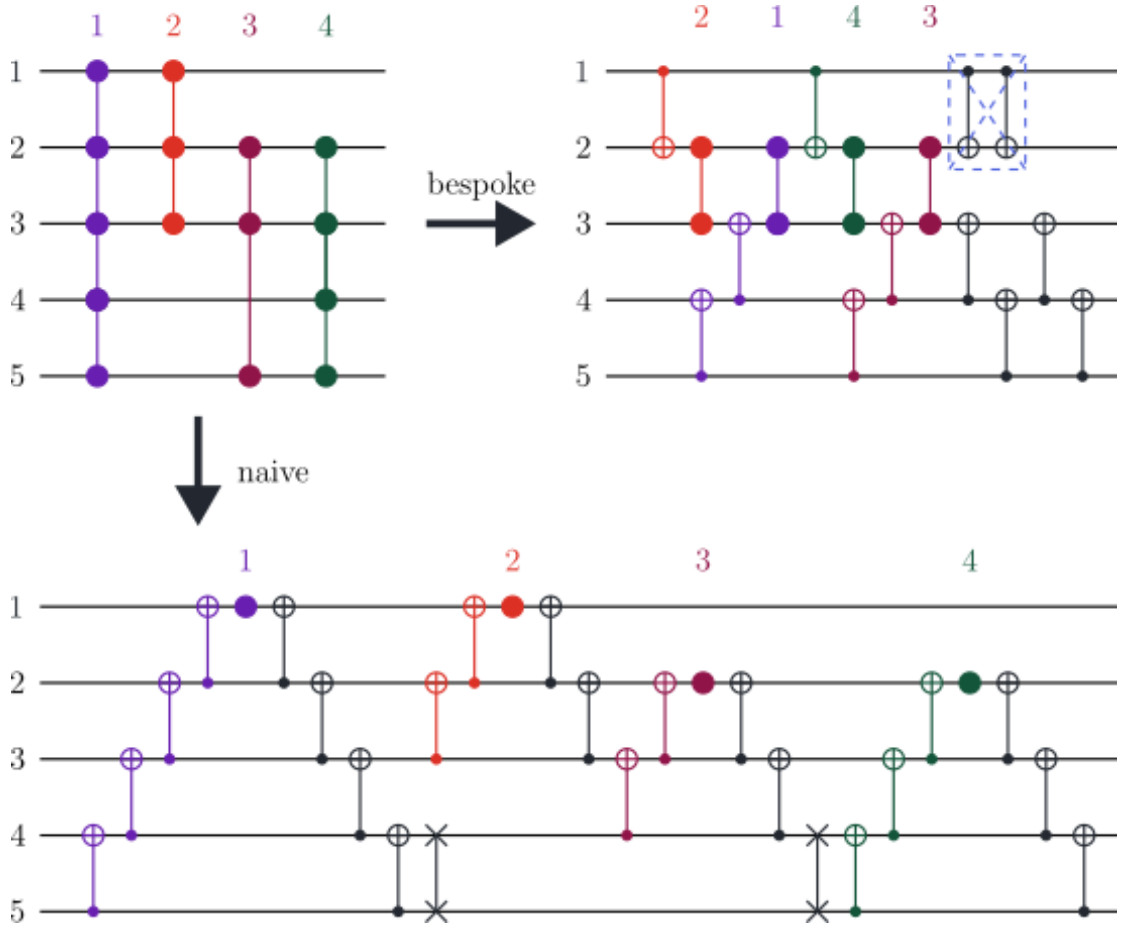


Fig. 5.9 An example of bespoke HUBO-QAOA transpilation on a linear nearest-neighbour topology. Dots represent qubits participating in a Z-rotation. (Upper-left) A sequence of 4 interactions involving 3 to 5 qubits. (Upper-right) The compiled circuit, using our transpilation strategy, comprises 10 CX gates and 4 R_{zz} gates. The orders of interactions 1 & 2 and 3 & 4 were swapped. Interaction 3 was implemented without any SWAP gates, even though it was not connected in the topology. Two of the CX gates added as part of the transpilation logic can be trivially cancelled. (Lower) The naively compiled circuit comprises 22 CX gates, 2 SWAP gates and 4 R_z gates. *Qiskit*'s inbuilt transpiler at the highest optimisation level achieves only two fewer two-qubit gates.

Table 5.4 A table comparing the two-qubit gate count and two-qubit depth of HUBO-QAOA cost Hamiltonians across different circuit transpilation methods. The "Default" columns refer to using the standard *Qiskit* transpiler with optimisation level 3. The R_z and R_{zz} columns refer to using our custom transpilers, where the rotations are performed with either R_z or R_{zz} gates, respectively.

Qubits	Two-qubit gate count			% count reduction		Two-qubit depth			% depth reduction	
	Default	R_z	R_{zz}	R_z	R_{zz}	Default	R_z	R_{zz}	R_z	R_{zz}
6	106	89	78	16.0	26.4	95	66	57	30.5	40.0
8	827	622	564	24.8	31.8	748	492	445	34.2	40.5
9	367	240	200	34.6	45.5	322	171	145	46.9	55.0
10	3523	3193	2727	9.4	22.6	3086	2418	2093	21.6	32.2
12	600	424	396	29.3	34.0	505	235	198	53.5	60.8
15	7028	6488	5416	7.7	22.9	6024	4303	3623	28.6	39.9
16	3142	2336	2045	25.7	34.9	2589	1171	1084	54.8	58.1
20	3724	3108	2756	16.5	26.0	3109	1481	1260	52.4	59.5

5.5 Discussion and future work

This chapter demonstrates a proof-of-concept pipeline for pangenome-guided short-read assembly using binary optimisation methods. The approach improves assembly contiguity while maintaining accuracy and is scalable to both classical and quantum architectures. Optimisation-based methods are robust in repetitive regions, where noise in copy-number estimation affects prescriptive tools such as *pathfinder*.

Our pipeline comprises three main stages: graph annotation, copy number estimation and path finding. Among annotation tools, *minigraph* produced the most accurate assemblies with minimal breaks and indels, though at the cost of slightly more contigs. For copy number estimation, we rely solely on *pathfinder*'s capabilities. Further exploration in this direction could be fruitful; one option is to train a machine-learning model to perform this task.

Our classical binary optimisation methods perform competitively with the state-of-the-art *pathfinder* tool. On the intermediate-size instances we test, the classical optimisers report good results within reasonably short time limits, often reporting better solutions than *pathfinder* on specific problems. Such solvers could complement existing pipelines by generating alternative assemblies for consensus refinement.

Quantum annealing results were encouraging on a set of middling-complexity problems, with solution quality broadly similar to *pathfinder*, even with only short time limits. This is a positive sign of the potential for QA to solve industry-scale problems in the future as hardware continues to improve. However, there is currently no evidence of utility in using these systems for our problem.

QUBO-QAOA simulations confirmed feasibility on small problems, and HUBO-QAOA methods provide a path to tackling larger ones. While performing a hardware experiment at industry scale is required to make any serious claims about quantum utility in this domain, we see this as a promising proof-of-principle.

Key open directions include:

1. **Edge weight integration:** incorporating edge weights into QUBO or HUBO formulations could better capture biological linkage and reduce the number of false joins, but requires handling inherently higher order terms and increasing interaction density in cost Hamiltonians.

2. **Node-size weighting:** larger nodes exhibit more reliable kmer statistics and are more significant for solution quality. They could be emphasised by modifying eq. (5.1) to

$$C_G(W) = \sum_{v \in V} \log(L(v)) \Big(\#W(v) - w(v) \Big)^2, \tag{5.15}$$

   where $L(v)$ is the sequence length of v.

3. **Diploid extensions:** benchmarking the performance of solvers on the *Diploid Tangle Resolution* problem will require more complex post-processing and analysis, such as allele-phasing metrics to evaluate correct chromosome assignment.

Overall, these results highlight the promise of hybrid quantum-classical optimisation for large-scale bioinformatics problems and lay the groundwork for exploring quantum utility in computational genomics.

Chapter 6

Conclusions

This thesis has explored methods through which theoretical research can advance near-term quantum computational utility. The overarching objective has been to design algorithms that address classically challenging, application-relevant tasks, while remaining compatible with the limitations of present-day quantum hardware.

The work took place at a pivotal stage in the evolution of quantum computing. Device sizes have expanded from tens to hundreds of qubits, yet error rates remain significant. Although hardware roadmaps promise steady progress, achieving practical impact in the short term will depend on co-designing algorithms in concert with the constraints and capabilities of physical devices.

Chapter 3: Simulation and learning of matchgate circuits

Chapter 3 investigated the **simulability** and **learnability** of matchgate-based quantum computers, to delineate the boundary between classical and quantum computational power. Understanding where efficient classical simulation remains possible is essential: quantum utility cannot arise in regimes where classical methods can reproduce quantum behaviour in polynomial time.

We derived an independent set of constraints that fully characterise Gaussian states in eq. (3.28), enabling analytical results such as the classification of all linearly dependent triples of Gaussian states in proposition 3.1.2 and an upper bound on the Gaussian fidelity of Haar-random states in proposition 3.1.4.

In section 3.2, we developed a polynomial-query learning algorithm for Gaussian operations, proposition 3.2.1.

Finally, we proposed a definition of the *matchgate hierarchy*, definition 3.3.1, analogous to the well-known Clifford hierarchy, and showed that the Clifford hierarchy is contained within it. We then proved that operations from the k^{th} level of the hierarchy

can also be learned in polynomially many queries for any k in proposition 3.3.1.

Chapter 4: Parameterized quantum algorithms for closest string problems

Chapter 4 tackled the **Closest String** and **Closest Substring Problems**, fundamental tasks in string processing and especially bioinformatics. Framing them in the context of parameterized complexity, we developed a trio of algorithms for the CSP in propositions 4.0.1 to 4.0.3 and one for the CSSP, proposition 4.0.4.

The proposed algorithms for the CSP were:

1. A generic nested quantum search applicable to any distance metric.
2. A random-walk-based algorithm for the Hamming distance; and
3. A quantum dynamic programming algorithm for the Levenshtein distance.

We also presented a quantum approach to the CSSP based on a quantisation of a search-based classical algorithm.

Each algorithm represents a speedup over classical baselines in its specialised parameter regimes. Importantly, we established that the nested quantum search is tight under the QSETH conjecture, marking one of the first parameterized tightness results for quantum algorithms in computational biology.

Chapter 5: Pangenome-guided sequence assembly via binary optimisation

Chapter 5 extended the exploration of quantum utility to bioinformatics, developing a novel end-to-end framework for **pangenome-guided sequence assembly** based on binary optimisation. The problem is chosen to be classically challenging yet amenable to quantum optimisation methods.

Classical experiments using *Gurobi* and *MQLib* demonstrated that optimisation-based assembly can produce high-quality results, comparable to and occasionally surpassing the classical state-of-the-art *pathfinder*, as seen in table 5.2. All pangenome-guided methods significantly outperformed *de novo* assemblers across key metrics, especially contig count.

Quantum annealing tests on *D-Wave* hardware confirmed feasibility for moderate problem sizes, and classical simulations of QAOA validated the proof-of-principle on gate-model devices. A HUBO-QAOA form reduced the qubit resources from $\mathcal{O}(n^2)$ to $\mathcal{O}(n\log(n))$, and a custom transpilation method reduced the corresponding circuit

depth by up to 60%, crucial steps for practical execution on noisy intermediate-scale hardware.

The pipeline introduced in this chapter lays the foundation for quantum elements in genomic analysis, highlighting current limitations and demonstrating promising routes to quantum utility.

Outlook and future directions

Collectively, the research presented in this thesis advances both the theoretical understanding and the practical realisation of near-term quantum algorithms. Several main insights arise:

1. **Boundaries of simulation:** Identifying classically tractable circuit classes informs domains where genuine quantum advantage could arise.

2. **Parameterized complexity and algorithm design:** Fine-grained parametrisation allows targeted speedups rather than general asymptotic ones.

3. **Domain-specific design:** Integrating quantum optimisation within bioinformatics pipelines, such as pangenome-guided sequence assembly, leads to new algorithmic ideas and improved performance.

Future work can build on these ideas.

The main open question of chapter 3 is to find low-rank decompositions of Gaussian magic states, or to conclusively prove that none can exist, firmly defining the boundary of classical simulation for these circuits.

Quantisations of more advanced or heuristic classical algorithms could give further quantum speedups for the problems discussed in chapter 4.

Finally, from chapter 5, applying HUBO-QAOA approaches to biologically realistic pangenomes and testing them on next-generation quantum processes will reveal the potential for quantum utility in this domain.

Quantum computing is at a transitional moment: powerful enough to tackle practical tasks, but constrained by noise and scale. This thesis contributes to navigating that transition by proposing algorithms that are hardware-aware, domain-informed, and mathematically grounded. By pursuing practical utility alongside theoretical rigour, we move closer to a future where quantum computers play a role in solving problems of genuine scientific relevance.

Bibliography

[1] Joshua Cudby and Sergii Strelchuk. *Gaussian decomposition of magic states for matchgate computations*. arXiv:2307.12654 [quant-ph]. Apr. 2024. DOI: 10.48550/arXiv.2307.12654. URL: http://arxiv.org/abs/2307.12654 (visited on 04/16/2024).

[2] Joshua Cudby and Sergii Strelchuk. *Learning Gaussian Operations and the Matchgate Hierarchy*. arXiv:2407.12649 [quant-ph]. July 2024. DOI: 10.48550/arXiv.2407.12649. URL: http://arxiv.org/abs/2407.12649 (visited on 06/10/2025).

[3] Joshua Cudby and Sergii Strelchuk. *Parameterized quantum algorithms for closest string problems*. arXiv:2510.15529 [quant-ph]. Oct. 2025. DOI: 10.48550/arXiv.2510.15529. URL: http://arxiv.org/abs/2510.15529 (visited on 10/21/2025).

[4] Josh Cudby et al. "Pangenome-guided sequence assembly via binary optimization". en. In: *Briefings in Bioinformatics* 27.1 (Jan. 2026), bbag084. ISSN: 1467-5463, 1477-4054. DOI: 10.1093/bib/bbag084. URL: https://academic.oup.com/bib/article/doi/10.1093/bib/bbag084/8499369 (visited on 02/27/2026).

[5] Anthony Hey. *Feynman And Computation*. 1st. CRC Press, 2002. ISBN: 978-0-429-50045-9.

[6] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". en. In: *SIAM Journal on Computing* 26.5 (Oct. 1997). arXiv:quant-ph/9508027, pp. 1484–1509. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539795293172. URL: http://arxiv.org/abs/quant-ph/9508027 (visited on 06/13/2025).

[7] Eric W. Weisstein. *Number Field Sieve*. May 2025. URL: https://mathworld.wolfram.com/NumberFieldSieve.html (visited on 06/13/2025).

[8] Joschka Roffe. "Quantum Error Correction: An Introductory Guide". In: *Contemporary Physics* 60.3 (July 2019). arXiv:1907.11157 [quant-ph], pp. 226–245. ISSN: 0010-7514, 1366-5812. DOI: 10.1080/00107514.2019.1667078. URL: http://arxiv.org/abs/1907.11157 (visited on 06/13/2025).

[9] W. Cai et al. "Bosonic quantum error correction codes in superconducting quantum circuits". In: *Fundamental Research* 1.1 (Jan. 2021). arXiv:2010.08699 [quant-ph], pp. 50–67. ISSN: 26673258. DOI: 10.1016/j.fmre.2020.12.006. URL: http://arxiv.org/abs/2010.08699 (visited on 06/13/2025).

[10] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010. ISBN: 978-0-511-99277-3. URL: https://books.google.co.uk/books?id=JRz3jgEACAAJ.

[11] Jacob Biamonte et al. "Quantum Machine Learning". en. In: *Nature* 549.7671 (Sept. 2017). arXiv:1611.09347 [quant-ph], pp. 195–202. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature23474. URL: http://arxiv.org/abs/1611.09347 (visited on 06/13/2025).

[12] Yaswitha Gujju, Atsushi Matsuo, and Rudy Raymond. *Quantum Machine Learning on Near-Term Quantum Devices: Current State of Supervised and Unsupervised Techniques for Real-World Applications*. arXiv:2307.00908 [quant-ph]. June 2024. DOI: 10.48550/arXiv.2307.00908. URL: http://arxiv.org/abs/2307.00908 (visited on 06/13/2025).

[13] Wilfred Salmon. "Learning in Quantum Mechanics". PhD thesis. University of Cambridge, 2024. URL: https://www.repository.cam.ac.uk/handle/1810/384800.

[14] Ivan Kassal et al. "Simulating chemistry using quantum computers". In: *Annual Review of Physical Chemistry* 62.1 (May 2011). arXiv:1007.2648 [quant-ph], pp. 185–207. ISSN: 0066-426X, 1545-1593. DOI: 10.1146/annurev-physchem-032210-103512. URL: http://arxiv.org/abs/1007.2648 (visited on 06/13/2025).

[15] Yudong Cao et al. "Quantum Chemistry in the Age of Quantum Computing". en. In: *Chemical Reviews* 119.19 (Oct. 2019), pp. 10856–10915. ISSN: 0009-2665, 1520-6890. DOI: 10.1021/acs.chemrev.8b00803. URL: https://pubs.acs.org/doi/10.1021/acs.chemrev.8b00803 (visited on 06/13/2025).

[16] Mitchell Chiew. "Defining, classifying and optimising fermion–qubit mappings". PhD thesis. University of Cambridge, 2025. URL: https://www.repository.cam.ac.uk/handle/1810/385205.

[17] Mahdi Boroujeni et al. "Approximating Edit Distance in Truly Subquadratic Time: Quantum and MapReduce". en. In: *Journal of the ACM* 68.3 (June 2021), pp. 1–41. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/3456807. URL: https://dl.acm.org/doi/10.1145/3456807 (visited on 02/12/2024).

[18] Aritra Sarkar et al. "QiBAM: Approximate Sub-String Index Search on Quantum Accelerators Applied to DNA Read Alignment". en. In: *Electronics* 10.19 (Oct. 2021), p. 2433. ISSN: 2079-9292. DOI: 10.3390/electronics10192433. URL: https://www.mdpi.com/2079-9292/10/19/2433 (visited on 10/17/2023).

[19] Daniel Gibney et al. *Near-Optimal Quantum Algorithms for Bounded Edit Distance and Lempel-Ziv Factorization*. arXiv:2311.01793 [quant-ph]. Nov. 2023. URL: http://arxiv.org/abs/2311.01793 (visited on 01/08/2024).

[20] Jing-Kai Fang et al. "Divide-and-Conquer Quantum Algorithm for Hybrid de novo Genome Assembly of Short and Long Reads". en. In: *PRX Life* 2.2 (Apr. 2024), p. 023006. ISSN: 2835-8279. DOI: 10.1103/PRXLife.2.023006. URL: https://link.aps.org/doi/10.1103/PRXLife.2.023006 (visited on 04/23/2025).

[21] W. Pauli. "Über den Zusammenhang des Abschlusses der Elektronengruppen im Atom mit der Komplexstruktur der Spektren". In: *Zeitschrift für Physik* 31.1 (Feb. 1925), pp. 765–783. ISSN: 0044-3328. DOI: 10.1007/BF02980631. URL: https://doi.org/10.1007/BF02980631.

[22] P. Jordan and E. Wigner. "Über das Paulische Äquivalenzverbot". de. In: *Zeitschrift für Physik* 47.9 (Sept. 1928), pp. 631–651. ISSN: 0044-3328. DOI: 10.1007/BF01331938. URL: https://doi.org/10.1007/BF01331938 (visited on 04/16/2024).

[23] A. Yu. Kitaev. "Quantum computations: algorithms and error correction". In: *Russian Math. Surveys* 52.6 (1997), pp. 1191–1249. DOI: https://doi.org/10.1070/RM1997v052n06ABEH002155.

[24] Daniel Gottesman. *The Heisenberg Representation of Quantum Computers*. arXiv:quant-ph/9807006. July 1998. DOI: 10.48550/arXiv.quant-ph/9807006. URL: http://arxiv.org/abs/quant-ph/9807006 (visited on 06/16/2025).

[25] H. Bombin and M. A. Martin-Delgado. "Topological Quantum Distillation". In: *Physical Review Letters* 97.18 (Oct. 2006). arXiv:quant-ph/0605138, p. 180501. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.97.180501. URL: http://arxiv.org/abs/quant-ph/0605138 (visited on 11/11/2025).

[26] Daniel Gottesman and Isaac L. Chuang. "Quantum Teleportation is a Universal Computational Primitive". In: *Nature* 402.6760 (Nov. 1999). arXiv:quant-ph/9908010, pp. 390–393. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/46503. URL: http://arxiv.org/abs/quant-ph/9908010 (visited on 08/30/2023).

[27] Xinlan Zhou, Debbie W. Leung, and Isaac L. Chuang. "Methodology for quantum logic gate constructions". In: *Physical Review A* 62.5 (Oct. 2000). arXiv:quant-ph/0002039, p. 052316. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.62.052316. URL: http://arxiv.org/abs/quant-ph/0002039 (visited on 04/16/2024).

[28] Tefjol Pllaha et al. "Un-Weyl-ing the Clifford Hierarchy". In: *Quantum* 4 (Dec. 2020). arXiv:2006.14040 [quant-ph], p. 370. ISSN: 2521-327X. DOI: 10.22331/q-2020-12-11-370. URL: http://arxiv.org/abs/2006.14040 (visited on 03/26/2024).

[29] Nadish de Silva. "Efficient quantum gate teleportation in higher dimensions". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 477.2251 (July 2021). arXiv:2011.00127 [math-ph, physics:quant-ph], p. 20200865. ISSN: 1364-5021, 1471-2946. DOI: 10.1098/rspa.2020.0865. URL: http://arxiv.org/abs/2011.00127 (visited on 04/16/2024).

[30] Narayanan Rengaswamy, Robert Calderbank, and Henry D. Pfister. "Unifying the Clifford Hierarchy via Symmetric Matrices over Rings". In: *Physical Review A* 100.2 (Aug. 2019). arXiv:1902.04022 [quant-ph], p. 022304. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.100.022304. URL: http://arxiv.org/abs/1902.04022 (visited on 04/16/2024).

[31] Richard A. Low. "Learning and Testing Algorithms for the Clifford Group". In: *Physical Review A* 80.5 (Nov. 2009). arXiv:0907.2833 [quant-ph], p. 052314. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.80.052314. URL: http://arxiv.org/abs/0907.2833 (visited on 03/26/2024).

[32] Sergey Bravyi and Alexei Kitaev. "Universal quantum computation with ideal Clifford gates and noisy ancillas". en. In: *Physical Review A* 71.2 (Feb. 2005), p. 022316. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.71.022316. URL: https://link.aps.org/doi/10.1103/PhysRevA.71.022316 (visited on 07/09/2025).

[33] Pedro Sales Rodriguez et al. *Experimental Demonstration of Logical Magic State Distillation*. arXiv:2412.15165 [quant-ph]. Dec. 2024. DOI: 10.48550/arXiv.2412.15165. URL: http://arxiv.org/abs/2412.15165 (visited on 07/09/2025).

[34] Theodore J. Yoder et al. *Tour de gross: A modular quantum computer based on bivariate bicycle codes*. arXiv:2506.03094 [quant-ph]. June 2025. DOI: 10.48550/arXiv.2506.03094. URL: http://arxiv.org/abs/2506.03094 (visited on 11/18/2025).

[35] Sergey Bravyi, Graeme Smith, and John Smolin. "Trading classical and quantum computational resources". In: *Physical Review X* 6.2 (June 2016). arXiv:1506.01396 [quant-ph], p. 021043. ISSN: 2160-3308. DOI: 10.1103/PhysRevX.6.021043. URL: http://arxiv.org/abs/1506.01396 (visited on 07/09/2025).

[36] Hammam Qassim, Hakop Pashayan, and David Gosset. "Improved upper bounds on the stabilizer rank of magic states". In: *Quantum* 5 (Dec. 2021). arXiv:2106.07740 [quant-ph], p. 606. ISSN: 2521-327X. DOI: 10.22331/q-2021-12-20-606. URL: http://arxiv.org/abs/2106.07740 (visited on 07/09/2025).

[37] Leslie G. Valiant. "Quantum computers that can be simulated classically in polynomial time". In: *Proceedings of the thirty-third annual ACM symposium on Theory of computing*. STOC '01. New York, NY, USA: Association for Computing Machinery, July 2001, pp. 114–123. ISBN: 978-1-58113-349-3. DOI: 10.1145/380752.380785. URL: https://dl.acm.org/doi/10.1145/380752.380785 (visited on 04/12/2024).

[38] Barbara M. Terhal and David P. DiVincenzo. "Classical simulation of noninteracting-fermion quantum circuits". In: *Physical Review A* 65.3 (Mar. 2002). arXiv:quant-ph/0108010, p. 032325. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.65.032325. URL: http://arxiv.org/abs/quant-ph/0108010 (visited on 08/24/2023).

[39] Richard Jozsa and Akimasa Miyake. "Matchgates and classical simulation of quantum circuits". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 464.2100 (Dec. 2008). arXiv:0804.4050 [quant-ph], pp. 3089–3106. ISSN: 1364-5021, 1471-2946. DOI: 10.1098/rspa.2008.0189. URL: http://arxiv.org/abs/0804.4050 (visited on 08/09/2023).

[40] Sergey Bravyi. *Lagrangian representation for fermionic linear optics*. arXiv:quant-ph/0404180. Sept. 2004. DOI: 10.48550/arXiv.quant-ph/0404180. URL: http://arxiv.org/abs/quant-ph/0404180 (visited on 10/03/2025).

[41] Martin Hebenstreit et al. "All pure fermionic non-Gaussian states are magic states for matchgate computations". en. In: *Physical Review Letters* 123.8 (Aug. 2019). arXiv:1905.08584 [quant-ph], p. 080503. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.123.080503. URL: http://arxiv.org/abs/1905.08584 (visited on 08/20/2024).

[42] David Deutsch and Richard Jozsa. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (Jan. 1997), pp. 553–558. DOI: 10.1098/rspa.1992.0167. URL: https://doi.org/10.1098/rspa.1992.0167 (visited on 07/11/2025).

[43] Andris Ambainis. "Quantum query algorithms and lower bounds". en. In: *Classical and New Paradigms of Computation and their Complexity Hierarchies*. Ed. by Benedikt Löwe, Boris Piwinger, and Thoralf Räsch. Dordrecht: Springer Netherlands, 2004, pp. 15–32. ISBN: 978-1-4020-2776-5. DOI: 10.1007/978-1-4020-2776-5_2.

[44] Harry Buhrman and Ronald de Wolf. "Complexity measures and decision tree complexity: a survey". In: *Theoretical Computer Science*. Complexity and Logic 288.1 (Oct. 2002), pp. 21–43. ISSN: 0304-3975. DOI: 10.1016/S0304-3975(01)00144-X. URL: https://www.sciencedirect.com/science/article/pii/S030439750100144X (visited on 04/03/2025).

[45] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. arXiv:quant-ph/9605043. Nov. 1996. DOI: 10.48550/arXiv.quant-ph/9605043. URL: http://arxiv.org/abs/quant-ph/9605043 (visited on 07/29/2024).

[46] Christoph Durr and Peter Hoyer. *A Quantum Algorithm for Finding the Minimum*. arXiv:quant-ph/9607014. Jan. 1999. DOI: 10.48550/arXiv.quant-ph/9607014. URL: http://arxiv.org/abs/quant-ph/9607014 (visited on 07/29/2024).

[47] Frédéric Magniez et al. "Search via Quantum Walk". In: *SIAM Journal on Computing* 40.1 (Jan. 2011). arXiv:quant-ph/0608026, pp. 142–164. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/090745854. URL: http://arxiv.org/abs/quant-ph/0608026 (visited on 03/05/2024).

[48] Andris Ambainis. *Quantum walk algorithm for element distinctness*. arXiv:quant-ph/0311001. Apr. 2014. DOI: 10.48550/arXiv.quant-ph/0311001. URL: http://arxiv.org/abs/quant-ph/0311001 (visited on 03/12/2024).

[49] M. Szegedy. "Quantum Speed-Up of Markov Chain Based Algorithms". en. In: *45th Annual IEEE Symposium on Foundations of Computer Science*. Rome, Italy: IEEE, 2004, pp. 32–41. ISBN: 978-0-7695-2228-9. DOI: 10.1109/FOCS.2004.53. URL: http://ieeexplore.ieee.org/document/1366222/ (visited on 03/12/2024).

[50] Andris Ambainis et al. *Quantum Speedups for Exponential-Time Dynamic Programming Algorithms*. arXiv:1807.05209 [quant-ph]. July 2018. DOI: 10.48550/arXiv.1807.05209. URL: http://arxiv.org/abs/1807.05209 (visited on 07/28/2023).

[51] Adam Glos et al. *Quantum speedups for dynamic programming on n-dimensional lattice graphs*. arXiv:2104.14384 [quant-ph]. May 2021. DOI: 10.48550/arXiv.2104.14384. URL: http://arxiv.org/abs/2104.14384 (visited on 04/30/2024).

[52] C. H. Bennett. "Logical Reversibility of Computation". In: *IBM Journal of Research and Development* 17.6 (Nov. 1973), pp. 525–532. ISSN: 0018-8646. DOI: 10.1147/rd.176.0525. URL: https://ieeexplore.ieee.org/document/5391327/ (visited on 04/15/2025).

[53] P. Zuliani. "Logical reversibility". en. In: *IBM Journal of Research and Development* 45.6 (Nov. 2001), pp. 807–818. ISSN: 0018-8646, 0018-8646. DOI: 10.1147/rd.456.0807. URL: http://ieeexplore.ieee.org/document/5389037/ (visited on 04/08/2025).

[54] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. arXiv:1411.4028 [quant-ph]. Nov. 2014. DOI: 10.48550/arXiv.1411.4028. URL: http://arxiv.org/abs/1411.4028 (visited on 06/13/2025).

[55] Stuart Hadfield, Tad Hogg, and Eleanor G. Rieffel. *Analytical Framework for Quantum Alternating Operator Ans\"atze*. en. May 2021. DOI: 10.1088/2058-9565/aca3ce. URL: https://arxiv.org/abs/2105.06996v2 (visited on 06/13/2025).

[56] Ashley Montanaro and Leo Zhou. *Quantum speedups in solving near-symmetric optimization problems by low-depth QAOA*. arXiv:2411.04979 [quant-ph]. Feb. 2025. DOI: 10.48550/arXiv.2411.04979. URL: http://arxiv.org/abs/2411.04979 (visited on 06/13/2025).

[57] Leo Zhou et al. "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices". en. In: *Physical Review X* 10.2 (June 2020). arXiv:1812.01041 [cond-mat, physics:quant-ph], p. 021067. ISSN: 2160-3308. DOI: 10.1103/PhysRevX.10.021067. URL: http://arxiv.org/abs/1812.01041 (visited on 06/24/2024).

[58] Natasha Sachdeva et al. *Quantum optimization using a 127-qubit gate-model IBM quantum computer can outperform quantum annealers for nontrivial binary optimization problems*. en. arXiv:2406.01743 [quant-ph]. June 2024. URL: http://arxiv.org/abs/2406.01743 (visited on 06/14/2024).

[59] Johannes Weidenfeller et al. "Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware". en. In: *Quantum* 6 (Dec. 2022). arXiv:2202.03459 [quant-ph], p. 870. ISSN: 2521-327X. DOI: 10.22331/q-2022-12-07-870. URL: http://arxiv.org/abs/2202.03459 (visited on 06/13/2025).

[60] Adam Auton et al. "A global reference for human genetic variation". In: *Nature* 526.7571 (Oct. 2015), pp. 68–74. ISSN: 1476-4687. DOI: 10.1038/nature15393. URL: https://doi.org/10.1038/nature15393.

[61] F. Sanger, S. Nicklen, and A. R. Coulson. "DNA sequencing with chain-terminating inhibitors". en. In: *Proceedings of the National Academy of Sciences of the United States of America* 74.12 (Dec. 1977), p. 5463. DOI: 10.1073/pnas.74.12.5463. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC431765/ (visited on 06/05/2025).

[62] Kris A. Wetterstrand. *DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP)*. en. May 2023. URL: https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data (visited on 05/20/2025).

[63] E. W. Myers. "Toward simplifying and accurately formulating fragment assembly". eng. In: *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology* 2.2 (1995), pp. 275–290. ISSN: 1066-5277. DOI: 10.1089/cmb.1995.2.275.

[64] Chelsea A Matthews et al. "A gentle introduction to pangenomics". en. In: *Briefings in Bioinformatics* 25.6 (Sept. 2024), bbae588. ISSN: 1467-5463, 1477-4054. DOI: 10.1093/bib/bbae588. URL: https://academic.oup.com/bib/article/doi/10.1093/bib/bbae588/7902219 (visited on 06/05/2025).

[65] Glenn Hickey et al. "Genotyping structural variants in pangenome graphs using the vg toolkit". In: *Genome Biology* 21.1 (Feb. 2020), p. 35. ISSN: 1474-760X. DOI: 10.1186/s13059-020-1941-7. URL: https://doi.org/10.1186/s13059-020-1941-7 (visited on 06/09/2025).

[66] Jana Ebler et al. "Pangenome-based genome inference allows efficient and accurate genotyping across a wide spectrum of variant classes". en. In: *Nature Genetics* 54.4 (Apr. 2022), pp. 518–525. ISSN: 1546-1718. DOI: 10.1038/s41588-022-01043-w. URL: https://www.nature.com/articles/s41588-022-01043-w (visited on 06/09/2025).

[67] V. I. Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". ru. In: *Dokl. Akad. Nauk SSSR* 163.4 (1965), pp. 845–848.

[68] Saul B. Needleman and Christian D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins". In: *Journal of Molecular Biology* 48.3 (Mar. 1970), pp. 443–453. ISSN: 0022-2836. DOI: 10.1016/0022-2836(70)90057-4. URL: https://www.sciencedirect.com/science/article/pii/0022283670900574 (visited on 04/17/2024).

[69] Ragnar Groot Koerkamp. *A*PA2: up to 20 times faster exact global alignment.* en. Pages: 2024.03.24.586481 Section: New Results. Mar. 2024. DOI: 10.1101/2024.03.24.586481. URL: https://www.biorxiv.org/content/10.1101/2024.03.24.586481v1 (visited on 04/26/2024).

[70] J. Kevin Lanctot et al. "Distinguishing string selection problems". en. In: *Information and Computation* 185.1 (Aug. 2003), pp. 41–55. ISSN: 08905401. DOI: 10.1016/S0890-5401(03)00057-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S0890540103000579 (visited on 02/15/2024).

[71] H. T. Wareham et al. "STOCHASTIC HEURISTIC ALGORITHMS FOR TARGET MOTIF IDENTIFICATION (EXTENDED ABSTRACT)". en. In: *Biocomputing 2000*. Honolulu, Hawaii, USA: WORLD SCIENTIFIC, Dec. 1999, pp. 392–403. DOI: 10.1142/9789814447331_0037. URL: http://www.worldscientific.com/doi/abs/10.1142/9789814447331_0037 (visited on 06/19/2025).

[72] Timothy L Bailey. "STREME: accurate and versatile sequence motif discovery". In: *Bioinformatics* 37.18 (Sept. 2021), pp. 2834–2840. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab203. URL: https://doi.org/10.1093/bioinformatics/btab203 (visited on 06/19/2025).

[73] Jian Ye et al. "Primer-BLAST: A tool to design target-specific primers for polymerase chain reaction". In: *BMC Bioinformatics* 13.1 (June 2012), p. 134. ISSN: 1471-2105. DOI: 10.1186/1471-2105-13-134. URL: https://doi.org/10.1186/1471-2105-13-134.

[74] Konstantinos Prousalis et al. *A novel quantum algorithm for Biological Sequence Alignment using Quantum Accelerated Mapping in Seed-and-Extend Technique.* en. May 2024. DOI: 10.21203/rs.3.rs-4305700/v1. URL: https://www.researchsquare.com/article/rs-4305700/v1 (visited on 10/28/2024).

[75] Lloyd C. L. Hollenberg. "Fast Quantum Search Algorithms in Protein Sequence Comparison - Quantum Biocomputing". In: *Physical Review E* 62.5 (Nov. 2000). arXiv:quant-ph/0002076, pp. 7532–7535. ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.62.7532. URL: http://arxiv.org/abs/quant-ph/0002076 (visited on 11/01/2023).

[76] Shyan Akmal and Ce Jin. "Near-Optimal Quantum Algorithms for String Problems". en. In: *Algorithmica* 85.8 (Aug. 2023), pp. 2260–2317. ISSN: 1432-0541. DOI: 10.1007/s00453-022-01092-x. URL: https://doi.org/10.1007/s00453-022-01092-x (visited on 02/12/2024).

[77] Ce Jin and Jakob Nogler. *Quantum Speed-ups for String Synchronizing Sets, Longest Common Substring, and k-mismatch Matching*. arXiv:2211.15945 [cs]. Nov. 2022. DOI: 10.48550/arXiv.2211.15945. URL: http://arxiv.org/abs/2211.15945 (visited on 04/04/2025).

[78] Tomasz Kociumaka, Jakob Nogler, and Philip Wellnitz. *On the Communication Complexity of Approximate Pattern Matching*. en. arXiv:2403.18812 [quant-ph]. Mar. 2024. URL: http://arxiv.org/abs/2403.18812 (visited on 08/22/2024).

[79] Pradeep Niroula and Yunseong Nam. "A quantum algorithm for string matching". en. In: *npj Quantum Information* 7.1 (Feb. 2021). Number: 1, pp. 1–5. ISSN: 2056-6387. DOI: 10.1038/s41534-021-00369-3. URL: https://www.nature.com/articles/s41534-021-00369-3 (visited on 07/28/2023).

[80] H Ramesh and V Vinay. "String matching in O(n+m) quantum time". en. In: *Journal of Discrete Algorithms* 1.1 (Feb. 2003), pp. 103–110. ISSN: 15708667. DOI: 10.1016/S1570-8667(03)00010-8. URL: https://linkinghub.elsevier.com/retrieve/pii/S1570866703000108 (visited on 04/04/2025).

[81] Yibo Chen et al. "Haplotype-resolved assembly of diploid and polyploid genomes using quantum computing". In: *Cell Reports Methods* (Apr. 2024), p. 100754. ISSN: 2667-2375. DOI: 10.1016/j.crmeth.2024.100754. URL: https://www.sciencedirect.com/science/article/pii/S2667237524000857 (visited on 04/26/2024).

[82] Cupjin Huang et al. *Classical Simulation of Quantum Supremacy Circuits*. arXiv:2005.06787 [quant-ph]. May 2020. DOI: 10.48550/arXiv.2005.06787. URL: http://arxiv.org/abs/2005.06787 (visited on 07/11/2025).

[83] Feng Pan and Pan Zhang. *Simulating the Sycamore quantum supremacy circuits*. arXiv:2103.03074 [quant-ph]. Mar. 2021. DOI: 10.48550/arXiv.2103.03074. URL: http://arxiv.org/abs/2103.03074 (visited on 07/11/2025).

[84] Saeed Mehraban and Mehrdad Tahmasbi. *Quadratic Lower bounds on the Approximate Stabilizer Rank: A Probabilistic Approach*. arXiv:2305.10277 [quant-ph]. Mar. 2024. DOI: 10.48550/arXiv.2305.10277. URL: http://arxiv.org/abs/2305.10277 (visited on 07/11/2025).

[85] Sergey Bravyi et al. "Simulation of quantum circuits by low-rank stabilizer decompositions". en-GB. In: *Quantum* 3 (Sept. 2019), p. 181. DOI: 10.22331/q-2019-09-02-181. URL: https://quantum-journal.org/papers/q-2019-09-02-181/ (visited on 07/11/2025).

[86] Isaac L. Chuang and M. A. Nielsen. "Prescription for experimental determination of the dynamics of a quantum black box". In: *Journal of Modern Optics* 44.11-12 (Nov. 1997). arXiv:quant-ph/9610001, pp. 2455–2467. ISSN: 0950-0340, 1362-3044. DOI: 10.1080/09500349708231894. URL: http://arxiv.org/abs/quant-ph/9610001 (visited on 04/21/2024).

[87] J. F. Poyatos, J. I. Cirac, and P. Zoller. "Complete Characterization of a Quantum Process: the Two-Bit Quantum Gate". In: *Physical Review Letters* 78.2 (Jan. 1997). arXiv:quant-ph/9611013, pp. 390–393. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.78.390. URL: http://arxiv.org/abs/quant-ph/9611013 (visited on 04/21/2024).

[88] Charles H. Bennett and Stephen J. Wiesner. "Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states". In: *Physical Review Letters* 69.20 (Nov. 1992), pp. 2881–2884. DOI: 10.1103/PhysRevLett.69.2881. URL: https://link.aps.org/doi/10.1103/PhysRevLett.69.2881 (visited on 04/21/2024).

[89] L. Hormander. *An introduction to complex analysis in several variables*. ISSN. tex.lccn: 73081532. Elsevier Science, 1973. ISBN: 978-0-444-10523-3. URL: https://books.google.co.uk/books?id=MaM7AAAAQBAJ.

[90] Patrick Hayden et al. "Randomizing quantum states: Constructions and applications". In: *Communications in Mathematical Physics* 250.2 (Sept. 2004). arXiv:quant-ph/0307104, pp. 371–391. ISSN: 0010-3616, 1432-0916. DOI: 10.1007/s00220-004-1087-6. URL: http://arxiv.org/abs/quant-ph/0307104 (visited on 10/14/2025).

[91] Pauli Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". en. In: *Nature Methods* 17.3 (Mar. 2020), pp. 261–272. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2. URL: https://www.nature.com/articles/s41592-019-0686-2 (visited on 10/28/2025).

[92] Michał Oszmaniec et al. "Fermion Sampling: a robust quantum computational advantage scheme using fermionic linear optics and magic input states". en. In: *PRX Quantum* 3.2 (May 2022). arXiv:2012.15825 [math-ph, physics:quant-ph], p. 020328. ISSN: 2691-3399. DOI: 10.1103/PRXQuantum.3.020328. URL: http://arxiv.org/abs/2012.15825 (visited on 07/24/2024).

[93] Antonio Anna Mele and Yaroslav Herasymenko. *Efficient learning of quantum states prepared with few fermionic non-Gaussian gates*. arXiv:2402.18665 [quant-ph]. Feb. 2024. URL: http://arxiv.org/abs/2402.18665 (visited on 04/03/2024).

[94] Eric W Sayers et al. "GenBank 2023 update". en. In: *Nucleic Acids Research* 51.D1 (Jan. 2023), pp. D141–D144. ISSN: 0305-1048, 1362-4962. DOI: 10.1093/nar/gkac1012. URL: https://academic.oup.com/nar/article/51/D1/D141/6814443 (visited on 04/17/2024).

[95] William H.E. Day and F.R. McMorris. "Critical comparison of consensus methods for molecular sequences". en. In: *Nucleic Acids Research* 20.5 (1992), pp. 1093–1099. ISSN: 0305-1048, 1362-4962. DOI: 10.1093/nar/20.5.1093. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/20.5.1093 (visited on 04/23/2025).

[96] Quan Zou, Xiao Shan, and Yi Jiang. "A Novel Center Star Multiple Sequence Alignment Algorithm Based on Affine Gap Penalty and K-Band". In: *Physics Procedia*. 2012 International Conference on Medical Physics and Biomedical Engineering (ICMPBE2012) 33 (Jan. 2012), pp. 322–327. ISSN: 1875-3892. DOI: 10.1016/j.phpro.2012.05.069. URL: https://www.sciencedirect.com/science/article/pii/S187538921201382X (visited on 05/15/2025).

[97] Gramm, Niedermeier, and Rossmanith. "Fixed-Parameter Algorithms for CLOSEST STRING and Related Problems". en. In: *Algorithmica* 37.1 (Sept. 2003), pp. 25–42. ISSN: 1432-0541. DOI: 10.1007/s00453-003-1028-3. URL: https://doi.org/10.1007/s00453-003-1028-3 (visited on 02/27/2024).

[98] François Nicolas and Eric Rivals. "Hardness results for the center and median string problems under the weighted and unweighted edit distances". In: *Journal of Discrete Algorithms*. Combinatorial Pattern Matching (CPM) Special Issue 3.2 (June 2005), pp. 390–415. ISSN: 1570-8667. DOI: 10.1016/j.jda.2004.08.015. URL: https://www.sciencedirect.com/science/article/pii/S1570866704000620 (visited on 02/26/2024).

[99] Daniel Lokshtanov, Daniel Marx, and Saket Saurabh. "Slightly Superexponential Parameterized Problems". In: *SIAM Journal on Computing* 47.3 (Jan. 2018). arXiv:1902.08723 [cs], pp. 675–702. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/16M1104834. URL: http://arxiv.org/abs/1902.08723 (visited on 02/26/2024).

[100] Hiromitsu Maji and Taisuke Izumi. "Listing Center Strings Under the Edit Distance Metric". en. In: *Combinatorial Optimization and Applications*. Ed. by Zaixin Lu et al. Cham: Springer International Publishing, 2015, pp. 771–782. ISBN: 978-3-319-26626-8. DOI: 10.1007/978-3-319-26626-8_57.

[101] Chih Huai Cheng et al. "Efficient Algorithms for Some Variants of the Farthest String Problem". en. In: *Proceedings of Workshop on Combinatorial Math and Computation Theory*. 2004, pp. 266–272.

[102] Lusheng Wang and Binhai Zhu. "Efficient Algorithms for the Closest String and Distinguishing String Selection Problems". en. In: *Frontiers in Algorithmics*. Ed. by Xiaotie Deng, John E. Hopcroft, and Jinyun Xue. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 261–270. ISBN: 978-3-642-02270-8. DOI: 10.1007/978-3-642-02270-8_27.

[103] François Le Gall and Saeed Seddighin. "Quantum Meets Fine-grained Complexity: Sublinear Time Quantum Algorithms for String Problems". In: *Algorithmica* 85.5 (May 2023). arXiv:2010.12122 [quant-ph], pp. 1251–1286. ISSN: 0178-4617, 1432-0541. DOI: 10.1007/s00453-022-01066-z. URL: http://arxiv.org/abs/2010.12122 (visited on 02/05/2024).

[104] Ashley Montanaro. *Quantum pattern matching fast on average*. arXiv:1408.1816 [quant-ph]. Aug. 2015. DOI: 10.48550/arXiv.1408.1816. URL: http://arxiv.org/abs/1408.1816 (visited on 07/28/2023).

[105] Scott Aaronson, Daniel Grier, and Luke Schaeffer. *A Quantum Query Complexity Trichotomy for Regular Languages*. arXiv:1812.04219 [quant-ph]. Apr. 2019. DOI: 10.48550/arXiv.1812.04219. URL: http://arxiv.org/abs/1812.04219 (visited on 04/04/2025).

[106] Andrew M. Childs et al. *Quantum divide and conquer*. arXiv:2210.06419 [quant-ph]. Oct. 2022. DOI: 10.48550/arXiv.2210.06419. URL: http://arxiv.org/abs/2210.06419 (visited on 07/28/2023).

[107] Parisa Darbari, Daniel Gibney, and Sharma V. Thankachan. "Quantum Time Complexity and Algorithms for Pattern Matching on Labeled Graphs". In: *String Processing and Information Retrieval: 29th International Symposium, SPIRE 2022, Concepción, Chile, November 8–10, 2022, Proceedings*. Berlin, Heidelberg: Springer-Verlag, Nov. 2022, pp. 303–314. ISBN: 978-3-031-20642-9. DOI: 10.1007/978-3-031-20643-6_22. URL: https://doi.org/10.1007/978-3-031-20643-6_22 (visited on 07/28/2023).

[108] Bin Ma and Xiaoming Sun. "More Efficient Algorithms for Closest String and Substring Problems". In: *SIAM Journal on Computing* 39.4 (2010). _eprint: https://doi.org/10.1137/080739069, pp. 1432–1443. DOI: 10.1137/080739069. URL: https://doi.org/10.1137/080739069.

[109] Zhi-Zhong Chen and Lusheng Wang. "Fast Exact Algorithms for the Closest String and Substring Problems with Application to the Planted (L,d)-Motif Model". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8.5 (Sept. 2011). Conference Name: IEEE/ACM Transactions on Computational Biology and Bioinformatics, pp. 1400–1410. ISSN: 1557-9964. DOI: 10.1109/TCBB.2011.21. URL: https://ieeexplore.ieee.org/document/5708134 (visited on 03/14/2024).

[110] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which Problems Have Strongly Exponential Complexity?" en. In: *Journal of Computer and System Sciences* 63.4 (Dec. 2001), pp. 512–530. ISSN: 00220000. DOI: 10.1006/jcss.2001.1774. URL: https://linkinghub.elsevier.com/retrieve/pii/S002200000191774X (visited on 11/12/2025).

[111] Harry Buhrman, Subhasree Patro, and Florian Speelman. "A Framework of Quantum Strong Exponential-Time Hypotheses". en. In: *LIPIcs, Volume 187, STACS 2021* 187 (2021). Artwork Size: 19 pages, 844487 bytes ISBN: 9783959771801 Medium: application/pdf, 19:1–19:19. ISSN: 1868-8969. DOI: 10.4230/LIPICS.STACS.2021.19. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.STACS.2021.19 (visited on 05/23/2024).

[112] Pavel A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. en. The MIT Press, Aug. 2000. ISBN: 978-0-262-28152-2. DOI: 10.7551/mitpress/2022.001.0001. URL: https://direct.mit.edu/books/book/3924/Computational-Molecular-BiologyAn-Algorithmic (visited on 02/15/2024).

[113] D. Marx. "The Closest Substring problem with small distances". en. In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. Pittsburgh, PA, USA: IEEE, 2005, pp. 63–72. ISBN: 978-0-7695-2468-9. DOI: 10.1109/SFCS.2005.70. URL: http://ieeexplore.ieee.org/document/1530702/ (visited on 03/20/2024).

[114] Amir Abboud et al. *Can You Solve Closest String Faster than Exhaustive Search?* arXiv:2305.16878 [cs]. May 2023. DOI: 10.48550/arXiv.2305.16878. URL: http://arxiv.org/abs/2305.16878 (visited on 09/18/2025).

[115] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. "Quantum State Preparation with Optimal Circuit Depth: Implementations and Applications". In: *Physical Review Letters* 129.23 (Nov. 2022). arXiv:2201.11495 [quant-ph], p. 230504. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.129.230504. URL: http://arxiv.org/abs/2201.11495 (visited on 07/28/2023).

[116] Sara Ballouz, Alexander Dobin, and Jesse A. Gillis. "Is it time to change the reference genome?" In: *Genome Biology* 20.1 (Aug. 2019), p. 159. ISSN: 1474-760X. DOI: 10.1186/s13059-019-1774-4. URL: https://doi.org/10.1186/s13059-019-1774-4.

[117] Valerie A. Schneider et al. "Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly". en. In: *Genome Research* 27.5 (May 2017), pp. 849–864. ISSN: 1088-9051, 1549-5469. DOI: 10.1101/gr.213611.116. URL: http://genome.cshlp.org/lookup/doi/10.1101/gr.213611.116 (visited on 07/22/2025).

[118] Sairam Behera et al. "Comprehensive genome analysis and variant detection at scale using DRAGEN". en. In: *Nature Biotechnology* 43.7 (July 2025), pp. 1177–1191. ISSN: 1546-1696. DOI: 10.1038/s41587-024-02382-1. URL: https://www.nature.com/articles/s41587-024-02382-1 (visited on 07/22/2025).

[119] Van Hoan Do et al. "Pasa: leveraging population pangenome graph to scaffold prokaryote genome assemblies". In: *Nucleic Acids Research* 52.3 (Feb. 2024), e15. ISSN: 0305-1048. DOI: 10.1093/nar/gkad1170. URL: https://doi.org/10.1093/nar/gkad1170 (visited on 07/22/2025).

[120] Mikhail Kolmogorov et al. "Assembly of long, error-prone reads using repeat graphs". In: *Nature Biotechnology* 37.5 (May 2019), pp. 540–546. ISSN: 1546-1696. DOI: 10.1038/s41587-019-0072-8. URL: https://doi.org/10.1038/s41587-019-0072-8.

[121] Chenxi Zhou et al. *Oatk: a de novo assembly tool for complex plant organelle genomes*. Oct. 2024. DOI: https://doi.org/10.1101/2024.10.23.619857.

[122] Adam Glos, Aleksandra Krawiec, and Zoltán Zimborás. *Space-efficient binary optimization for variational computing*. en. arXiv:2009.07309 [cond-mat, physics:quant-ph]. Sept. 2020. URL: http://arxiv.org/abs/2009.07309 (visited on 05/31/2024).

[123] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. URL: https://www.gurobi.com.

[124] Iain Dunning, Swati Gupta, and John Silberholz. "What Works Best When? A Systematic Evaluation of Heuristics for Max-Cut and QUBO". en. In: *INFORMS Journal on Computing* 30.3 (Aug. 2018), pp. 608–624. ISSN: 1091-9856, 1526-5528. DOI: 10.1287/ijoc.2017.0798. URL: https://pubsonline.informs.org/doi/10.1287/ijoc.2017.0798 (visited on 06/16/2025).

[125] Gintaras Palubeckis. "Multistart Tabu Search Strategies for the Unconstrained Binary Quadratic Optimization Problem". In: *Annals of Operations Research* 131.1 (Oct. 2004), pp. 259–282. ISSN: 1572-9338. DOI: 10.1023/B:ANOR.0000039522.58036.68. URL: https://doi.org/10.1023/B:ANOR.0000039522.58036.68.

[126] Alex Daigle. *D-Wave Announces General Availability of Advantage2 Quantum Computer, Its Most Advanced and Performant System*. en-US. May 2025. URL: https://www.dwavequantum.com/company/newsroom/press-release/d-wave-announces-general-availability-of-advantage2-quantum-computer-its-most-advanced-and-performant-system/ (visited on 06/17/2025).

[127] Panagiotis Kl Barkoutsos et al. "Improving Variational Quantum Optimization using CVaR". en. In: *Quantum* 4 (Apr. 2020). arXiv:1907.04769 [quant-ph], p. 256. ISSN: 2521-327X. DOI: 10.22331/q-2020-04-20-256. URL: http://arxiv.org/abs/1907.04769 (visited on 06/16/2025).

[128] Ioannis Kolotouros and Petros Wallden. *An evolving objective function for improved variational quantum optimisation*. arXiv:2105.11766 [quant-ph]. June 2022. DOI: 10.1103/PhysRevResearch.4.023225. URL: http://arxiv.org/abs/2105.11766 (visited on 06/16/2025).

[129] Pengnian Cai et al. "Enhancing quantum approximate optimization with CNN-CVaR integration". In: *Quantum Information Processing* 24.2 (Jan. 2025), p. 37. ISSN: 1573-1332. DOI: 10.1007/s11128-025-04655-3. URL: https://doi.org/10.1007/s11128-025-04655-3.

[130] Atsushi Matsuo, Shigeru Yamashita, and Daniel J. Egger. "A SAT Approach to the Initial Mapping Problem in SWAP Gate Insertion for Commuting Gates". en. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E106.A.11 (Nov. 2023), pp. 1424–1431. ISSN: 0916-8508, 1745-1337. DOI: 10.1587/transfun.2022EAP1159. URL: https://www.jstage.jst.go.jp/article/transfun/E106.A/11/E106.A_2022EAP1159/_article (visited on 08/18/2025).

[131] Samantha V. Barron et al. "Provable bounds for noise-free expectation values computed from noisy samples". In: *Nature Computational Science* 4.11 (Nov. 2024). arXiv:2312.00733 [quant-ph], pp. 865–875. ISSN: 2662-8457. DOI: 10.1038/s43588-024-00709-1. URL: http://arxiv.org/abs/2312.00733 (visited on 11/17/2025).

[132] Elizabeth S. Meckes. *The Random Matrix Theory of the Classical Compact Groups*. Cambridge Tracts in Mathematics. Cambridge: Cambridge University Press, 2019. ISBN: 978-1-108-41952-9. DOI: 10.1017/9781108303453. URL: https://www.cambridge.org/core/product/06D446A342AACF0214BA492B49237394.

[133] Rajendra Bhatia. *Matrix Analysis*. en. Google-Books-ID: F4hRy1F1M6QC. Springer Science & Business Media, Nov. 1996. ISBN: 978-0-387-94846-1.

[134] Nathan McNew. *The Eigenvalue Gap and Mixing Time*. en. 2011.

[135] F. P. Kelly. *Reversibility and Stochastic Networks*. USA: Cambridge University Press, July 2011. ISBN: 978-1-107-40115-0.

[136] David A Levin and Yuval Peres. *Markov Chains and Mixing Times, second edition*. en. American Mathematical Society, 2006.

[137] N Alon and V. D Milman. "lambda$_1$, Isoperimetric inequalities for graphs, and superconcentrators". In: *Journal of Combinatorial Theory, Series B* 38.1 (Feb. 1985), pp. 73–88. ISSN: 0095-8956. DOI: 10.1016/0095-8956(85)90092-9. URL: https://www.sciencedirect.com/science/article/pii/0095895685900929 (visited on 04/07/2025).

[138] Michel Boyer et al. "Tight bounds on quantum searching". In: *Fortschritte der Physik* 46.4-5 (June 1998). arXiv:quant-ph/9605034, pp. 493–505. ISSN: 0015-8208, 1521-3978. DOI: 10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P. URL: http://arxiv.org/abs/quant-ph/9605034 (visited on 04/07/2025).

[139] Seyon Sivarajah et al. "t$|$ket$\rangle$: A Retargetable Compiler for NISQ Devices". In: *Quantum Science and Technology* 6.1 (Jan. 2021). arXiv:2003.10611 [quant-ph], p. 014003. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab8e92. URL: http://arxiv.org/abs/2003.10611 (visited on 10/07/2025).

[140] Colin Campbell and Edward Dahl. *QAOA of the Highest Order*. arXiv:2111.12754 [quant-ph]. Dec. 2021. DOI: 10.48550/arXiv.2111.12754. URL: http://arxiv.org/abs/2111.12754 (visited on 10/07/2025).

[141] Kristina Bell et al. *A Comparison of Quadratic and Higher-Order Representations for QAOA*. arXiv:2509.20127 [quant-ph]. Sept. 2025. DOI: 10.48550/arXiv.2509.20127. URL: http://arxiv.org/abs/2509.20127 (visited on 10/07/2025).

Appendix A

Simulating matchgate circuits

A.1 Triples of Gaussian states

Proposition A.1.1. *The manifold of solutions to the system of equations*

$$\tilde{\mathcal{F}} = \left\{ \sum_{i=2}^{4} (-1)^i a_{\overline{0}^{k_1,k_i}} a_{\overline{x}^{k_1,k_i}} = 0 : |x| = 4,\, D(0,\, x) = \{k_i : i = 1,\, \ldots,\, 4\} \right\} \tag{A.1}$$

has dimension $2n - 3$.

For convenience, define for any $x \in \mathcal{A}_n$ with $|x| = w$ and 1s in locations $i_0,\, i_1, \ldots,\, i_{w-1}$ the shorthand $x = [i_0,\, i_1, \ldots,\, i_{w-1}]$. In expressions such as $[b_0,\, b_1,\, c_0,\, c_1]$, assume $b_0 < b_1$ and $c_0 < c_1$ and the sets $\{b_0,\, b_1\}, \{c_0,\, c_1\}$ are disjoint, though their relative order is unspecified.

Proof. Suppose $a_{y^*} \neq 0$ for some $y^* = [b_0,\, b_1]$, which is required for a non-trivial linearly dependent triple. For $k = 2,\, 4,\, 6$, define $W_k = \{x \in \mathcal{A}_n : |x| = 4,\, d(x, y) = k\}$.

First, consider $x \in W_2$. Then $x = [b_0,\, b_1,\, c_0,\, c_1]$. Since $a_{y^*} = a_{[b_0,\, b_1]} \neq 0$, using the constraint from eq. (A.1) gives:

$$a_{[c_0,\, c_1]} = \frac{1}{a_{[b_0,\, b_1]}} \big(s_0^{b_0,\, b_1,\, c_0,\, c_1} a_{[b_0,\, c_0]} a_{[b_1,\, c_1]} + s_1^{b_0,\, b_1,\, c_0,\, c_1} a_{[b_0,\, c_1]} a_{[b_1,\, c_0]} \big), \tag{A.2}$$

where s_0 and s_1 are sign factors determined by the order of indices:

$$s_0 = \begin{cases} -1 & \text{if} \quad b_0 < c_0 < c_1 < b_1 \quad \text{or} \quad c_0 < b_0 < b_1 < c_1, \\ +1 & \text{otherwise,} \end{cases} \tag{A.3}$$

$$\text{and } s_1 = \begin{cases} -1 & \text{if} \quad b_1 < c_0 \quad \text{or} \quad c_1 < b_0, \\ +1 & \text{otherwise.} \end{cases} \tag{A.4}$$

Equation (A.2) fixes the $\binom{n-2}{2}$ amplitudes $a_z = a_{[c_0,\, c_1]}$ that have $d(y^*,\, z) = 4$. We now show that all remaining constraints in $\tilde{\mathcal{F}}$ are automatically satisfied, leaving $\binom{n}{2} - \binom{n-2}{2} = 2n - 3$ free variables. Thus, the manifold dimension is $2n - 3$.

Let $x \in W_4$. Then $x = [b_k,\, d_0,\, d_1,\, d_2]$ for $k = 0$ or 1. The corresponding constraint from eq. (A.1) reads

$$t_0^{b_k,d_0,d_1,d_2} a_{[b_k,\, d_0]} a_{[d_1,\, d_2]} + t_1^{b_k,d_0,d_1,d_2} a_{[b_k,\, d_1]} a_{[d_0,\, d_2]} + t_2^{b_k,d_0,d_1,d_2} a_{[b_k,\, d_2]} a_{[d_0,\, d_1]} = 0, \tag{A.5}$$

where the t_i are again index-dependent signs:

$$t_0 = \begin{cases} -1 & \text{if} \quad d_1 < b_k < d_2, \\ +1 & \text{otherwise,} \end{cases} \tag{A.6}$$

$$t_1 = \begin{cases} -1 & \text{if} \quad b_k < d_0 \quad \text{or} \quad d_2 < b_k, \\ +1 & \text{otherwise,} \end{cases} \tag{A.7}$$

$$\text{and } t_2 = \begin{cases} -1 & \text{if} \quad d_0 < b_k < d_1, \\ +1 & \text{otherwise.} \end{cases} \tag{A.8}$$

Each of $a_{[d_1,\, d_2]}, a_{[d_0,\, d_2]}$ & $a_{[d_0,\, d_1]}$ are fixed by eq. (A.2). Substituting in and multiplying out $a_{[b_0,\, b_1]}$, the constraint now reads

$$\begin{aligned} t_0^{b_k,d_0,d_1,d_2} a_{[b_k,\, d_0]} \big(& s_0^{b_0,b_1,d_1,d_2} a_{[b_0,\, d_1]} a_{[b_1,\, d_2]} + s_1^{b_0,b_1,d_1,d_2} a_{[b_0,\, d_2]} a_{[b_1,\, d_1]} \big) \\ & + t_1^{b_k,d_0,d_1,d_2} a_{[b_k,\, d_1]} \big(s_0^{b_0,b_1,d_0,d_2} a_{[b_0,\, d_0]} a_{[b_1,\, d_2]} + s_1^{b_0,b_1,d_0,d_2} a_{[b_0,\, d_2]} a_{[b_1,\, d_0]} \big) \\ & + t_2^{b_k,d_0,d_1,d_2} a_{[b_k,\, d_0]} \big(s_0^{b_0,b_1,d_0,d_1} a_{[b_0,\, d_0]} a_{[b_1,\, d_1]} + s_1^{b_0,b_1,d_0,d_1} a_{[b_0,\, d_1]} a_{[b_1,\, d_0]} \big) = 0. \end{aligned} \tag{A.9}$$

Labelling the terms of eq. (A.9) in order 1 to 6, we can pair them off as follows. If $k = 0$, pair $(1, 3)$, $(2, 5)$ & $(4, 6)$. If $k = 1$, instead pair $(1, 6)$, $(2, 4)$ & $(3, 5)$. Then it

can be shown that each pair sums to 0. For example, for $k = 0$, we have:

$$(1) + (3) \quad \propto \quad t_0^{b_0,d_0,d_1,d_2} s_0^{b_0,b_1,d_1,d_2} + t_1^{b_0,d_0,d_1,d_2} s_0^{b_0,b_1,d_0,d_2}, \tag{A.10}$$

$$(2) + (5) \quad \propto \quad t_0^{b_0,d_0,d_1,d_2} s_1^{b_0,b_1,d_1,d_2} + t_2^{b_0,d_0,d_1,d_2} s_0^{b_0,b_1,d_0,d_1}, \tag{A.11}$$

$$(4) + (6) \quad \propto \quad t_1^{b_0,d_0,d_1,d_2} s_1^{b_0,b_1,d_0,d_2} + t_2^{b_0,d_0,d_1,d_2} s_1^{b_0,b_1,d_0,d_1}. \tag{A.12}$$

t_0 and t_1 have the same sign only if $d_0 < b_0 < d_1$. In this case, $s_0^{b_0,b_1,d_1,d_2}$ and $s_0^{b_0,b_1,d_0,d_2}$ have different signs, as can be seen from checking the three possible relative locations of b_1. Thus $(1) + (3) = 0$. The remaining cases can all be checked similarly.

A similar argument holds for $y = [d_0,\, d_1,\, d_2,\, d_3] \in W_6$. □

A.2 An ϵ-net for the Gaussian states

Recall that we are seeking a finite set $\mathcal{N} \subset G_n$ such that for every $|\psi\rangle \in G_n$ there exists $|\tilde{\psi}\rangle \in \mathcal{N}$ with $\left\| |\psi\rangle - |\tilde{\psi}\rangle \right\|_1 < \epsilon$. For each $j \in \mathcal{A}_n$, define the regions

$$S_j \coloneqq \left\{ |\psi\rangle = \sum_{j' \in \mathcal{A}_n} c_{j'} |j'\rangle \in G_n : |c_j| \geq |c_{j'}| \quad \forall\, j' \in \mathcal{A}_n \right\}. \tag{A.13}$$

The union $\bigcup_j S_j$ covers G_n.

In each S_j, Gaussian states are defined by amplitudes with labels within Hamming distance 2 of j, given by the set

$$C_j = \{c_k : d_H(j,k) \leq 2\}, \tag{A.14}$$

with

$$|c_j| \in [2^{\frac{1-n}{2}},\, 1], \tag{A.15}$$

$$|c_k| \in [0,\, |c_j|] \quad \forall\, c_k \in C \setminus \{c_j\}. \tag{A.16}$$

Let $\mathcal{N}_j = \{\, |\tilde{\psi}_i\rangle\} \subset S_j$ be a maximal set of states such that for any pair $|\tilde{\psi}_i\rangle\,, |\tilde{\psi}_{i'}\rangle \in \mathcal{N}_j$,

$$\exists\, x \in \mathcal{A}_n \text{ with } d_H(j,x) \leq 2 \text{ and } |\, \langle x|\tilde{\psi}_i\rangle - \langle x|\tilde{\psi}_{i'}\rangle| \geq \eta. \tag{A.17}$$

Let $\mathcal{N} = \bigcup_j \mathcal{N}_j$. To show that $\mathcal{N}$ forms an ϵ-net, we prove two lemmas:

Lemma A.2.1. *For $\eta = 2^{-n^2}\epsilon$, we have $\sup_{|\psi\rangle \in S_j} \inf_{|\tilde{\psi}\rangle \in \mathcal{N}_j} \left\| |\psi\rangle - |\tilde{\psi}\rangle \right\|_1 \leq \epsilon \;\forall\, j \in \mathcal{A}_n$.*

Lemma A.2.2. *The cardinality of $\mathcal{N}_j$ is upper bounded by $|\mathcal{N}_j| \leq 2^{3n^2+1}\eta^{-n^2}$.*

Combining both results yields the following proposition, corresponding to proposition 3.1.3 in the main text.

Proposition 3.1.3. *Let $\epsilon = 2^{-l}$. Then there is an ϵ-net $\mathcal{N}$ with cardinality $\log_2 |\mathcal{N}| \leq n^4 + 3n^2 + n + ln^2$.*

Proof. By lemma A.2.1, we have

$$\sup_{|\psi\rangle \in G} \inf_{|\tilde{\psi}\rangle \in \mathcal{N}} \left\| |\psi\rangle - |\tilde{\psi}\rangle \right\|_1 \leq \sup_{j \in \mathcal{A}_n} \sup_{|\psi\rangle \in S_j} \inf_{|\tilde{\psi}\rangle \in \mathcal{N}_j} \left\| |\psi\rangle - |\tilde{\psi}\rangle \right\|_1 \leq \epsilon, \tag{A.18}$$

so $\mathcal{N}$ is a valid ϵ-net.

By lemma A.2.2 with $\eta = 2^{-n^2}\epsilon$, we have

$$|\mathcal{N}| = \sum_j |\mathcal{N}_j| \leq 2^{n-1} \cdot 2^{3n^2+1} \left(2^{-n^2}\epsilon\right)^{-n^2} \tag{A.19}$$

$$= 2^{n^4+3n^2+n} \epsilon^{-n^2}. \tag{A.20}$$

Let $\epsilon = 2^{-l}$. Then $\log_2 |\mathcal{N}| \leq n^4 + 3n^2 + n + ln^2$. □

Proof of lemma A.2.2. Without loss of generality, consider the region S_0 and sub-net $\mathcal{N}_0$.

We use a volume argument to estimate the number of states in the sub-net. Note that each state $|\tilde{\psi}_i\rangle = \sum_{x \in \mathcal{A}_n} \tilde{c}_x |x\rangle \in \mathcal{N}_j$ occupies a disjoint region of $\binom{n}{2}$-dimensional (complex) space with volume $\left(\pi(\frac{\eta}{2})^2\right)^{\binom{n}{2}}$.

The centre of each of these disjoint regions lies in S_0. The volume of the total region that they are contained in, V, can be estimated by integrating over each free component $\tilde{c}_x$ with $|x| \leq 2$.

Viewed in $\mathbb{R}^2$, $\tilde{c}_0$ lies in an annulus with inner radius $2^{\frac{1-n}{2}}$ and outer radius 1. The inner radius comes from the requirement that $\tilde{c}_0$ is the largest component, and the outer radius comes from the normalisation condition. Each $\tilde{c}_x$ with $|x| = 2$ certainly lies in a disk of radius $|\tilde{c}_0|$. The total region that bounds the disjoint regions surrounding each state extends at most $\eta/2$ from each of these radii. Thus, we see that the volume

V is no greater than

$$V = 2\pi \int_{2^{\frac{1-n}{2}}-\frac{\eta}{2}}^{1+\frac{\eta}{2}} dr_0 \left\{ \left(\pi (r_0 + \frac{\eta}{2})^2 \right)^{\frac{n(n-1)}{2}} \right\} \tag{A.21}$$

$$= 2\pi^{\frac{n(n-1)}{2}+1} \left[\frac{(r_0 + \frac{\eta}{2})^{n(n-1)+1}}{n(n-1)+1} \right]_{2^{\frac{1-n}{2}}-\frac{\eta}{2}}^{1+\frac{\eta}{2}} \tag{A.22}$$

$$= 2\pi^{\frac{n(n-1)}{2}+1} \left(\frac{(1+\eta)^{n(n-1)+1}}{n(n-1)+1} - \frac{(2^{\frac{1-n}{2}})^{n(n-1)+1}}{n(n-1)+1} \right) \tag{A.23}$$

$$\leq 2 \cdot \pi^{\frac{n^2}{2}} \cdot 2^{n(n-1)+1} \tag{A.24}$$

$$\leq 2 \cdot 2^{n^2} \cdot 2^{n^2-n} = 2^{2n^2-n+1}, \tag{A.25}$$

where eq. (A.24) holds for any $\eta \leq 1$.

The number of states in any $\mathcal{N}_j$ is then no more than

$$|\mathcal{N}_j| \leq \frac{V}{\left(\pi(\eta/2)^2\right)^{\binom{n}{2}}} \tag{A.26}$$

$$\leq 2^{2n^2-n+1} \cdot (4/\pi)^{n(n-1)/2} \cdot \eta^{-n(n-1)} \tag{A.27}$$

$$\leq 2^{2n^2-n+1} \cdot 2^{n(n-1)/2} \cdot \eta^{-n(n-1)} \tag{A.28}$$

$$\leq 2^{\frac{5}{2}n^2-\frac{3}{2}n+1} \eta^{-n^2} \tag{A.29}$$

$$\leq 2^{3n^2+1} \eta^{-n^2}. \qquad \square$$

Proof of lemma A.2.1. We will frequently use the identity

$$|AB - A'B'| \leq |A - A'||B| + |B - B'||A'|, \tag{A.30}$$

Again, we focus on S_0 without loss of generality. For any given state $|\psi\rangle \in S_0$, we wish to find a $|\tilde{\psi}\rangle \in \mathcal{N}_0$ that is close to $|\psi\rangle$ and bound the distance.

Let $|\psi\rangle = \sum_{x \in \mathcal{A}_n} c_x |x\rangle$ and $|\tilde{\psi}\rangle = \sum_{x \in \mathcal{A}_n} \tilde{c}_x |x\rangle$. By the definition of $\mathcal{N}_0$ given in eq. (A.17), there exists a choice of $|\tilde{\psi}\rangle$ such that:

$$\delta_x \coloneqq |c_x - \tilde{c_x}| < \eta, \quad \forall\ |x| \leq 2. \tag{A.31}$$

Otherwise, $|\psi\rangle$ would be in $\mathcal{N}_0$, since $\mathcal{N}_0$ was chosen to be maximal.

Let $\Delta_0 = \Delta_2 \coloneqq \eta$ serve as the upper bound for δ_x for all $|x| \leq 2$. We seek upper bounds $\Delta_4, \ldots, \Delta_n$ for δ_x for $|x| = 4, \ldots, n$. We proceed by induction.

Suppose we have bounded $\delta_x \leq \Delta_{w'}$ for all $|x| = w'$, $w' = 2, 4, \ldots, w-2$. For any $|x| = w$, we have a constraint $c_0 c_x = f_x$, where f_x consists of an alternating sum of $(w-1)$ terms of the form $c_y c_{y'}$ with $|y| = 2$ and $|y'| = w-2$. We have a similar constraint $\tilde{c}_0 \tilde{c}_x = \tilde{f}_x$ for the amplitudes of $|\tilde{\psi}\rangle$. We may then bound δ_x:

$$\begin{aligned}
\delta_x &= |c_x - \tilde{c}_x| && \text{(A.32)}\\
&= \left| \frac{f_x}{c_0} - \frac{\tilde{f}_x}{\tilde{c}_0} \right| && \text{(A.33)}\\
&\leq |c_0|^{-1} |\tilde{c}_0|^{-1} \left(|\tilde{c}_0 - c_0||f_x| + |\tilde{c}_0||f_x - \tilde{f}_x| \right), && \text{(A.34)}
\end{aligned}$$

where the last line follows from eq. (A.30). We can bound each term in this expression:

$$\begin{aligned}
|c_0|^{-1}, |\tilde{c}_0|^{-1} &\leq 2^{\frac{n-1}{2}}, && \text{(A.35)}\\
|\tilde{c}_0 - c_0| = \delta_0 &\leq \Delta_0 = \Delta_2, && \text{(A.36)}\\
|f_x| = |c_0||c_x| &\leq 1, && \text{(A.37)}\\
|\tilde{c}_0| &\leq 1, && \text{(A.38)}\\
|f_x - \tilde{f}_x| &\leq \sum_{i=1}^{w-1} |c_{y_i} c_{y'_i} - \tilde{c}_{y_i} \tilde{c}_{y'_i}| && \text{(A.39)}\\
&\leq (w-1)(\Delta_2 + \Delta_{w-2}), && \text{(A.40)}
\end{aligned}$$

where eq. (A.39) follows from the triangle equality and eq. (A.40) from eq. (A.30). Substituting in to eq. (A.34), we obtain:

$$\delta_x \leq 2^{n-1}(\Delta_2 + (w-1)(\Delta_2 + \Delta_{w-2})) \qquad \text{(A.41)}$$

Since this is true for any x with $|x| = w$, and $\Delta_2 \leq \Delta_{w-2}$, we may choose

$$\Delta_w = 2^{n-1}(2w-1)\Delta_{w-2}. \qquad \text{(A.42)}$$

The solution to this recurrence relation with initial condition $\Delta_2 = \eta$ is

$$\Delta_w = \frac{1}{3} 2^{\frac{1}{2}(n+2)w - n - 1} \frac{\Gamma(\frac{w}{2} + \frac{3}{4})}{\Gamma(\frac{3}{4})} \eta. \qquad \text{(A.43)}$$

We then have

$$\left\| |\psi\rangle - |\tilde{\psi}\rangle \right\|_1 = \sum_{x \in \mathcal{A}_n} |c_x - \tilde{c}_x| \tag{A.44}$$

$$\leq \sum_{w=0}^{\frac{n}{2}} \binom{n}{2w} \Delta_{2w} \tag{A.45}$$

$$\leq 2^{n-1} \Delta_n \tag{A.46}$$

$$\leq \frac{1}{3} \cdot 2^{\frac{1}{2}n^2+n-2} \cdot \frac{\Gamma(\frac{n}{2}+\frac{3}{4})}{\Gamma(\frac{3}{4})} \eta. \tag{A.47}$$

Using $\Gamma(z+1) = z\Gamma(z)$, we may obtain

$$\frac{\Gamma(\frac{n}{2}+\frac{3}{4})}{\Gamma(\frac{3}{4})} \leq \left(\frac{n}{2}\right)^{\frac{n}{2}} \leq 2^{\frac{n}{2}(\log_2(n)-1)}. \tag{A.48}$$

Substituting into eq. (A.47), we finally have

$$\left\| |\psi\rangle - |\tilde{\psi}\rangle \right\|_1 \leq \frac{1}{3} 2^{\frac{1}{2}(n^2+n\log_2(n)+n-2)} \eta \leq 2^{n^2} \eta. \tag{A.49}$$

□

A.3 Symmetry-restricted Gaussian rank of two copies of a magic state

Recall the statement of the proposition:

Proposition 3.1.5. *The symmetry-restricted Gaussian rank of* $|M\rangle^2$ *is* $\tilde{\chi}_G(|M\rangle^2) = 4$.

We give a sketch proof of proposition 3.1.5. In particular, we seek a symmetry-restricted decomposition with three symmetry-restricted Gaussian terms:

$$|M\rangle^2 = \frac{1}{2}(|0\rangle + |15\rangle + |240\rangle + |255\rangle) \equiv \sum_{i=1}^{3} \sum_x a_x^i |x\rangle = \sum_{i=1}^{3} c_i |\psi_i\rangle , \tag{A.50}$$

where the $|x\rangle$ are invariant under Z_3Z_4 and Z_7Z_8 and the $|\psi_i\rangle$ are Gaussian states. In the body text, we discussed how this symmetry reduces the set of Gaussian constraints to a more manageable set. In particular, all but two of the constraints are reduced to a single pair of terms, and we can represent them using a grid. For convenience, we reproduce that grid here in table A.1.

We consider several cases and find a contradiction in each.

Table A.1 Constraints on an 8-qubit, symmetry-restricted Gaussian state, assuming $a_0 \neq 0$, are determined by examining a component within the body and multiplying it by the upper-leftmost entry, which equals the product of its row and column headers. For example, $a_{15}a_0 = a_3a_{12}$, $a_{90}a_0 = a_{66}a_{24}$, etc.

0	3	65	66	129	130	192	195
12	15	77	78	141	142	204	207
20	23	85	86	149	150	212	215
24	27	89	90	153	154	216	219
36	39	101	102	165	166	228	231
40	43	105	106	169	170	232	235
48	51	113	114	177	178	240	243
60	63	125	126	189	190	252	255

A.3.1 $a_0^i \neq 0$, $i = 1, 2, 3$

The simplest case is when all three states have a non-vanishing coefficient a_0. In this case, table A.1 lists all the two-term constraints without any complications.

As an illustrative example, consider the coefficients a_{15}^3 and a_{51}^3. Note that in binary, 15 and 51 have weight four, so two constraints are acting on each of these coefficients. The former comes from $|\psi_3\rangle$ being a Gaussian state; the second from the decomposition in eq. (A.50). These are shown in eqs. (A.51) and (A.53). Within these constraints, we can further replace (almost) all a_x^3 with expressions in terms of a_x^1 and a_x^2 using eq. (A.50). We then replace any a_x^i where $|x| \geq 4$ with an expression in terms of a_y^i, where $|y| \leq 2$, using the Gaussian state constraints. The final expressions are shown in eqs. (A.52) and (A.54).

$$a_{15}^3 = (\frac{1}{2} - a_{15}^1 - a_{15}^2) = \frac{a_{12}^3 a_3^3}{a_0^3} \tag{A.51}$$

$$\implies \frac{1}{2}a_0^3 = \frac{a_3^1}{a_0^1}\left(\frac{1}{2}a_{12}^1 - a_{12}^1 a_0^2 + a_{12}^2 a_0^1\right) + \frac{a_3^2}{a_0^2}\left(\frac{1}{2}a_{12}^2 - a_{12}^2 a_0^1 + a_{12}^1 a_0^2\right), \tag{A.52}$$

$$a_{51}^3 = -(a_{51}^1 + a_{51}^2) = \frac{a_{48}^3 a_3^3}{a_0^3} \tag{A.53}$$

$$\implies 0 = \frac{a_3^1}{a_0^1}\left(\frac{1}{2}a_{48}^1 - a_{48}^1 a_0^2 + a_{48}^2 a_0^1\right) + \frac{a_3^2}{a_0^2}\left(\frac{1}{2}a_{48}^2 - a_{48}^2 a_0^1 + a_{48}^1 a_0^2\right). \tag{A.54}$$

Equations (A.52) and (A.54) are structurally very similar. For notational simplicity, we define a symbol which captures the terms in brackets:

$$f_j^i \coloneqq \frac{1}{2}a_j^i + (-1)^i(a_0^1 a_j^1 - a_0^2 a_j^0). \tag{A.55}$$

Then proceeding similarly for each other x with $|x| \geq 4$, we obtain the matrix equation:

$$\begin{pmatrix} A \\ A' \end{pmatrix} \begin{pmatrix} X & X' \end{pmatrix} = \begin{pmatrix} \frac{1}{2}a_0^3 I_3 & 0 \\ 0 & 0 \end{pmatrix}, \tag{A.56}$$

where

$$A = \begin{pmatrix} f_{12}^0 & f_{12}^1 \\ f_{48}^0 & f_{48}^1 \\ f_{60}^0 & f_{60}^1 \end{pmatrix}, \qquad A' = \begin{pmatrix} f_{20}^0 & f_{20}^1 \\ f_{24}^0 & f_{24}^1 \\ f_{36}^0 & f_{36}^1 \\ f_{40}^0 & f_{40}^1 \end{pmatrix}, \tag{A.57}$$

$$X = \begin{pmatrix} a_3^1/a_0^1 & a_{192}^1/a_0^1 & a_{195}^1/a_0^1 \\ a_3^2/a_0^2 & a_{192}^2/a_0^2 & a_{195}^2/a_0^2 \end{pmatrix}, \; X' = \begin{pmatrix} a_{65}^1/a_0^1 & a_{66}^1/a_0^1 & a_{129}^1/a_0^1 & a_{130}^1/a_0^1 \\ a_{65}^2/a_0^2 & a_{66}^2/a_0^2 & a_{129}^2/a_0^2 & a_{130}^2/a_0^2 \end{pmatrix}. \tag{A.58}$$

Note that $\mathrm{rank}(AX) \leq \min(\mathrm{rank}(A), \mathrm{rank}(X)) \leq 2$, whereas $\mathrm{rank}(a_0^3 I_3) = 3$ since, by assumption, $a_0^3 \neq 0$. Thus, there is no solution to eq. (A.56), and this case leads to a contradiction.

A.3.2 $a_0^3 = 0,\ a_0^i \neq 0,\ i = 1, 2$

Recall that when $a_0^3 = 0$, the conditions for the state to be Gaussian require several other coefficients also to vanish: either the first column or the first row of table A.1. By symmetry, without loss of generality, choose the first column. In other words, we set the following coefficients to 0:

$$a_{12}^3 = a_{20}^3 = a_{24}^3 = a_{36}^3 = a_{40}^3 = a_{48}^3 = a_{60}^3 = 0. \tag{A.59}$$

Then note that, by eq. (A.50),

$$a_j^2 = -a_j^1 \text{ for } j = 12,\ 20,\ 24,\ 36,\ 40,\ 48,\ 60. \tag{A.60}$$

Finally, recall that there are a pair of constraints not captured by table A.1, given by eq. (3.51). Applying the relevant constraint, we obtain:

$$a^1_{60} = \frac{a^1_{12}a^1_{48} - a^1_{20}a^1_{40} + a^1_{24}a^1_{36}}{a^1_0} = -\frac{a^2_{12}a^2_{48} - a^2_{20}a^2_{40} + a^2_{24}a^2_{36}}{a^2_0} = -a^2_{60}. \tag{A.61}$$

By eq. (A.60), either $a^2_0 = -a^1_0$ or $a^1_{60} = a^2_{60} = 0$.

In the first case, we immediately have $\sum_i a^i_0 = 0$, which is not possible by eq. (A.50). In the second case, since a^1_0, $a^2_0 \neq 0$, we have

$$a^1_j = a^2_j = 0 \text{ for } j = 63, 125, 126, 189, 190, 252, 255 \;\; (\text{since } a^1_j \propto a^1_{60}, a^2_j \propto a^2_{60}), \tag{A.62}$$

$$\implies a^3_j = 0 \text{ for } j = 63, 125, 126, 189, 190, 252 \;\; (\text{since } \sum_i a^i_j = 0). \tag{A.63}$$

Consider a^3_{255}. Since $a^1_{255} = a^2_{255} = 0$ we must have $a^3_{255} \neq 0$. Since the rest of the row containing 255 is 0 for $|\psi_3\rangle$, it must be the case that only the last column is non-zero for $|\psi_3\rangle$. In particular, $a^3_{240} = 0$ and $a^3_{15} = 0$.

Finally, by eq. (A.60) ,

$$\frac{1}{2} = \sum_i a^i_{15} = \frac{a^1_{12}a^1_3}{a^1_0} + \frac{a^2_{12}a^2_3}{a^2_0} + 0 = a^1_{12}\left(\frac{a^1_3}{a^1_0} - \frac{a^2_3}{a^2_0}\right), \tag{A.64}$$

$$0 = \sum_i a^i_{51} = \frac{a^1_{48}a^1_3}{a^1_0} + \frac{a^2_{48}a^2_3}{a^2_0} + 0 = a^1_{48}\left(\frac{a^1_3}{a^1_0} - \frac{a^2_3}{a^2_0}\right). \tag{A.65}$$

Clearly we must have $a^1_{48} = a^2_{48} = 0$. But then $a^1_{240} = a^2_{240} = 0$ by the Gaussian constraints in table A.1. So all $a^i_{240} = 0$, contradicting eq. (A.50).

A.3.3 $a^2_0 = a^3_0 = 0,\ a^1_0 \neq 0$

This case spawns several sub-cases depending on which column of table A.1 we delete for each of $|\psi_2\rangle$ and $|\psi_3\rangle$. In each case, the reasoning follows much as the above, and we exclude the details.

Appendix B

Learning Gaussian operations

B.1 Bounding the error of learning signs of Q

We wish to obtain upper bounds on $\mathbb{P}(F < \kappa)$ for $F \in \{|Q_{2l-1,1}Q_{2l,k}|, |Q_{2l,1}\,Q_{2l-1,k}|, |Q_{2l-1,1}\,Q_{2l,k} - Q_{2l,1}\,Q_{2l-1,k}|\}$, for any $l \in [n]$, $k \in [2n] \setminus \{1\}$, where the probability is over the Haar distribution on the orthogonal matrices. We will fix $l = 1$, $k = 2$ and use a union bound to obtain the full result.

Since Q is orthogonal, the first and second rows of Q are random orthogonal unit vectors. We may obtain them by performing Gram-Schmidt orthonormalisation on two random points on the unit sphere. Let the two random points be given by the vectors $\mathbf{v_1}$ and $\mathbf{v_2}$. We choose to parametrise them in polar coordinates by

$$\mathbf{v_i} = \big(\sin\big(\theta_1^i\big),\, \cos\big(\theta_1^i\big)\cos\big(\theta_2^i\big),\, \ldots\big), \tag{B.1}$$

where $\theta_1^i \sim U[0,\, 2\pi)$ and $\theta_2^i \sim U[0,\, \pi)$, and all angles are distributed independently.

We first consider $F_1 \coloneqq |Q_{11}Q_{22}|$, with $F_2 \coloneqq |Q_{21}Q_{12}|$ following similarly. We have

$$\begin{aligned}
\mathbb{P}(F_1 < \kappa) &= \mathbb{P}\left(\left|\sin\big(\theta_1^1\big)\cos\big(\theta_1^2\big)\cos\big(\theta_2^2\big)\right| < \kappa\right) && \text{(B.2)}\\
&\leq \mathbb{P}\left(\min\left(\left|\sin\big(\theta_1^1\big),\, \cos\big(\theta_1^2\big),\, \cos\big(\theta_2^2\big)\right|\right) < \kappa^{1/3}\right) && \text{(B.3)}\\
&\leq \mathbb{P}\left(\left|\sin\big(\theta_1^1\big)\right| < \kappa^{1/3}\right) + \mathbb{P}\left(\left|\cos\big(\theta_1^2\big)\right| < \kappa^{1/3}\right) + \mathbb{P}\left(\left|\cos\big(\theta_2^2\big)\right| < \kappa^{1/3}\right) && \text{(B.4)}\\
&= \mathcal{O}(\kappa^{1/3}), && \text{(B.5)}
\end{aligned}$$

to leading order.

We now consider $F_3 \coloneqq |Q_{11}Q_{22} - Q_{21}Q_{12}| = \left|\det\big(Q|_{\{1,2\},\{1,2\}}\big)\right|$. To begin, we consider the probability of this term being small if we did not perform the Gram-Schmidt procedure, denoting the corresponding quantity $\tilde{F}_3$. Since the determinant of

a 2-by-2 matrix is the area of the parallelogram defined by the rows of the matrix, we can express $\tilde{F}_3$ as

$$\tilde{F}_3 = \|\Pi(\mathbf{v_1})\|\|\Pi(\mathbf{v_2})\| \sin(\phi), \tag{B.6}$$

where Π is the orthogonal projection onto the first two coordinates and ϕ is the angle between the projected vectors in the plane. Noting that $\|\Pi(\mathbf{v_i})\| = |\cos(\theta_2^i)|$ and $\phi \sim U[0, 2\pi)$, and using the shorthand $c(\cdot) = \cos(\cdot)$ and $s(\cdot) = \sin(\cdot)$, we may compute

$$\begin{aligned}
\mathbb{P}(\tilde{F}_3 < \kappa) &= \mathbb{P}\left(\left|c(\theta_2^1)c(\theta_2^i)s(\phi)\right| < \kappa\right) && \text{(B.7)}\\
&= \mathbb{P}\left(|c(\theta_2^1)c(\theta_2^i)s(\phi)| < \kappa \;\middle|\; |c(\theta_2^1)c(\theta_2^i)| < \zeta\right) \cdot \mathbb{P}\left(\left|c(\theta_2^1)c(\theta_2^i)\right| < \zeta\right) \\
&\quad + \mathbb{P}\left(|c(\theta_2^1)c(\theta_2^i)s(\phi)| < \kappa \;\middle|\; |c(\theta_2^1)c(\theta_2^i)| \geq \zeta\right) \cdot \mathbb{P}\left(\left|c(\theta_2^1)c(\theta_2^i)\right| \geq \zeta\right) && \text{(B.8)}\\
&\leq \mathbb{P}\left(\left|c(\theta_2^1)c(\theta_2^i)\right| < \zeta\right) + \mathbb{P}\left(|\zeta s(\phi)| < \kappa\right) && \text{(B.9)}\\
&= \mathcal{O}\left(\sqrt{\zeta}\right) + \mathcal{O}(\kappa/\zeta) && \text{(B.10)}
\end{aligned}$$

to leading order. Optimising over ζ, we obtain the bound $\mathbb{P}(\tilde{F}_3 < \kappa) \leq \mathcal{O}(\kappa^{1/3})$.

We now consider the effect of orthonormalisation of $\mathbf{v_2}$ on the quantity F_3. Let the resulting vector be $\tilde{\mathbf{v}}_\mathbf{2} = \lambda \mathbf{v_2} + \mu \mathbf{v_1}$, where $\lambda := (1 - (\mathbf{v_1} \cdot \mathbf{v_2})^2)^{-0.5}$. The corresponding determinant in the plane of the first two coordinates is $F_3 = \lambda \tilde{F}_3$, since the $\mathbf{v_1}$ part of $\tilde{\mathbf{v}}_\mathbf{2}$ does not contribute. Since $\lambda \in [1, \infty)$, we have

$$\mathbb{P}(F_3 < \kappa) \leq \mathbb{P}(\tilde{F}_3 < \kappa) = \mathcal{O}(\kappa^{1/3}). \tag{B.11}$$

B.2 Bounding the distance between M_Q and $M_{Q'}$

Given a matrix Q' which approximates Q up to term-wise additive error η, we wish to bound the "distance" $D(M_Q, M_{Q'})$ between the corresponding unitary operators M_Q and $M_{Q'}$, as defined by Equations (1.13) and (1.14). We first consider the error in the anti-symmetric matrix h that generates Q via $Q = \exp(4h)$. Let $h' = h + \frac{1}{4}E$, where E is the error term whose norm we will seek to bound.

We use the series expansion of the matrix logarithm:

$$E = 4(h' - h) = \log(Q + \eta) - \log(Q) = \int_0^\infty dz \left\{ \frac{I}{Q + zI} \eta \frac{I}{Q + zI} \right\} + \mathcal{O}(\|\eta\|_2^2). \tag{B.12}$$

Therefore, to leading order,

$$\|E\|_2 \leq \int_0^\infty dz \left\{ \left\| \frac{I}{Q + zI} \right\|_2 \|\eta\|_2 \left\| \frac{I}{Q + zI} \right\|_2 \right\}. \tag{B.13}$$

We recall that, for any matrix A,

$$\|A\|_2 = \sqrt{\sum_i \sigma_i^2(A)} = \sqrt{\sum_i \sigma_i^{-2}(A^{-1})}, \tag{B.14}$$

where the $\sigma_i(A)$ are the singular values of A.

Letting the eigenangles of Q be $\{\theta_i\}_{i=1}^{2n}$, it is easy to check that $\sigma_i(Q + zI) = \sqrt{1 + 2\cos(\theta_i)z + z^2}$. Then Equation (B.13) becomes

$$\|E\|_2 \leq \|\eta\|_2 \int_0^\infty dz \left\{ \sum_i \frac{1}{1 + 2\cos(\theta_i)z + z^2} \right\} \tag{B.15}$$

$$= \|\eta\|_2 \left(\sum_i \left| \frac{\theta_i}{\sin(\theta_i)} \right| \right). \tag{B.16}$$

Since the eigenangles of a Haar-random orthogonal matrix in $2n$ dimensions are uniformly distributed (in complex conjugate pairs) on the unit disc [132], we may bound the probability of the term in brackets being overly large:

$$\mathbb{P}\left(\sum_i \left| \frac{\theta_i}{\sin(\theta_i)} \right| \geq \kappa \right) \leq \mathbb{P}\left(\bigcup_i \left| \frac{\theta_i}{\sin(\theta_i)} \right| \geq \frac{\kappa}{2n} \right) \tag{B.17}$$

$$\leq 2n\mathbb{P}\left(\left| \frac{\theta}{\sin(\theta)} \right| \geq \frac{\kappa}{2n} \right) \tag{B.18}$$

$$\leq 4n\mathbb{P}\left(\theta \in \left[\pi - \frac{2n\pi}{\kappa}, \pi \right) \right) \tag{B.19}$$

$$= \frac{8n^2}{\kappa}. \tag{B.20}$$

Taking $\kappa = \Omega(n^2)$ then implies $\sum_i \left| \frac{\theta_i}{\sin(\theta_i)} \right| < \kappa$ with high probability.

We may also bound $\|\eta\|_2$ using the entry-wise expression for the 2-norm:

$$\|\eta\|_2 = \sqrt{\sum_{\mu\nu} |\eta_{\mu\nu}|^2} \leq \sqrt{4n^2 \cdot \eta^2} = 2n\eta. \tag{B.21}$$

Therefore, $\|E\|_2 = \mathcal{O}(n^3\eta)$ with high probability.

We are now prepared to compute the desired distance,

$$D(M_Q, M_{Q'}) = \sqrt{1 - \left| d^{-1} \operatorname{Tr}\left(M_Q^\dagger M_{Q'} \right) \right|^2}.$$

Using $\mathrm{Tr}(\exp(A+B)) \leq \mathrm{Tr}(\exp(A)\exp(B))$ for Hermitian matrices A, B (see, e.g., [133]), we find:

$$\mathrm{Tr}\left(M_Q^\dagger M_{Q'}\right) = \mathrm{Tr}(\exp(-iH)\exp(iH')) \tag{B.22}$$

$$\geq \mathrm{Tr}(\exp(-iH+iH')) \tag{B.23}$$

$$= \mathrm{Tr}\left(\exp\left(-\frac{1}{4}\sum_{\mu\nu} E_{\mu\nu}\gamma_\mu\gamma_\nu\right)\right) \tag{B.24}$$

$$= \mathrm{Tr}\left(I - \frac{1}{4}\sum_{\mu\nu} E_{\mu\nu}\gamma_\mu\gamma_\nu + \frac{1}{32}\sum_{\mu\nu\sigma\tau} E_{\mu\nu}E_{\sigma\tau}\gamma_\mu\gamma_\nu\gamma_\sigma\gamma_\tau + \mathcal{O}(\|E\|_2^3))\right) \tag{B.25}$$

$$= d\left(1 - \frac{1}{32}\sum_{\mu\nu} E_{\mu\nu}^2 + \mathcal{O}(\|E\|_2^3)\right), \tag{B.26}$$

recalling that $\mathrm{Tr}(\gamma_S) = 0$ for $S \neq \emptyset$, and that h, h' and therefore E are anti-symmetric. Finally, to leading order we have:

$$D(M_Q,\, M_{Q'}) = \sqrt{1 - \left|1 - \frac{1}{32}\sum_{\mu\nu} E_{\mu\nu}^2\right|^2} \tag{B.27}$$

$$\leq \sqrt{\frac{1}{16}\left|\sum_{\mu\nu} E_{\mu\nu}^2\right|} \tag{B.28}$$

$$\leq \frac{1}{4}\sqrt{\sum_{\mu\nu} |E_{\mu\nu}|^2} \tag{B.29}$$

$$= \frac{1}{4}\|E\|_2 = \mathcal{O}(n^3\eta). \tag{B.30}$$

Appendix C

Quantum search via random walks

Here, we provide a full background on the quantum search via random walk method from Magniez et al. [47] and how it applies to our search algorithm. We first give a brief introduction to the theory of Markov chains in appendix C.1; for a complete treatment, see, e.g., [134]. We then discuss the functioning of the search algorithm in appendix C.2. Finally, in appendix C.3, we derive the relevant properties of the symmetric Markov chain that we describe in the main text.

C.1 Markov chains, eigenvalue gaps and congestion

A Markov chain is a stochastic process in which the probability of each event depends only on the state attained after the previous event. The chain has a state space Ω consisting of all the states the chain can be in. We consider discrete time chains, which can be described by a sequence of random variables $X_0, X_1, X_2, \ldots$ taking values in Ω.

We identify a chain with its transition matrix $P = (p_{xy})_{x,y\in\Omega}$, where p_{xy} is the probability of a transition from x to y. We must have $p_{xy} \geq 0$ and $\sum_{y\in\Omega} p_{xy} = 1$ to have the correct probabilistic interpretation. The eigenvalues of P therefore have magnitude at most 1. It is often helpful to associate a Markov chain with a random walk on an edge-weighted, directed graph $G = (\Omega, E)$ with weight on edge $x \to y$ given by p_{xy}.

The *mixing time* of a Markov chain is the minimum time t_{mix} such that the time-t_{mix} distribution is "close" to the stationary distribution π. The mixing time plays a central role in the time complexity of random walk search. For example, consider algorithm 1 with $t_1 \approx t_{\text{mix}}$. Intuitively, leaving the chain for t_{mix} time steps yields a well-mixed distribution close to π. Repeating for t_2 steps represents repeated sampling from π. If we have a good bound on the probability under π of sampling a marked element, we

can use a Chernoff bound to argue that algorithm 1 succeeds with high probability. To bound the mixing time of chains, we will need several definitions.

A chain is said to be *irreducible* if every state is reachable from every other state. A state x has *period* given by $\gcd\{n > 0 : \mathbb{P}(X_n = x \mid X_0 = x) > 0\}$. A state is *aperiodic* if it has period 1, and an irreducible chain is *aperiodic* if any state is. An irreducible, aperiodic chain is *ergodic*. A state x is *positive recurrent* if, starting from x at time 0, the chain will return to x in finite time with probability 1. An irreducible chain is *positive recurrent* if any state is.

A *stationary distribution* $\pi = (\pi_x)$ of a Markov chain P is a left eigenvector of P with eigenvalue 1 and with entries satisfying $\pi_x \geq 0$ and $\sum_x \pi_x = 1$. Ergodic chains always have a unique stationary distribution. Moreover, all other eigenvalues have magnitude strictly less than 1.

For an ergodic chain P, the time-reversed Markov chain $P^* = (p^*_{xy})$ of P is defined by the equations $\pi_x p_{xy} = \pi_y p^*_{yx}$. P is said to be *reversible* if $P^* = P$. A reversible chain has real eigenvalues. We can check reversibility by using Kolmogorov's Criterion, stated in theorem C.1.1.

Theorem C.1.1 (Kolmogorov's Criterion [135])**.** *An ergodic, positive recurrent Markov Chain P with finite state space Ω is reversible if and only if P satisfies*

$$p_{x_1x_2}p_{x_2x_3}\ldots p_{x_{n-1}x_n}p_{x_nx_1} = p_{x_1x_n}p_{x_nx_{n-1}}\ldots p_{x_3x_2}p_{x_2x_1} \tag{C.1}$$

for all finite sequences of states $x_1, \ldots, x_n \in \Omega$.
In words, eq. (C.1) *states that for any closed loop path, the probability of traversing the loop in the forward and backwards directions must be equal.*

A Markov chain P is *lazy* if the probability of remaining in any state is at least $1/2$. A lazy, reversible Markov chain has non-negative eigenvalues. We can therefore write the eigenvalues of P in descending order as $\lambda_1 = 1 > \lambda_2 \geq \ldots \lambda_n \geq 0$.

We can then define the *eigenvalue gap* of the chain by $\delta(P) \coloneqq |\lambda_1| - |\lambda_2| = 1 - \lambda_2$. The eigenvalue gap is closely related to the mixing time of a Markov chain. Indeed, we have lower and upper bounds for the mixing time:

Theorem C.1.2 (e.g., [136])**.** *Let P be a reversible Markov chain with stationary distribution π. Let $\pi_{\min} \coloneqq \min_{x\in\Omega} \pi_x$. Then,*

$$\log(2)\left(\frac{1}{\delta(P)} - 1\right) \leq t_{mix} \leq \frac{1}{\delta(P)}\log\left(\frac{4}{\pi_{\min}}\right). \tag{C.2}$$

While this is a tight bound, the eigenvalue gap can be prohibitively hard to compute in practice. A common tool for bounding the eigenvalue gap is the *conductance* property. Loosely, the conductance is a measure of how well-connected the chain is.

We first define the *ergodic flow* $Q(S,T)$ between subsets $S, T \subset \Omega$ by $Q(S,\, T) \coloneqq \sum_{x\in S,\, y\in T} \pi_x p_{xy}$. For a subset S, we also define $\pi(S) \coloneqq \sum_{x\in S} \pi_x$ and the *bottleneck ratio* $\Phi(S) \coloneqq \frac{Q(S,S^c)}{\pi(S)}$. The *conductance* Φ is then given by:

$$\Phi \coloneqq \min_{\substack{S\subset\Omega \\ \pi(S)\leq 1/2}} \Phi(S). \tag{C.3}$$

A result of Alon and Milman [137] bounds the eigenvalue gap by the conductance.

Theorem C.1.3 ([137])**.** *For a lazy, reversible Markov chain P, the eigenvalue gap $\delta(P)$ satisfies:*

$$\frac{\Phi^2}{2} \leq \delta(P) \leq 2\Phi. \tag{C.4}$$

The conductance remains hard to compute in many cases. We therefore require one further bound, via the *congestion* ρ.

For any pair of states $x,\, y$, define a canonical path $\gamma_{xy} = (x = z_0,\, z_1,\, \ldots,\, z_{k-1},\, z_k = y)$. The canonical path need not be the shortest path between the two points, but that will suffice for our purposes.

Let $\Gamma = \{\gamma_{xy} : x,\, y \in X\}$ be the set of all canonical paths. The congestion is given by:

$$\rho = \rho(\Gamma) = \max_{(u,\,v)\in E} \left\{ \frac{1}{\pi_u p_{uv}} \sum_{\substack{x,\, y\in\Omega: \\ u,\, v\in\gamma_{xy}}} \pi_x \pi_y \right\}. \tag{C.5}$$

The $\pi_u p_{uv}$ term is intuitively the "capacity" of the edge $(u,\, v)$ in the graph, in that it relates how much traffic the edge sees in the stationary distribution. The sum then counts the "load" the edge experiences within the family of canonical paths. The congestion is then the maximum load of any edge, normalised by its capacity. As expected, high congestion corresponds to low conductance:

Theorem C.1.4 (e.g., [134])**.** *For any reversible Markov chain and any choice of canonical paths Γ,*

$$\Phi \geq \frac{1}{2\rho(\Gamma)}. \tag{C.6}$$

Combining theorem C.1.3 & theorem C.1.4, we may trivially obtain:

Theorem C.1.5. *For a lazy, reversible Markov chain P and any choice of canonical paths Γ, the eigenvalue gap $\delta(P)$ is bounded by:*

$$\delta(P) \geq \frac{1}{8\rho(\Gamma)^2}. \tag{C.7}$$

C.2 A quantum search algorithm

In this appendix, we briefly summarise the functioning of the quantum search algorithm due to Magniez et al [47].

Having prepared a state $|\pi\rangle$ that encodes the stationary distribution (with cost S), the goal is to prepare the normalised projection of $|\pi\rangle$ onto the marked subspace, which we call $|\mu\rangle$. The standard, Grover-esque method is to implement a rotation in the 2D space $\mathcal{S} \coloneqq \mathrm{span}(|\pi\rangle, |\mu\rangle)$. As usual, this rotation is effected by composing two reflections.

The first reflection is about a state $\left|\mu^{\perp}\right\rangle$, a state orthogonal to $|\mu\rangle$. In $\mathcal{S}$, this is equivalent (up to a phase) to reflecting about the marked subspace. This can be done at the cost of checking whether an element is marked, accruing a complexity of C.

The second reflection is about the state $|\pi\rangle$. This could be done by rotating $|\pi\rangle$ to a fixed reference state (e.g. $|0\rangle$), reflecting about that state, and then rotating back. This induces a cost of $2(S+U)$, since it requires the operator used to prepare $|\pi\rangle$ and its inverse. In general, S is costly and should be avoided. Instead, the authors propose using a quantum walk operator that can be implemented at a cost of only $4U$, combined with a phase estimation procedure. Since the chain is known to have a unique eigenvalue 1 corresponding to the eigenvector π, the phase estimation algorithm can distinguish between parts of the state aligned with or orthogonal to $|\pi\rangle$.

By deriving a quadratic relation between the eigenvalue gap of the chain P and the gap in the phase of the quantum walk, it can be shown that only $\mathcal{O}\left(1/\sqrt{\delta}\right)$ calls to the walk are needed. The $\mathcal{O}(1/\sqrt{\epsilon})$ scaling follows from a technical requirement to use a randomised Grover's algorithm [138], since the probability of a marked element is not known *a priori*.

C.3 A Markov chain on a perfect tree

We first restate the definition of P from the main text.

Let $\mathcal{T}$ be a perfect $(d+1)$-ary tree with depth d. Let Ω be the set of nodes of $\mathcal{T}$, given by:

$$\Omega = \{\mathbf{j} = (j_1, \ldots, j_l, 0, \ldots, 0) \in \{0, \ldots, d+1\}^d : 0 \leq l \leq d; 1 \leq j_i \leq d\}. \tag{C.8}$$

The transition probabilities of P are:

$$\text{If } \mathbf{j} \text{ is neither root nor leaf,} \quad p_{\mathbf{jj}'} = \begin{cases} \frac{1}{2} & \text{if } \mathbf{j}' = \mathbf{j}, \\ \frac{1}{4(d+1)} & \text{if } \mathbf{j}' \text{ is a child of } \mathbf{j}, \\ \frac{1}{4} & \text{if } \mathbf{j}' \text{ is the parent of } \mathbf{j}, \\ 0 & \text{otherwise.} \end{cases} \tag{C.9}$$

$$\text{If } \mathbf{j} \text{ is a leaf,} \quad p_{\mathbf{jj}'} = \begin{cases} \frac{1}{2} & \text{if } \mathbf{j}' = \mathbf{j}, \\ \frac{1}{2} & \text{if } \mathbf{j}' \text{ is the parent of } \mathbf{j}, \\ 0 & \text{otherwise.} \end{cases} \tag{C.10}$$

$$\text{If } \mathbf{j} \text{ is the root,} \quad p_{\mathbf{jj}'} = \begin{cases} \frac{1}{2} & \text{if } \mathbf{j}' = \mathbf{j}, \\ \frac{1}{2(d+1)} & \text{if } \mathbf{j}' \text{ is a child of } \mathbf{j}, \\ 0 & \text{otherwise.} \end{cases} \tag{C.11}$$

The set of edges of P is $E = \{(\mathbf{j}, \mathbf{j}') \in \Omega \times \Omega : j_i = j'_i, i = 1, \ldots, l; j_{l+1} = 0; l = d-1 \vee j'_{l+2} = 0\}$.

P is easily seen to be lazy and ergodic. Using Kolomogorov's Criterion (theorem C.1.1), P is also reversible due to the following. All closed loops must consist of a path that doubles back on itself, since the in-degree of each vertex is 1. Therefore, for any transition probability $p_{\mathbf{jj}'}$ that appears in the forward direction, there must also be a corresponding $p_{\mathbf{j}'\mathbf{j}}$. Since the backward direction is formed by swapping the direction of each transition in the forward direction, simply interchanging each of the above pairs of terms results in the backward direction matching the forward, as required.

To compute the stationary distribution, we use the symmetry of the walk to infer that for any nodes $\mathbf{j}, \mathbf{j}'$ in the same layer, $\pi_{\mathbf{j}} = \pi_{\mathbf{j}'}$. It is therefore helpful to consider a simpler Markov chain Q defined on the layers of the tree, with state space $\chi = \{l : 0 \leq l \leq d\}$. Q is coupled to P such that if P is in state $\mathbf{j}$ in level l, then Q is in state l. If the stationary distribution of Q is $\tilde{\pi}$, then π can be recovered by distributing the probability mass $\tilde{\pi}_l$ equally among all the nodes at layer l.

A tri-diagonal matrix gives the transition probabilities of Q:

$$Q = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & & & & & \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & & & & \\ & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & & & \\ & & & \ddots & & & & \\ & & & & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & \\ & & & & & \frac{1}{2} & \frac{1}{2} & \end{bmatrix}. \tag{C.12}$$

We may further simplify by writing $Q = \frac{1}{2}(I + \tilde{Q})$, and noting that $\tilde{\pi}Q = \tilde{\pi} \Leftrightarrow \tilde{\pi}\tilde{Q} = \tilde{\pi}$. $\tilde{Q}$ defines a symmetric random walk with reflecting barriers. A direct computation yields:

$$\tilde{\pi}_l = \begin{cases} \frac{1}{2d} & \text{if } l = 0,\ d, \\ \frac{1}{d} & \text{if } l = 1, \ldots, d-1. \end{cases} \tag{C.13}$$

For any pair of nodes $\mathbf{j}, \mathbf{j}'$, take the canonical path $\gamma_{\mathbf{j}\mathbf{j}'}$ to be the shortest path from $\mathbf{j}$ to $\mathbf{j}'$. We may now compute the congestion of P, which is given by eq. (C.14):

$$\rho = \max_{(\mathbf{j},\mathbf{j}') \in E} \left\{ \rho_{\mathbf{j}\mathbf{j}'} \coloneqq \frac{1}{\pi_{\mathbf{j}} p_{\mathbf{j}\mathbf{j}'}} \sum_{\substack{\mathbf{k},\mathbf{k}' \in \Omega: \\ \mathbf{j},\mathbf{j}' \in \gamma_{\mathbf{k}\mathbf{k}'}}} \pi_{\mathbf{k}} \pi_{\mathbf{k}'} \right\}. \tag{C.14}$$

For a self-edge $(\mathbf{j}, \mathbf{j})$, the congestion $\rho_{\mathbf{j}\mathbf{j}}$ is 0 since no canonical path goes through any self-edges.

For an edge between distinct nodes $\mathbf{j}, \mathbf{j}'$, assume $\mathbf{j}$ is in layer l and $\mathbf{j}'$ is in layer $l+1$. Then $\mathbf{j}, \mathbf{j}' \in \gamma_{\mathbf{k}\mathbf{k}'}$ if and only if $\mathbf{k}$ equals or is a descendent of $\mathbf{j}'$ and $\mathbf{k}'$ is not a descendant of $\mathbf{j}'$, or vice versa.

For $\mathbf{j}'$, $\mathbf{k} \in \Omega$, write $\mathbf{k} \preceq \mathbf{j}'$ if $\mathbf{k} = \mathbf{j}'$ or $\mathbf{k}$ is a descendant of $\mathbf{j}'$. Then, for $\mathbf{j}$ not the root, we have:

$$\rho_{\mathbf{j}\mathbf{j}'} = \frac{1}{\pi_{\mathbf{j}} p_{\mathbf{j}\mathbf{j}'}} \left(\sum_{\substack{\mathbf{k}:\mathbf{k} \preceq \mathbf{j}' \\ \mathbf{k}':\mathbf{k}' \not\preceq \mathbf{j}'}} \pi_{\mathbf{k}} \pi_{\mathbf{k}'} + \sum_{\substack{\mathbf{k}':\mathbf{k}' \preceq \mathbf{j}' \\ \mathbf{k}:\mathbf{k} \not\preceq \mathbf{j}'}} \pi_{\mathbf{k}} \pi_{\mathbf{k}'} \right) \tag{C.15}$$

$$= d(d+1)^l \cdot 4(d+1) \cdot 2 \sum_{\mathbf{k}:\mathbf{k} \preceq \mathbf{j}'} \pi_{\mathbf{k}} \sum_{\mathbf{k}':\mathbf{k}' \not\preceq \mathbf{j}'} \pi_{\mathbf{k}'}. \tag{C.16}$$

For $\mathbf{j} = \mathbf{0}$, the root, we have a slightly modified expression, since $\pi_{\mathbf{0}}$ is reduced by a factor of 2 and $p_{\mathbf{0j'}}$ is increased by a factor of 2:

$$\rho_{\mathbf{0j'}} = 2d(d+1)^l \cdot 2(d+1) \cdot 2 \sum_{\mathbf{k}:\mathbf{k}\preceq\mathbf{j'}} \pi_{\mathbf{k}} \sum_{\mathbf{k'}:\mathbf{k'}\not\preceq\mathbf{j'}} \pi_{\mathbf{k'}}. \tag{C.17}$$

Clearly, these modifications cancel, and we may treat the expressions on the same footing. Then we may compute the sums:

$$\sum_{\mathbf{k}:\mathbf{k}\preceq\mathbf{j'}} \pi_{\mathbf{k}} = \frac{\tilde{\pi}_{l+1}}{(d+1)^{l+1}} + \frac{(d+1)\tilde{\pi}_{l+2}}{(d+1)^{l+2}} + \frac{(d+1)^2\tilde{\pi}_{l+3}}{(d+1)^{l+3}} + \ldots + \frac{(d+1)^{d-l-1}\tilde{\pi}_d}{(d+1)^d} \tag{C.18}$$

$$= \frac{1}{d(d+1)^{l+1}} \left((d-l-1) + \frac{1}{2} \right) \tag{C.19}$$

$$= \frac{2(d-l)-1}{2d(d+1)^{l+1}}. \tag{C.20}$$

$$\sum_{\mathbf{k'}:\mathbf{k'}\not\preceq\mathbf{j'}} \pi_{\mathbf{k'}} = 1 - \sum_{\mathbf{k}:\mathbf{k}\preceq\mathbf{j'}} \pi_{\mathbf{k}} \tag{C.21}$$

$$= \frac{2d(d+1)^{l+1} - 2(d-l) + 1}{2d(d+1)^{l+1}}. \tag{C.22}$$

Noting that these expressions depend only on l and d, we may write ρ_l for the congestion of an edge from layer l to layer $l+1$. Finally, we have:

$$\rho_l = 8d(d+1)^{l+1} \cdot \frac{\big(2(d-l)-1\big)\big(2d(d+1)^{l+1} - 2(d-l) + 1\big)}{4d^2(d+1)^{2l+2}} \tag{C.23}$$

$$= \frac{2\big(2(d-l)-1\big)\big(2d(d+1)^{l+1} - 2(d-l) + 1\big)}{d(d+1)^{l+1}}. \tag{C.24}$$

This is maximised for $l = 0$, where the congestion takes the value

$$\rho_0 = \frac{2(2d-1)(2d^2+1)}{d(d+1)} < 8d. \tag{C.25}$$

By theorem C.1.5, the eigenvalue gap $\delta(P)$ of P is bounded by

$$\delta(P) \geq \frac{1}{8\rho^2} > \frac{1}{2^9 d^2}. \tag{C.26}$$

Appendix D

Bespoke transpilation methods for HUBO-QAOA

This appendix describes the bespoke transpilers we implemented for the HUBO–QAOA algorithm. The inputs are a Hamiltonian expressed as a real sum of Pauli strings (with only I and Z terms), a target hardware connectivity and a SWAP strategy. The hardware topology needs to have at least as many physical qubits as there are virtual qubits in the Hamiltonian. Implementations are currently provided for linear nearest-neighbour, heavy-hex, grid and all-to-all topologies. SWAP strategies are not relevant for the all-to-all connectivity.

We choose a SWAP depth d and initial qubit mapping via a search framed as MAX-SAT, see section 5.4 for details. The Hamiltonian is partitioned into sets

$$S_0,\, S_1,\, \ldots,\, S_d,\, S_\infty, \tag{D.1}$$

where S_i contains terms whose qubits first become contiguous after i SWAP layers under the chosen initial mapping. Terms in S_∞ never become contiguous and are routed using default methods.

For each S_i, we have a set of terms that are guaranteed to be neighbouring in the hardware layout after i SWAP layers. The 1- and 2-qubit terms can trivially be implemented using R_z and R_{zz} gates, respectively. So we need only consider the multi-qubit terms acting on at least 3 qubits. A naive implementation consisting of a CX network, a rotation, and the inverse CX network typically incurs a significant circuit depth.

One mitigating strategy is to use a *factoring* approach, as introduced in [139] and further explored in [140, 141]. The key idea, as presented in [141], is that any term

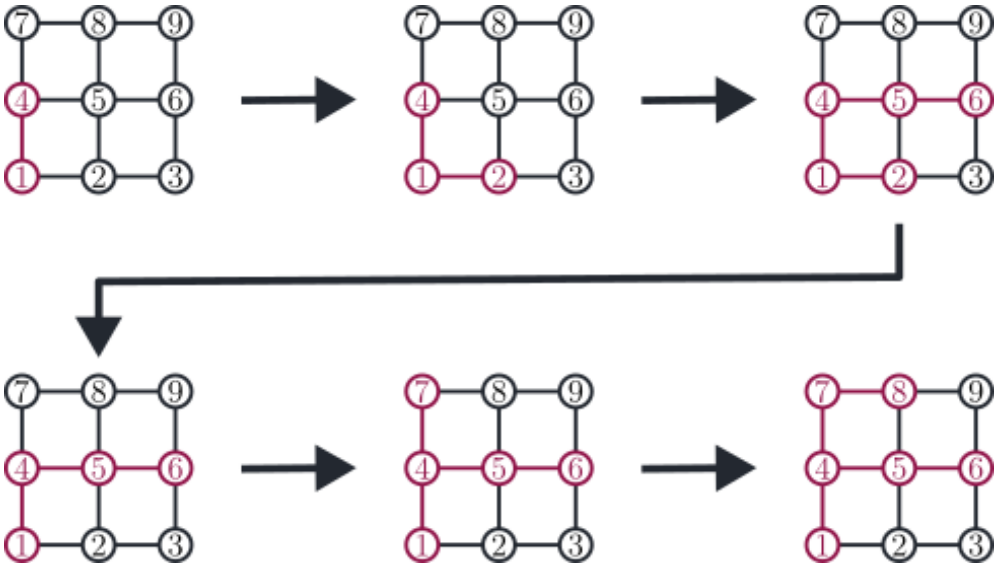


Fig. D.1 An example of a chain of interactions on a grid topology that can be implemented using minimal CX gates by our bespoke transpiler. Interactions at each link of the chain are shown in red. Each interaction shares the common rotation site (1, 4).

which is a strict subset of another term can be implemented without incurring any CX cost by placing an R_z gate at an appropriate point in the CX network for the other term.

This approach has two limitations. First, it compiles all terms into R_z gates, incurring a small fixed overhead in two-qubit gates. Second, it ignores hardware connectivity: on devices with limited connectivity, arbitrary CX orderings are infeasible, preventing the required factoring.

Our method achieves greater factoring without this overhead and is inherently *hardware-aware*.

Rather than restricting to nested sequences, the algorithm finds "chains" that may increase or decrease in size. Each adjacent pair of interactions in a chain forms a subset–superset relation, and all interactions share a common "rotation site" – a single qubit or qubit pair where the R_z or R_{zz} rotation is applied. Two-qubit rotation sites are used here to further reduce circuit depth. An example is shown in fig. D.1.

Each chain term is inexpensive to implement, requiring few CX gates, and the CX network is torn down only after the final term. Allowing both expansion and contraction of chains maximises CX cancellation, while careful chain selection minimises depth and respects hardware connectivity.

Pseudocode for transpiling the multi-qubit interactions at a single layer is given in algorithm 3, with supporting procedures in algorithms 4 and 5.

D.1 The main loop

The main loop in algorithm 3 iterates until all interactions have been implemented. At each iteration, the algorithm operates in one of two modes depending on whether a chain of interactions is currently active.

If no chain is active, a small interaction is removed from the list. The algorithm then attempts to start a chain by locating a *suitable superset*; intuitively, a term that contains the current interaction's qubits and shares at least one pair of neighbouring qubits, which becomes the rotation site. If such a superset is found, the current interaction is applied to the circuit using that rotation site, the next interaction is removed from the list, and the loop proceeds. Otherwise, the initial interaction is applied and immediately torn down via the EndChain function.

When a chain is active, the algorithm first applies the next interaction in the chain using the fixed rotation site. It then tries to extend the chain by searching for either a suitable superset or subset that shares the rotation site. If such an extension is found, the chain continues; otherwise, it is terminated.

The `Blocked` variable tracks all qubits included in chains so far. This ensures that subsequent chains act only on disjoint qubit subsets, allowing them to execute in parallel. If no disjoint subset remains, `Blocked` is reset and the search restarts.

As a further optimisation, interactions in S_∞ are allowed to serve as chain candidates throughout the algorithm. Although their qubits never form a contiguous set under the chosen SWAP strategy, they can sometimes be realised with only linearly many CX gates – typically when a superset of their qubits has just appeared earlier in a chain.

An example of this is shown in fig. 5.9, though the associated logic is omitted from algorithm 3 for clarity.

D.2 Chain selection, interaction computing and chain ending logic

D.2.1 Superset and Subset

The Superset and Subset functions in algorithm 4 operate similarly. Each iterates over the set of available interactions, searching for candidates that act on qubit sets which are supersets or subsets of the current interaction in the chain.

For the superset case, if no rotation site has been fixed, all possible connected pairs of qubits within the current interaction are tested as potential rotation sites. For the subset case, the rotation site must already lie within the candidate interaction.

This function first checks that all information stored in the qubits of `Missing` is connected to the rotation site. It then verifies that each connected component of `Missing` is also connected to the rotation site. Together, these conditions guarantee that some valid CX network exists to propagate information to the rotation site. To further limit circuit depth, MissingConnected calls CanFix, which determines whether each connected component can be resolved using at most a linear number of CX gates. If all checks pass, the candidate superset or subset is accepted and appended to the chain.

D.2.2 ComputeInteraction

The ComputeInteraction function constructs and applies the CX network required to implement a single interaction, followed by the appropriate R_{zz} rotation. The variable `MissingInfo` stores the qubits whose logical information must be added to or removed from the rotation site.

At each iteration, the algorithm selects a qubit from `MissingInfo` that is connected to the rotation site (the `Candidate`) and queues a CX gate between the candidate and one qubit in the rotation site. It then explores all the candidate's neighbours, pushing them onto a stack for potential inclusion in the CX network.

While the stack is non-empty, the algorithm pops a qubit and checks whether it still belongs to the updated `MissingInfo`, given the CX operations applied so far. If so, it queues another CX between this qubit and the neighbour that added it, then adds all of its neighbours to the stack. This process continues until the stack is empty, ensuring that an entire connected component of `MissingInfo` is resolved before proceeding.

Once all components have been processed, the queued CX gates are applied in reverse order, building information toward the rotation site. The `Stored` mapping is updated to track which qubits currently store which logical information, and finally, an R_{zz} rotation with angle ϕ_I is applied to the rotation site.

D.2.3 EndChain

The EndChain function finalises a chain when no further interactions can be added. All queued CX operations are reversed, returning each qubit to a state where it stores

only its own information. Other variables used in the main loop are reset to their starting values.

Algorithm 3 Pseudocode to transpile the interactions that are implementable at some layer of the bespoke HUBO-QAOA transpilation.

Require: List of interactions $\mathcal{I} = [I_1, \ldots, I_k]$, where I_j is a tuple of qubit indices

```
Available ← SortByLength(ℐ)                          ▷ increasing interaction size
CX ← []                                              ▷ list of CNOT pairs (control, target)
Stored ← {q : (q, ) for all qubits q}
Blocked ← ∅ Next ← None; Rotation ← None
while |ℐ| > 0 do
    if Next = None then                              ▷ start a chain
        Next ← Pop(ℐ)                                ▷ pop a (smallest) interaction
        Remove(Available, Next)
        Blocked ← Blocked ∪ Next
        (ChainNext, Rotation) ← Superset(Next, Available, Stored, None)
        if ChainNext ≠ None then
            Append(CX, ComputeInteraction(Next, Rotation, Stored))
            Next ← ChainNext
            Remove(ℐ, Next); Remove(Available, Next)
            Blocked ← Blocked ∪ Next
        else                                         ▷ no chain from this interaction
            Rotation ← PickAnyEdge(Next)             ▷ any two connected vertices
            ComputeInteraction(Next, Rotation, Stored)
            EndChain([], Next)
    else                                             ▷ continue a chain
        Append(CX, ComputeInteraction(Next, Rotation, Stored))
        (ChainNext, _) ← Superset(Next, Available, Stored, Rotation)
        if ChainNext ≠ None then
            Next ← ChainNext
            Remove(ℐ, Next); Remove(Available, Next)
            Blocked ← Blocked ∪ Next
            continue                                 ▷ go to top of while loop
        (ChainNext, _) ← Subset(Next, Available, Stored, Rotation)
        if ChainNext ≠ None then
            Next ← ChainNext
            Remove(ℐ, Next); Remove(Available, Next)
            Blocked ← Blocked ∪ Next
            continue                                 ▷ go to top of while loop
        EndChain(CX, Blocked)
    Available ← Filter(Available, I ↦ I ∩ Blocked = ∅)
    if |Available| = 0 then
        Blocked ← ∅
        Available ← SortByLength(ℐ)
return
```

Algorithm 4 Functions for choosing the next interaction in the chain, as defined in the HUBO-QAOA transpilation.

```
1: function SUPERSET(Next, Available, Stored, Rotation)
2:     for I ∈ Available do
3:         if I ⊇ Next then
4:             Missing ← I \ Next
5:             if Rotation ≠ None then
6:                 if MISSINGCONNECTED(Missing, Rotation, Stored) then
7:                     return I, Rotation
8:             else
9:                 for all Rotation ∈ COMBINATIONS(Next, 2) do
10:                    if MISSINGCONNECTED(Missing, Rotation, Stored) then
11:                        return I, Rotation
12:    return None, None

13: function SUBSET(Next, Available, Stored, Rotation)
14:    for I ∈ Available do
15:        if I ⊆ Next and Rotation ⊆ I then
16:            Missing ← Next \ I
17:            if MISSINGCONNECTED(Missing, Rotation, Stored) then
18:                return I, Rotation
19:    return None, None

20: function MISSINGCONNECTED(Missing, Rotation, Stored)
21:    Nodes ← RotationSite ∪ △_{q∈Missing} Stored[q]
22:    Components ← CONNECTEDCOMPONENTS(Missing)
23:    if (
           CONNECTED(Nodes)
           and ALL(CONNECTED(Component, Rotation) for Component ∈
       Components)
           and ALL(CANFIX(Component, Stored) for Component ∈ Components)
       ) then
24:        return True
25:    return False

26: function CANFIX(Component, Stored)
27:    if ∃S ⊆ Component : △_{s∈S} Stored[s] = Component then
28:        return CONNECTED(S)
29:    return False
```

Algorithm 5 Functions for applying an interaction in a chain and ending a chain of interactions, as defined in the HUBO-QAOA transpilation.

function ComputeInteraction(I, Rotation, Stored)
    MissingInfo $\leftarrow I \triangle \big(\triangle_{x \in \texttt{Rotation}} \texttt{Stored}[x] \big)$
    CX $\leftarrow$ []; NewCX $\leftarrow$ []
    while |MissingInfo| > 0 **do**
        for all $q \in$ MissingInfo, $x \in$ Rotation **do**
            if Connected(q, x) **then**
                Candidate $\leftarrow q$; Site $\leftarrow x$
                break
        Append(NewCX, (Candidate, Site))
        Stack $\leftarrow$ []
        for all Neighbour $\in$ Neighbours(Candidate) **do**
            Append(Stack, (Candidate, Neighbour))
        while |Stack| > 0 **do**
            Previous, Qubit $\leftarrow$ pop(Stack)
            MissingFromComponent $\leftarrow$ MissingInfo$\triangle \big(\triangle_{(c,_) \in \texttt{NewCX}} \texttt{Stored}[c] \big)$
            if $q \in$ MissingFromComponent **then**
                Append(NewCX, (Qubit, Previous))
                for Neighbour $\in$ Neighbours(Qubit) **do**
                    Append(Stack,(Qubit, Neighbour))
        Covered $\leftarrow \bigcup_{(c,_) \in \texttt{NewCX}} \texttt{Stored}[c]$
        MissingInfo $\leftarrow$ MissingInfo $\setminus$ Covered
        Extend(CX, Reverse(NewCX))
    for all $(c, t) \in$ CX **do**
        ApplyCX(c, t)
        Stored$[t] \leftarrow$ Stored$[t] \triangle$ Stored$[c]$
    ApplyRzz(Rotation, $\phi_{\texttt{I}}$)
    return CX

function EndChain(CX, Blocked)
    for all $(c, t) \in$ Reverse(CX) **do**
        ApplyCX(c, t)
    CX $\leftarrow$ []
    Stored $\leftarrow \{q : (q,)$ for all qubits $q\}$
    Next $\leftarrow$ None; Rotation $\leftarrow$ None;
    return