Quantum Simulation: Upper and Lower Bounds

by

Bryan O'Gorman

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor K. Birgitta Whaley, Co-chair
Professor Umesh Vazirani, Co-chair
Assistant Professor Avishay Tal

Summer 2021

Quantum Simulation: Upper and Lower Bounds

Abstract

Quantum Simulation: Upper and Lower Bounds

by

Bryan O'Gorman

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor K. Birgitta Whaley, Co-chair

Professor Umesh Vazirani, Co-chair

Quantum computers now exist, and will likely continue to get bigger and better. But will they ever be useful? That is, are there real-world problems that future quantum computers will be able to solve well enough in a reasonable amount of time, but for which classical computers cannot do the same? This thesis presents a collection of results that together address this question from different directions. The first part limits the utility of quantum computers. We show how to characterize and minimize the cost (in time and memory) of simulating quantum computations on a classical computer in terms of the congestion (a graph property) of a graphical representation of the quantum computation. Therefore, for a quantum computation to have an advantage over classical computation, it must have large congestion. Even when that is the case, better classical simulations, costly though they may be, can help validate and develop quantum computers. We also prove that the fundamental problem of quantum chemistry, the electronic structure problem, is QMA-complete. Therefore, under standard complexity-theoretic assumptions, the solution must be represented using a quantum state, and yet still even quantum computers cannot efficiently find it. The second part includes methods for applying and compiling quantum algorithms in order to maximize the utility of the hardware we have, now and in the future. We introduce the strategy of generalized swap networks and show how to use them to compile quantum algorithms for quantum chemistry and classical optimization. We combine quantum computation with constraint programming in two ways: we show how to combine existing quantum algorithms to speed up the solution of constraint programming problems, which capture a wide range of realistic application domains, and also discuss the application of constraint programming to compiling quantum algorithms. We also present a framework for generalizing the Quantum Approximate Optimization Algorithm to what we call the Quantum Alternating Operator Ansatz. Many of the algorithms are heuristic, but by improving the efficiency of their implementation on actual devices, we improve our chances of successfully trying them out and seeing if they

work or not.

To my family

Contents

Acknowledgments

First, I'd like to thank my advisors Birgitta Whaley and Umesh Vazirani, whose deep wisdom in their respective fields I have benefited from enormously, and who gave me guidance and freedom in the right proportions. Their leadership has made Berkeley a great place to do interdisciplinary work in quantum computing.

Thanks also to Avishay Tal for being on both my committees, and Lin Lin for being on my QE committe and a collaborator.

I've had the privilege of being part of the Quantum Artificial Intelligence Laboratory (QuAIL) at NASA Ames Research Center almost since its inception, and for most of my graduate work have been supported by a generous NASA Space Technology Research Fellowship. QuAIL is a special place at the intersection of academic and industrial work, of theory and application. Eleanor Rieffel, who now leads QuAIL, has been a mentor, friend, and collaborator throughout my graduate work. Thanks to Jeremy Frank, Minh Do, and Kyle Booth for learning quantum computing and teaching me classical planning and scheduling, resulting in many fruitful collaborations. Thanks also to Davide Venturelli and Zhihu Wang for our many collaborations.

As a member of three groups, I cannot hope to enumerate here all of the people that have benefited me. I'd like to thank Norm Tubman, Bill Huggins, Chinmay Nirkhe, Yunchao Liu, and Zeph Landau in particular for many interesting conversations. Most of the work described in this thesis was collaborative, and I thank my many other collaborators, besides those listed above: Sandy Irani, James Whitfield, Bill Fefferman, Joonho Lee, Unpil Baek, Stuart Hadfield, Jeffrey Marshall, Tony Tran, Jarrod McClean, Fabian Faulstich, Qinyi Zhu, Yiheng Qiu, Steven White, Ryan Babbush, Rupak Biswas, Thanh Nguyen, Parvathi Narayan, and Sasha Nanda.

Thanks to Michael Crescimanno for giving me my first taste of research and introducing me to quantum computing, Alan Aspuru-Guzik for giving me the opportunity to do my first work in quantum computing, and Alejandro Perdomo-Ortiz for bringing me to NASA and the many collaborations we had.

Many thanks to the UC Berkeley Symphony Orchestra for providing the soundtrack to my graduate work, especially Maestro David Milnes, my teacher Loren Mach, Katherine Zhou, Austin Darnell, Byron Zhou, and all the other percussionists, musicians, and composers with whom I've made music.

Thanks to my partner Leela, for bringing joy to my life, reading my drafts, and making me laugh. The biggest credit goes to my parents Donna Coen O'Gorman and Daniel O'Gorman, who have always supported me, and my sibling Rachel, who has always put up with me.

Chapter 1

Introduction

The theories of computation and quantum mechanics, midwifed early in the last century, are foundational to our modern understanding and manipulation of the world. The computational lens now pervades modern science, natural and social, and computers themselves now mediate much of our social and economic lives. Quantum mechanics is arguably the most precisely validated physical theory we have. Its use is necessary to explain important phenomena and allowed the engineering of amazing technologies, from displays and solar panels to computers themselves. At the intersection of computation and quantum mechanics is quantum computation: the theory and engineering of computational devices whose operation is quantum mechanical at the logical level (as opposed to computers whose *implementation* is quantum mechanical).

Just years after quantum computing was first proposed, independently by Manin [110] and Feynman [65], Deutsch [53] showed that in theory quantum computation can be better than classical, at least for some contrived problem. In the following decade, Shor invented their titular algorithm for factoring numbers[144], thereby showing that quantum computation can be advantageous for a problem of practical interest, at least compared to the best known classical algorithm. Subsequently, there has been significant progress in theoretical, algorithmic, and experimental quantum computing. Just two years ago, Google demonstrated a programmable quantum device [8] for simulating which there is no known efficient classical algorithm, achieving what is known as "quantum supremacy".

This thesis explores several ways of understanding the power of quantum computation in theory and improving the power of quantum computers in practice.

Classical simulation of quantum computations The only way currently known to verify that a putative quantum device has successfully performed random circuit sampling (the task performed in Google's "quantum supremacy" experiment) is to classically simulate the same experiment and compare the results. Architecture design choices and experimental noise can be exploited to make the classical simulations more efficient, but it is expected that all classical simulations have inherently exponential scaling. Nevertheless, classical simulation of quantum circuits is a useful tool, both for validating quantum devices and for designing quantum algorithms. Most state-of-the-art large-scale classical simulations of quantum circuits employ the formalism of *tensor networks*, a way of expressing certain quantities as sums of products that has found widespread use both as a tool for understanding and for simulating quantum systems (and classical counting problems). Chapter 2 of this thesis describes one way of parameterizing a tensor network that tightly characterizes both the time and space resources necessary to calculate its value via matrix multiplication (as all such calculations are done in practice). In contemporary simulations, memory is often the limiting factor; indeed, this work grew out of conversations with Benjamin Villalonga, who, as a leading contributor to efforts to simulate Google's experiments, was seeking ways of understanding and minimizing the bottlenecking memory requirements. The characterization of the time resources quantitatively matches the dominant term of earlier work [111], but is also more precise and conceptually much cleaner, which hopefully increases its potential utility. That the characterization is tight within a realistic computational model and also yields the memory requirements exemplifies its utility.

Computational complexity Random circuit sampling is of no value beyond demonstrating "quantum supremacy". The true goal of building quantum computers is to perform practically useful tasks better than classical computers can. Simulating quantum systems is the seminal task (as proposed by Feynman) and continuous to be considered one of the most promising applications of quantum computers. In particular, there is hope that quantum computers can be useful in solving the *electronic structure problem* (approximating the ground state energy of electrons subject to external potentials), the fundamental problem of quantum chemistry. Chapter 3 of this thesis limits what we can hope for in this regard, by proving that a reasonable formalization of the problem is QMA-complete. This implies that, under standard assumptions ($\mathsf{QMA} \neq \mathsf{BQP}$ and $\mathsf{QMA} \neq \mathsf{QCMA}$, respectively), not even quantum computers can efficiently solve the problem *in general*, even though the answer has no concise classical description. In other words, the significant structure imposed by physics on the electronic structure Hamiltonian is not enough to make the problem easy for quantum computers. But practically important instances of the electronic structure problem may have some additional, yet-unknown structure that can be exploited.

Quantum algorithms Classical algorithms and hardness results upper bound the relative and absolute power of quantum computers. Quantum algorithms lower bound it, by showing what quantum computers *can* do. As with classical algorithms, quantum algorithms can be both rigorous (with proven success probabilities) and heuristic (with intuitive justification but not proven to work). Section 5.2 of this thesis describes rigorous quantum algorithms for constraint programming, with a practical focus on early fault-tolerant devices. Many of the most successful classical algorithms are heuristic, and there is no reason to expect that the same will be true of quantum algorithms. We consider two of the most popular heuristic quantum algorithms: the Quantum Approximate Optimization algorithm for classical constraint satisfaction problems and the Variational Quantum Eigensolver for quantum Hamiltonians.

Both take the form of a parameterized quantum state (known as an *ansatz*), whose parameters are then optimized.

Outline This thesis is organized as follows. Part I describes two results that upper bound the possible advantage of quantum computation. Chapter 2 describes the parameterization of tensor network contraction, a general method for computing quantities in quantum computation. Chapter 3 proves that the fundamental problem of quantum chemistry, the electronic structure problem, is QMA-complete. Part II presents methods for applying and compiling quantum algorithms in order to maximize the utility of the hardware we have, now and in the future. Chapter 4 introduces the strategy of generalized swap networks and shows how to use them to compile quantum algorithms for quantum chemistry and classical optimization. Chapter 5 describes work at the intersection of quantum computation and classical optimization: the generalized Quantum Alternating Operator Ansatz in Section 5.1, the application of quantum computation to Constraint Programming and vice versa in Section 5.2, and the construction of phase transitions in instance families of the Single-Machine Scheduling problem in Section 5.3.

Part I

Upper bounding quantum advantage

Chapter 2

Parameterization of tensor network contraction

This chapter reproduces Ref. 123 with minor modification. In this chapter, we present a conceptually clear and algorithmically useful framework for parameterizing the costs of tensor network contraction. Our framework is completely general, applying to tensor networks with arbitrary bond dimensions, open legs, and hyperedges. The fundamental objects of our framework are rooted and unrooted contraction trees, which represent classes of contraction orders. Properties of a contraction tree correspond directly and precisely to the time and space costs of tensor network contraction. The properties of rooted contraction trees give the costs of parallelized contraction algorithms. We show how contraction trees relate to existing tree-like objects in the graph theory literature, bringing to bear a wide range of graph algorithms and tools to tensor network contraction. Independent of tensor networks, we show that the edge congestion of a graph is almost equal to the branchwidth of its line graph.

2.1 Introduction

Tensor networks are widely used in chemistry and physics. Their graphical structure provides an effective way for expressing and reasoning about quantum states and circuits. As a model for quantum states, they have been very successful in expressing ansatzes in variational algorithms (e.g., PEPS, MPS, and MERA). As a model for quantum circuits, they have been used in state-of-the-art simulations [56, 57, 68, 129, 153]. In the other direction, quantum circuits can also simulate tensor networks, in the sense that (additively approximate) tensor network contraction is complete for quantum computation [6].

The fundamental computation in the application of tensor networks is tensor network contraction, i.e., computing the single tensor represented by a tensor network. Tensor network contraction is #P-hard in general [17] but fixed-parameter tractable. Markov and Shi [111] defined the contraction complexity of a tensor network and showed that contraction can be done in time that scales exponentially only in the treewidth of the line graph of the tensor network. Given a tree decomposition of the line graph of a tensor network, a contraction order can be found such that the contraction takes time exponential in the width of the decomposition, and vice versa. However, the translation between contraction orders and tree decompositions does not account for polynomial prefactors. This is acceptable in theory, where running times of $O(n2^n)$ and of $O(2^n)$ are both "exponential"; in practice, the difference between $\Theta(n2^n)$ and $\Theta(2^n)$ can be the difference between feasible and infeasible.

We give an alternative characterization of known results in terms of tree embeddings of the tensor network rather than tree decompositions of the line graph thereof. In this context, we call such tree embeddings *contraction trees*. While one can efficiently interconvert between a contraction tree of a tensor network and a tree decomposition of the line graph, contraction trees exactly model the matrix multiplications done by a contraction algorithm in an abstract way. That is, the time complexity of contraction is exactly and directly expressed as a property of contraction trees, in contrast to tree decompositions of line graphs, which only capture the exponent. Our approach is thus more intuitive and precise, and easily applies to tensor networks with arbitrary bond dimensions and open legs.

We show that contraction trees also capture the space needed by a matrix-multiplication-based contraction algorithm. In practice, space often competes with time as the limiting constraint. Even further, we can express the time used by parallel algorithms as a property of *rooted contraction trees*, which are to contraction orders as partial orders are to total orders.

In a contraction tree, tensors are assigned to the leaves and each wire is "routed" through the tree from one leaf to another. The congestion of a vertex of the contraction tree is the number of such routings that pass through it, and similarly for the congestion of an edge. The vertex congestion of a graph G, denoted $\mathrm{vc}(G)$, is the minimum over contraction trees of the maximum congestion of a vertex, and similarly for the edge congestion, denoted $\mathrm{ec}(G)$. Formally, our main results are the following two theorems.

Theorem 1. A tensor network $(G, \mathcal{M})$ can be contracted in time $n2^{\mathrm{vc}(G)+1}$ and space $n2^{\mathrm{vc}(G)+1}$, or in time $2^{1.5\mathrm{ec}(G)+1}$ and space $2^{\mathrm{ec}(G)+1}$. More precisely, the tensor network can be contracted in time $\min_{(T,b)} \sum_{t \in T} 2^{\mathrm{vc}(t)}$, where the minimization is over contraction trees (T, b). The contraction can be done using space equal to the minimum weighted, directed modified cutwidth of a rooted contraction tree using edge weights $w(f) = 2^{\mathrm{ec}(f)}$. If the contraction is done as a series of matrix multiplications, these precise space and time bounds are tight (though not necessarily simultaneously achievable).

Theorem 2. A parallel algorithm can contract a tensor network $(G, \mathcal{M})$ in time $\min_{(T,b)} \max_l \sum_t 2^{\mathrm{vc}(t)}$, where the minimization is over rooted contraction trees (T, b), the maximization is over leaves l of T, and the summation is over vertices of t on the unique path from the leaf l to the root r. In other words, the time is the minimum vertex-weighted height of a rooted contraction tree, where the weight of a vertex is $w(t) = 2^{\mathrm{vc}(t)}$. If the contraction is done as matrix multiplications in parallel, this is tight.

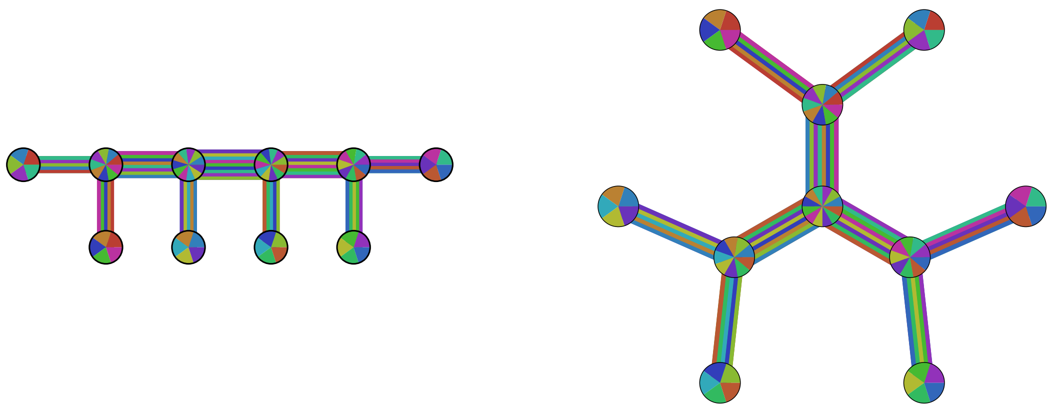

Figure 2.1: Two unrooted contraction trees for a tensor network with 6 tensors. Each color corresponds to a wire of the tensor network. The inclusion of a color in the representation of a vertex or edge of the contraction tree indicates the contribution of the corresponding wire's weight to the congestion of the vertex or edge, respectively.

Given a tree decomposition of a line graph with width $k - 1$, we can efficiently construct a contraction tree of the original graph with vertex congestion k. Thus one immediate application of our framework is as a way of precisely assessing the costs of contraction implied by different tree decompositions (even of the same width) computed using existing methods. This is especially useful in distinguishing between contraction orders that have the same time requirements but different space requirements; prior to this work, there was no comprehensive way of quantifying the space requirements, which in practice can be the limiting factor. Alternatively, one can start with existing algorithms for computing good branch decompositions, which can be converted into contraction trees of small edge congestion. More broadly, identifying the right abstraction (i.e., contraction trees) and precise quantification of the space and time costs is a foundation for minimizing those costs as much as possible.

In Section 2.2, we go over the graph-theoretic concepts that are the foundation of this work. In Section 2.3, we present seemingly unrelated graph properties in a unified framework that may be of independent interest. Section 2.3, while strictly unnecessary for understanding the main results, helps explain the relationship between our work and prior work. In Section 2.4, we introduce the cost model on which our results are based. In Section 2.5, we give our main results. In Section 2.6, we discuss extensions and generalizations of the main ideas. In Section 2.7, we prove that the edge congestion of a graph is almost equal to the branchwidth of its line graph.

There are several possible directions for future work:

- Proving the hardness of exactly or approximately computing the vertex or edge congestion of a graph, including of special cases like planar graphs.

- Inventing algorithms (that aren't simply disguised treewidth or branchwidth algorithms) for finding small-congestion contraction trees.
- Exploring the space-time trade-off of vertex and edge congestions. They are always within a small multiplicative constant of each other, but can they be exactly minimized simultaneously? If not, what does the trade-off look like, particularly for graphs of practical interest like 2D and 3D grids.
- Parallelizing at larger scale. In our discussion of parallelized algorithms, we neglected communication costs. While this is probably reasonable at a relatively small number of parallel processes (i.e., that can be on a single multi-processor node of a cluster), at larger scales it may become material and worth trying to minimize.
- Adapting our methods to approximate tensor network contraction.
- Finding analogous methods for optimizing tensor-network ansatzes. For example, it is known that optimizing (bounded-bond dimension) tree tensor networks is easy. Can this be generalized in a parameterizable way as we did for tensor network contraction?

2.2 Background

Let $[i,n] = \{j \in \mathbb{Z} | i \leq j \leq n\}$, $[n] = [1,n]$, and $[\mathbf{n}] = [n_1] \times [n_2] \times \cdots \times [n_r]$ for $\mathbf{n} = (n_1, \ldots, n_r) \in (\mathbb{Z}^+)^r$. Let $G[S] = \left(V, E \cap \binom{S}{2}\right)$ be the subgraph of G induced by a subset of the vertices $S \subset V(G)$. For two disjoint sets of vertices of an edge-weighted graph, $w(S,S') = \sum_{\{u,v\} \in E | u \in S, v \in S'} w(\{u,v\})$ is the sum of the weights of the edges between $S \subset V$ and $S' \subset V$. More generally, for r disjoint sets of vertices, $w(S_1, \ldots, S_r) = \sum_{\{i,j\} \in \binom{[r]}{2}} w(S_i, S_j)$ is the sum of the weights of the edges with endpoints in distinct sets. In this context, we will denote singleton sets by their sole elements, e.g., $w(u,v) = w(\{u\},\{v\}) = w(\{u,v\})$. Let $N(v)$ be the neighbors of a vertex v.

2.2.1 Tensor networks and contraction

A *tensor* can be defined in several equivalent ways. Most concretely, it is a multidimensional array. Specifically, a rank-r tensor is an r-dimensional array of size $\mathbf{d} = (d_1, \ldots, d_r)$. More abstractly, a tensor is a collection of numbers indexed by the Cartesian product of some set of indices, e.g., $[T_{i_1, i_2 \ldots, i_r}]_{(i_1, i_2, \ldots, i_r) \in [d_1] \times [d_2] \times \cdots \times [d_r]}$ indexed by $\mathbf{i} \in [\mathbf{n}]$. Alternatively, a tensor can be thought of as a multilinear map $T : [\mathbf{d}] \to \mathbb{C}$. (Our focus will be on complex-valued tensors.)

Definition 1 (Tensor network)**.** A *tensor network* $(G, \mathcal{M})$ is an undirected graph G with edge weights w and a set of tensors $\mathcal{M} = \{\mathbf{M}_v | v \in V(G)\}$ such that $\mathbf{M}_v$ is a $|N(v)|$-rank

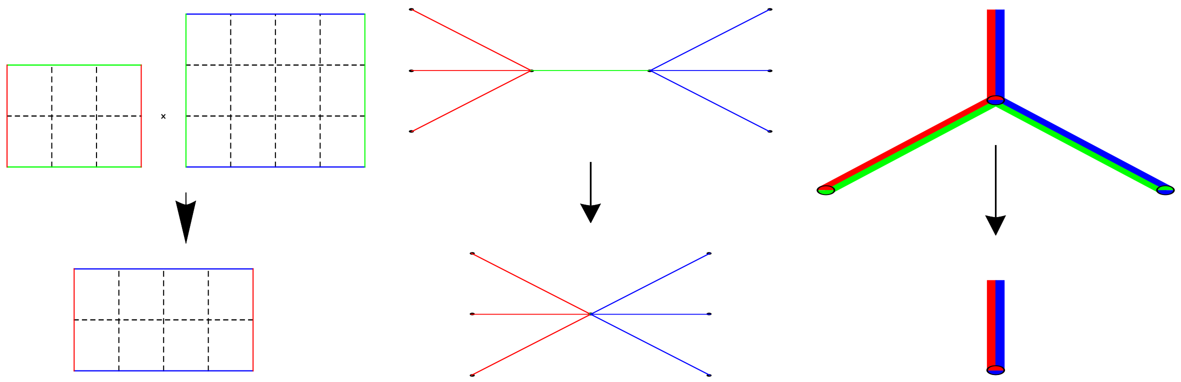

Figure 2.2: Three ways of viewing the contraction of two tensors. Left: multiplication of a $d_{\mathrm{L}} \times d_{\mathrm{M}}$-rank tensor with a $d_{\mathrm{M}} \times d_{\mathrm{R}}$-rank tensor, resulting in a $d_{\mathrm{L}} \times d_{\mathrm{R}}$-rank tensor. Middle: contraction of a degree $w_{\mathrm{L}} + w_{\mathrm{M}}$ vertex with a degree $w_{\mathrm{M}} + w_{\mathrm{R}}$ vertex, resulting in a degree $w_{\mathrm{L}} + w_{\mathrm{R}}$ vertex. Right: removing a close pair of leaves (with congestions $w_{\mathrm{L}} + w_{\mathrm{M}}$ and $w_{\mathrm{M}} + w_{\mathrm{R}}$) of a contraction tree, leaving a new leaf with congestion $w_{\mathrm{L}} + w_{\mathrm{R}}$.

tensor with $2^{\deg(v)}$ entries, where $\deg(v) = \sum_{u \in N(v)} w(\{u, v\})$ is the weighted degree of v. Each edge e corresponds to an index $i \in \left[2^{w(e)}\right]$ along which the adjacent tensors are to be contracted.

The contraction of two tensors is the summation over the values of their shared indices. Graphically, this is like an edge contraction of the edge adjacent to the two tensors. The result is a new tensor that takes the place of the two original ones. [1] See Figure 2.2. Let v_1 and v_2 be the vertices contracted into the new vertex $v_{\{1,2\}}$. The weight of an edge between the new vertex and any other vertex v' is $w\left(v_{\{1,2\}}, v'\right) = w\left(\{v_1, v_2\}, v'\right) = w\left(v_1, v'\right) + w\left(v_2, v'\right)$.

Except in Section 2.6, we assume that all tensor networks have no "open legs", i.e., every edge connects two vertices (tensors). In this case, the value of a tensor network is the single number that results from contracting all of its edges. Each contraction reduces the number of vertices (tensors) by one, so the network is fully contracted by $n - 1$ contractions. We call a sequence of such contractions a *contraction order*. The value of the tensor network is independent of the contraction order, but the cost of doing the contraction can vary widely depending on the contraction order. Each contraction is identified by an edge, but that edge may not be in the original graph, i.e., its adjacent vertices may have been formed by earlier contractions. One way of specifying a contraction order is by a sequence of edges of the original graph that constitute a spanning tree thereof. In Section 2.5, we introduce the notion

[1] Note that this is a "parallel" model of contraction, whereas Markov and Shi use "one-edge-at-a-time" contraction of multigraphs. They are equivalent in the sense that an edge with integer weight can be considered as that number of (unweighted) parallel edges. The parallel model more closely matches how contraction is done in practice. It also allows for arbitrary bond dimension, whereas the multigraph model requires that all bond dimensions be powers of the same base.

of contraction trees, which allow for a conceptually clear way of expressing contraction orders that makes manifest the associated temporal and spatial costs.

Exactly computing the value of a tensor network is #P-hard [16], as a tensor network can be constructed that counts the number of satisfying assignments to a satisfiability instance or the number of proper colorings of a graph. Even multiplicative and additive approximation is NP-hard [6]. Interestingly, approximating the value of a tensor network with bounded degree and bounded bond dimension is BQP-complete [6]. That is, not only can tensor networks simulate quantum circuits, but quantum circuits can simulate tensor networks as well. In this sense, tensor networks and quantum circuits are computationally equivalent.

2.2.2 Treewidth and branchwidth

This section is intended primarily to establish notation and recapitulate the standard definitions of the graph properties used in the present work. For a more thorough and pedagogical treatment, see Diestel's excellent textbook [54]. Many instances of graph problems that are hard in general are actually easy when instance graphs are restricted to trees. In many such cases, this generalizes in the sense that it is possible to characterize the hardness of an instance by how "tree-like" it is, as captured by the treewidth of the graph. The treewidth of a graph is defined in terms of an optimal tree decomposition. Treewidth has several alternative characterizations; one of these, *elimination width*, is the basis of Markov and Shi's result equating treewidth and contraction complexity.

Definition 2 (Tree decomposition)**.** A *tree decomposition* of a graph $G = (V, E)$ is a tuple $(T, \mathcal{X})$ of a tree T and a tuple $\mathcal{X} = (X_t)_{t \in V(T)}$ of subsets (called *bags*) of the vertices of G with the following properties.

1. For every edge $\{u, v\} \in E(G)$, there is some bag $X \in \mathcal{X}$ that contains both endpoints: $u, v \in X$.

2. For every vertex $v \in V(G)$ of G, the subtree S_v of T induced by the bags $S_v = \{X \in \mathcal{X} | v \in X\}$ containing v is non-empty and connected.

Definition 3 (Width and treewidth)**.** The *width* of a tree decomposition $(T, \mathcal{X})$ of a graph G is one less than the size of the largest bag: $\text{width}(G, T, \mathcal{X}) = \text{width}(\mathcal{X}) = \max_{X \in \mathcal{X}} |X| - 1$. The *treewidth* of a graph is the minimum width of a tree decomposition of the graph.

A related concept is that of path decompositions and pathwidth, defined analogously to tree decompositions and treewidth, except restricted to paths rather than trees.

Definition 4 (Path decomposition and pathwidth)**.** A *path decomposition* of a graph G is a tree decomposition $(T, \mathcal{X})$ of G such that T is a path. The pathwidth $\text{pathwidth}(G)$ of G is the minimum width of a path decomposition of G.

Definition 5 (Branch decomposition)**.** A *branch decomposition* of a graph $G = (V, E)$ is a tuple (T, b) of a binary tree T and a bijective function $b : E(G) \to V(T)$ between the edges E of G and the leaves of T.

For each vertex $v \in V(G)$ of G, let $S_v \subset V(T)$ be the minimal spanning tree of T that contains all the leaves corresponding to edges adjacent to v.

Definition 6 (Branchwidth)**.** The *width*, denoted $\text{width}_G(T, b, \{s, t\})$, of an edge $\{s, t\} \in E(T)$ of a branch decomposition (T, b) of a graph G is $|\{v \in V(G) | \{s, t\} \subset S_v\}|$, i.e., the number of vertices of G such that the subtree $T[S_v]$ contains $\{s, t\}$. The *width* of the branch decomposition is the largest width of an edge, $\text{width}_G(T, b) = \max_{f \in E(T)} \text{width}_G(T, b, f)$. The *branchwidth* $\text{branchwidth}(G) = \min_{(T,b)} \text{width}_G(T, b)$ of a graph is the minimum width of a branch decomposition thereof.

2.2.3 Congestion

There is an alternative but less explored way of quantifying how "tree-like" a graph is: the minimum congestion of a tree embedding, introduced by Bienstock [18].[2]

Definition 7 (Tree embedding)**.** A *tree embedding* of a graph G is a tuple (T, b) of a binary tree T and a bijection $b : V(G) \to V(T)$ between the vertices of G and the leaves of T.

Let $S_{v,w}$ be the unique path between the leaves $b(v)$ and $b(w)$ of T.

Definition 8 (Congestion)**.** The congestion of a vertex $v \in V(T)$ (resp., edge $f \in E(T)$) is the total weight of the edges $e \in E(G)$ whose subtrees S_e include v (resp., f).

2.2.4 Cutwidth

Definition 9 (Cutwidth)**.** Let $f : V \to [n]$ be a linear ordering of the vertices of a graph $G = (V, E)$. The cutwidth of f is the maximum number of edges that cross a gap:

$$\max_{i \in [n-1]} |\{\{u, v\} \in E | f(u) \leq i < f(v)\}| .$$

The modified cutwidth of f is the maximum number of edges that cross a vertex:

$$\max_{i \in [n]} |\{\{u, v\} \in E | f(u) < i < f(v)\}| .$$

The cutwidth (resp., modified cutwidth) of a graph is the minimum cutwidth (resp., modified cutwidth) of a linear ordering. For edge weighted graphs, the weighted cutwidth and modified cutwidth count the total weights of the relevant edge sets rather than their cardinalities. For a directed acyclic graph, the directed cutwidth (resp., modified cutwidth) is the minimum cutwidth (resp., modified cutwidth) of a linear ordering that is topologically sorted according to the graph.

[2] Note that this is entirely distinct from a different type of congestion problem in which the goal is find routings for some specified set of pairs of terminals.

Leaves	Subtrees	Minimization over	Target family: Trees	Target family: Caterpillars
Edges	Vertices	Vertices	Treewidth	Pathwidth
Edges	Vertices	Edges	Branchwidth	
Vertices	Edges	Vertices	Vertex congestion	Modified cutwidth
Vertices	Edges	Edges	Edge congestion	

Figure 2.3: Table of graph properties. Each row corresponds to an instantiation of Equation 2.1.

2.2.5 Parameterized complexity

Approximating both treewidth and pathwidth to within a constant factor is NP-hard, though there exist efficient algorithms for logarithmic and polylogarithmic approximations, respectively [24, 25]. However, deciding whether or not the treewidth is at most some constant can be done in linear time (albeit it with an enormous prefactor) [22]. For many graph problems, e.g., Maximum Independent Set, there exist algorithms whose run time is exponential only in the treewidth or pathwidth, i.e., given the instance graph and a tree decomposition thereof of width k, the algorithm runs in time $2^k n^{O(1)}$ [7]. The Exponential Time Hypothesis (ETH) implies that several such parameterized complexity results are optimal, in the sense that there exists no $2^{o(k)} n^{O(1)}$ algorithm [52].

The situation is similar for branchwidth. Computing the branchwidth of a graph is in general NP-hard, but can be done efficiently for planar graphs [142]. (Whether computing the treewidth of a planar graph is NP-hard is an open question.) As is the case for treewidth, there is a constructive linear time algorithm for deciding whether or not the branchwidth is at most some constant (and in this case with better constant factors) [23]. Good branch decompositions can be used to implement dynamic programming algorithms for problems such as the traveling salesman problem [46].

Computing the vertex congestion of a graph is claimed to be NP-hard [18], but no proof appears in the literature.

Computing the (edge) cutwidth is NP-hard, but for any constant k, a linear ordering of cutwidth k (for all variants) can be found in linear time if one exists [21].

2.3 Unified framework of graph properties

In this section, we present a unified framework of various graph properties, as captured in the following combined definition:

A $\left\{\begin{matrix}\text{tree decomposition}\\ \text{branch decomposition}\\ \text{tree embedding}\\ \text{tree embedding}\end{matrix}\right\}$ of a $\left\{\begin{matrix}\text{vertex}\\ \text{vertex}\\ \text{edge}\\ \text{edge}\end{matrix}\right\}$-weighted graph G is a tuple (T, b) of a binary tree T and a bijection b between the leaves of T and the $\left\{\begin{matrix}\text{edges}\\ \text{edges}\\ \text{vertices}\\ \text{vertices}\end{matrix}\right\}$ of G. The bijection b implies a subtree for every $\left\{\begin{matrix}\text{vertices}\\ \text{vertices}\\ \text{edges}\\ \text{edges}\end{matrix}\right\}$ of the graph. The $\left\{\begin{matrix}\text{treewidth}\\ \text{branchwidth}\\ \text{vertex congestion}\\ \text{edge congestion}\end{matrix}\right\}$ of the graph is the minimum over all $\left\{\begin{matrix}\text{tree decompositions}\\ \text{branch decompositions}\\ \text{tree embeddings}\\ \text{tree embeddings}\end{matrix}\right\}$ of the maximum total weight of all subtrees containing any $\left\{\begin{matrix}\text{vertex}\\ \text{edge}\\ \text{vertex}\\ \text{edge}\end{matrix}\right\}$. The $\left\{\begin{matrix}\text{pathwidth}\\ \text{—}\\ \text{modified cutwidth}\\ \text{—}\end{matrix}\right\}$ is defined in the same way as $\left\{\begin{matrix}\text{treewidth}\\ \text{—}\\ \text{vertex congestion}\\ \text{—}\end{matrix}\right\}$ except that T is restricted to be a caterpillar.

(2.1)

Let's unpack this. For branchwidth and congestions, (2.1) is the standard definition. For the others, (2.1) is non-standard but equivalent to the standard definitions. Writing them all in this way helps elucidate the relationships between them, which are obscured by the standard definitions.

Note that both the vertex and edge congestions of a graph G are defined as optimal properties of the same type of object, namely a tree embedding (T, b). For every edge $e \in E(G)$, the mapping $b : V(G) \to V(T)$ of the vertices to leaves of the tree implies a minimal subtree S_e connecting the leaves of T corresponding to its endpoints in G. (For an edge of size 2, this subtree S_e is a path, but the definition allows for hyperedges as well.) The vertex and edge congestions are then the maximum total weight of subtrees that contain any vertex or edge, respectively, of the tree T.

There is a similar relationship between treewidth and branchwidth. Usually, we think of a tree decomposition of a graph $G = (V, E)$ as a tree T and a subtree S_v for every vertex in $V(G)$ such that the subtrees for every pair of adjacent (in G) vertices overlap. In (2.1), S_v is specified implicitly as the (unique) minimal spanning subtree of T that connects the leaves of T corresponding to the edges of G that are incident to v. By design, this tree T and set of subtrees is the same as that for a branch decomposition. The treewidth and branchwidth are the maximum total weight of subtrees (now corresponding to vertices of G) that contain any vertex or edge, respectively, of the tree T.

So we see that the congestions are defined by the overlap of subtrees of T corresponding to edges of G and that the tree- and branchwidths are defined by the overlap of subtrees of T corresponding to vertices of G, the former implied by a mapping from vertices of G to leaves of T and the latter by a mapping from edges of G to leaves of T. The vertex congestion and treewidth are concerned with the overlap at vertices of T, and the edge congestion and branchwidth with the overlap on edges of T. Thus we have made the analogy that treewidth : branchwidth :: (vertex congestion) : (edge congestion). For example, that [138] $\mathrm{bw}(G) \leq \mathrm{tw}(G) \leq \frac{3}{2}\mathrm{bw}(G)$ and [18] $\mathrm{ec}(G) \leq \mathrm{vc}(G) \leq \frac{3}{2}\mathrm{ec}(G)$ is no coincidence.

Now consider the line graph $L(G) = (E, \{\{e, e'\} \subset E | e \cap e' \neq \emptyset\})$ of a graph $G = (V, E)$. Suppose we have a tree embedding (T, b) of the original graph G, with an implied subtree T_e for every edge $e \in E(G)$. Because the vertices of the line graph $L(G)$ correspond to the edges of G, this can be considered as a branch decomposition of the line graph $L(G)$. For every pair of edges $e, e' \in E(G) = V(L(G))$ that are adjacent in the line graph, the corresponding subtrees $S_e, S_{e'}$ intersect at the leaf $b(v) \in V(T)$, where $v \in e, e'$ is the vertex of G adjacent to e and e'. The vertex congestion of the tree embedding (T, b) is the width of (T, b) interpreted as a tree decomposition, and the edge congestion of the tree embedding is the width of (T, b) interpreted as a branch decomposition. This implies that $\mathrm{tw}(L(G)) \leq \mathrm{vc}(G)$ and $\mathrm{bw}(L(G)) \leq \mathrm{ec}(G)$. Actually, these inequalities are tight or almost so: $\mathrm{tw}(L(G)) = \mathrm{vc}(G)$ and $\mathrm{bw}(L(G)) \leq \mathrm{ec}(G) \leq \mathrm{bw}(L(G)) + \left\lfloor \frac{G)}{3} \right\rfloor$. The other direction, going from a tree decomposition to a tree embedding, requires seeing that a tree decomposition of a line graph can be made to have a particular structure, specifically that the edges of $L(G)$ corresponding to each vertex of G can be mapped to disjoint subtrees of T. The equality was shown by Harvey and Wood [80] and captures how our characterization of the temporal costs of tensor network contraction relates to earlier characterizations. However, our characterization in terms of tree embeddings, while mathematically equivalent to that in terms of tree decompositions of line graphs, allows for a conceptually cleaner and more fine-grained perspective. We prove the inequalities in Section 2.7.

Theorem 3. *The edge congestion of graph G is at least the branchwidth of its line graph and at most the same plus a third of its maximum degree. Furthermore, a tree embedding with edge congestion $k + \lfloor G)/3 \rfloor$ can be efficiently computed from a branch decomposition of width k and a branch decomposition of width k can be efficiently computed from a tree embedding with edge congestion k.*

For vertex congestion and treewidth (which concern the overlap of subtrees at vertices), the requirement that the mapping be a bijection with the leaves of the tree can be dropped, as can the requirement that the tree be binary. Yet these requirements are without loss of generality, as any tree embedding or tree decomposition can be modified to satisfy these without increasing its vertex congestion or treewidth, respectively. For edge congestion and branchwidth, which concern overlap over edges, the bijection and degree requirements are essential.

The usual definitions for pathwidth and modified cutwidth are in terms of paths (or, equivalently, linear orderings), whereas in (2.1) we allowed them to be caterpillars. This is equivalent, and allows us to relate the properties just discussed with their linear variants. In particular, the relationship between the bubblewidth of a tensor network $(G, \mathcal{M})$ and its "contraction complexity" is almost the same as that between the modified cutwidth of the graph and its vertex congestion, in the sense that the bubblewidth is exactly equal to the cutwidth and $\mathrm{cw}(G) \leq \mathrm{mcw}(G) \leq \mathrm{cw}(G) + G)$.

We can make another analogy, that treewidth : (vertex congestion) :: pathwidth : (modified cutwidth) . For example, [21, 80] $\frac{1}{2}(\mathrm{tw}(G) + 1) \leq \mathrm{vc}(G) + 1 \leq G)(\mathrm{tw}(G) + 1)$ and

$\mathrm{pw}(G) \leq \mathrm{mcw}(G) + 1 \leq G)\,(\mathrm{pw}(G) + 1)$.

The (unmodified) cutwidth is a linear analog to what Ostrovskii called the "tree congestion" of a graph [127]; the tree congestion is the same as the edge congestion except that there is a bijection between *all* the vertices of the binary tree, rather than just the leaves.

2.4 Contraction costs

Our primary motivation is minimizing the time and space costs of tensor network contraction. Ideally, for instances of interest we would like to *provably* minimize the cost, which entails tight lower bounds and the corresponding constructions that meet them. Given the formal hardness of tensor network contraction and the informal hardness of proving lower bounds, we restrict our attention to minimizing the cost of tensor network contraction as it is most commonly done: as a series of matrix multiplications.

First, how much space is required to store a tensor network $(G, \mathcal{M})$? Each tensor $\mathbf{M}_v$ consists of 2^v numbers; this is the main component of the space requirements. Technically, we must also keep track of the graph G and the weights of its edges $E(G)$ as well as a dope vector for each tensor indicating how the tensor is laid out in memory; these will be negligible. Our memory accounting will be in units of whatever is used to store a single entry of a tensor. While in general, the bit depth of an entry may scale non-trivially with instance size, practical implementations will use a fixed-width data type.

Then, what do we need to do a contraction of two tensors? Suppose we want to contract a $(d_\mathrm{L}, d_\mathrm{M})$ tensor with a $(d_\mathrm{M}, d_\mathrm{R})$ tensor along their shared dimension d_M. The input tensors require a total of $d_\mathrm{M}(d_\mathrm{L} + d_\mathrm{R})$ space and the output tensor $d_\mathrm{L} d_\mathrm{R}$. In theory, it should be possible to do the contraction using no more space than that required by the larger of the input tensors and output tensor. In practice, new memory is allocated for the new tensor, it is populated with the appropriate data from the input tensors, and then the memory for the latter is freed. We assume the second cost model, in which memory is simultaneously allocated both for the tensors to be contracted and for the tensor that results from their contraction, but our ideas are straightforwardly modifiable for plausible variants.

The contraction itself is essentially matrix multiplication, and a straightforward implementation will take time $d_\mathrm{L} \cdot d_\mathrm{M} \cdot d_\mathrm{R}$. There exist Strassen-like algorithms for matrix multiplication with better asymptotic runtime, but the constant pre-factors are so large and the straightforward algorithm so heavily optimized that they are of little practical value given the size of currently available machines.

Lastly, in order to implement a tensor contraction as a matrix multiplication, the tensors must be laid out commensurately in memory. If they are not, then the data of one tensor or both must be permuted to make them so. This permutation can effectively be done in place and in linear time. In practice, the permutation time is negligible compared to the matrix multiplication time.

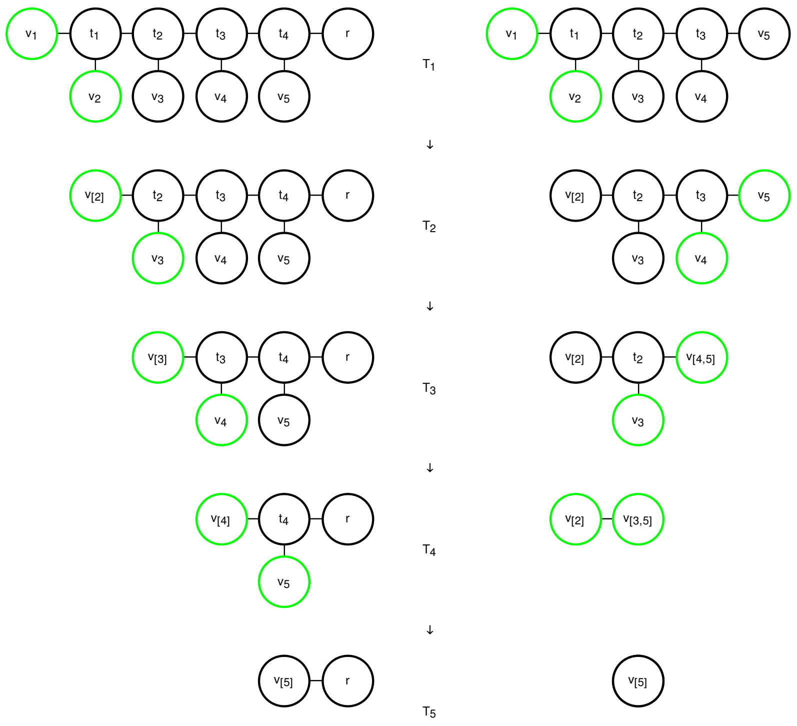

Figure 2.4: Two series of contraction trees (unrooted and rooted on left and right, respectively) for a tensor network with 5 tensors. From top to bottom, the contraction trees for the initial, intermediate, and final tensor networks. The pair of leaves corresponding to the next pair of tensors to be contracted are highlighted in green.

2.5 Contraction orders and trees

In this section, we present our main contribution: a graph-theoretic characterization of the temporal and spatial costs of families of contraction orders.

2.5.1 Linear contraction orders

We start with a special case of contraction orders. Let a *linear contraction order* be one specified by an ordering of the vertices $(v_1, v_2, \ldots, v_n)$. That is, the first contraction is of vertices v_1 and v_2 to form a new vertex $v_{1,2}$. The second contraction is of $v_{1,2}$ and v_3 to form $v_{1,2,3}$, and so on. We represent such a contraction order by what we call a *rooted contraction tree*. The contraction tree of a linear contraction order is a binary caterpillar tree with $n + 1$

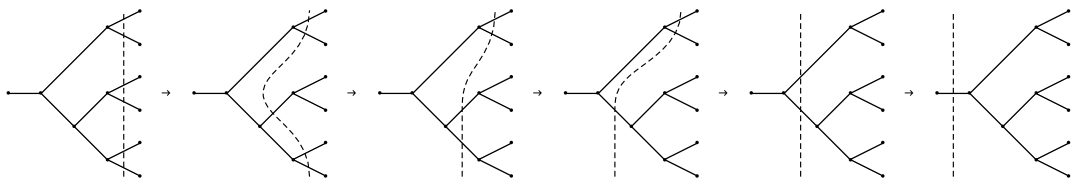

Figure 2.5: The intermediate states of a contraction procedure. The tree pictured is a rooted contraction tree, with the root at the left. The dashed line crosses edges of the contraction tree adjacent to tensors held in memory.

leaves, one for each vertex of the original graph and a special leaf called the *root*, as shown in Figure 2.4. The root leaf is at one of the "ends" of the tree. Each vertex v_i for $i \in [2, n]$ is at distance $n + 2 - i$ from the root, and vertex v_1 is at distance n therefrom. We denote such a contraction tree by (T, b), where T is the tree and $b : V(G) \to V(T) \cup \{r\}$ is the bijection between the vertices of G and the leaves of T together with the root r.

Recall that for a tensor network $(G, \mathcal{M})$, we are using the convention that the weight $w(u, v)$ of an edge $\{u, v\}$ is the logarithm of the bond dimension of wire connecting tensors $\mathbf{M}_u$ and $\mathbf{M}_v$. For each edge $\{u, v\}$ of G there is a unique path in T between $b(u)$ and $b(v)$, which we call a *routing*. Assign the weight $w(u, v)$ to every vertex and edge on this path, including the endpoints $b(u)$ and $b(v)$. We say that the *congestion* of a vertex or edge of T, denoted $\text{con}(v)$ or $\text{con}(e)$, is the sum of the weights of all the routings that include it. Label the non-root leaves of T by $l_i = b(v_i)$ and the internal vertices by t_i for $i \in [n - 1]$, where t_{n-1} is closest to the root and t_1 is farthest. For concision, identify t_0 with t_1.

We now show that these congestions capture the costs of the contraction order. First, note that for each vertex $v \in G$, the congestion $\text{con}(e)$ of the edge $e \in E(T)$ adjacent to $b(v)$ gives the size of the tensor $\mathbf{M}_v$, in the sense that $\text{con}(e) = \sum_{u \in V(G)} w(v, u) = v$, so that $2^{\text{con}(e)}$ is the product of the bond dimensions of the tensor M_v. Now, consider the first contraction, of vertices v_1 and v_2, i.e., tensors $\mathbf{M}_{v_1}$ and $\mathbf{M}_{v_2}$. The bond dimension of the wire between them is $2^{w(v_1,v_2)}$. The product of the bond dimensions of $\mathbf{M}_{v_1}$ with tensors besides v_2 is $2^{\deg_{v_1} - w(u,v)}$, and similarly for $\mathbf{M}_{v_2}$. As discussed in Section 2.4, the contraction can be done in time $2^{\deg_{v_1} - w(u,v)} \cdot 2^{w(u,v)} \cdot 2^{\deg_{v_2} - w(v_1,v_2)} = 2^{w(v_1,v_2,V(G)\setminus\{v_1,v_2\})}$, where $w(v_1, v_2, V(G) \setminus \{v_1, v_2\})$ is the total weight of edges across the tripartite cut. This is exactly the congestion of the vertex $t \in V(T)$ adjacent to both $b(v_1)$ and $b(v_2)$. Suppose that we have done the contraction, yielding a new tensor network containing the contracted vertex $v_{1,2}$. The size of this new tensor $\mathbf{M}_{v_{1,2}}$ is $2^{w(\{v_1,v_2\},V\setminus\{v_1,v_2\})} = 2^{\text{con}(\{t_1,t_2\})}$. If we continue with the contractions, we notice an exciting pattern. We can identify each contraction with an internal vertex of T. The congestion of that vertex gives the time to do the contraction, and the congestion of the adjacent edge nearest the root gives the space of the resulting contracted tensor. The congestion of the leaves, which is equal to the congestions of the adjacent edges and gives the size of the corresponding tensors, can be interpreted as giving the time required to simply read in the tensors of the initial network to be contracted. Overall, the total time of all the contractions is $\sum_{t\in V(T)} 2^{\mathrm{con}(t)} \leq 2n \cdot 2^{\mathrm{vertcon}_{T,b}(G)}$, where $\mathrm{vertcon}_{T,b}(G) = \max_{t\in V(T)} \mathrm{con}(t)$. Furthermore, each edge $e \in E(T)$ corresponds to a tensor $\mathbf{M}_e$ that appears at some point in the series of contractions; those adjacent to leaves correspond to the initial tensors and internal edges to tensors resulting from contractions. The congestion of each edge gives the size of the corresponding tensor, in the sense that the size of $\mathbf{M}_f$ is $2^{\mathrm{con}(f)}$. At any point point in the contraction order, there are at most n tensors, so the required memory is at most $n2^{\mathrm{edgecon}_{T,b}(G)}$, where $\mathrm{edgecon}_{T,b}(G) = \max_{f\in E(T)} \mathrm{con}(f)$. As shown in Section 2.3, the minimum vertex congestion over all linear contraction orders is exactly equal the vertex cutwidth of G. It is closely related to the bubblewidth of earlier work [6], which is exactly equal to the edge cutwidth. The minimum edge congestion over all linear contraction orders is exactly equal to the edge cutwidth of G.

2.5.2 General contraction orders

We now turn our attention to general (i.e., not necessarily linear) contraction orders. The first generalization we make is to remove the root. In other words, for each linear contraction order we form an *unrooted contraction tree* exactly as before except that leaves of T are in unqualified bijection with the vertices of G. This unrooted contraction tree can be interpreted as corresponding to 2^{n-2} different contraction orders in the following way. Define a pair of leaves in a binary tree to be *close* if they are at distance 2. In the caterpillar binary trees we have considered thus far, there are two pairs of close leaves, at each "end" of the tree. Before, we used a rooted caterpillar contraction tree to represent the unique contraction order given by contracting the two non-root close leaves until we got to the root. Now, the unrooted caterpillar contraction tree represents the family of contraction orders that can be specified by contracting *either* pair of a close leaves of the contraction tree until a single vertex remains. Importantly, it remains true that every one of these contraction orders takes time exactly $\sum_{t\in V(T)} 2^{\mathrm{con}(t)} = \tilde{O}\left(2^{\mathrm{vertcon}_{T,b}(G)}\right)$ and space $\tilde{O}\left(2^{\mathrm{edgecon}_{T,b}(G)}\right)$.

The second generalization we make is to remove the restriction to caterpillar trees.

Definition 10. A *rooted contraction tree* (T, b) of a tensor network $(G, \mathcal{M})$ is a rooted binary tree T and a bijection $b : V(G) \to V(T)$ between the vertices (tensors) of G and the (non-root) leaves of T. An *unrooted contraction tree* (T, b) is an unrooted binary tree T and a bijection $b : V(G) \to V(T)$ between the vertices of G and the leaves of T.

An unrooted contraction tree represents a set of contraction orders in the following way. Suppose we have a contraction order $e_1, \dots, e_{n-1}$; Each edge can be written as $e_i = \{v_S, v_{S'}\}$ for some disjoint $S, S' \subset V(G)$, where v_S is the vertex formed by contracting the vertices in S. We start with an empty forest $T_1 = (V(G), \emptyset)$. For each contraction $e_i = \{v_S, v_{S'}\}$, we add a new vertex $v_{S\cup S'}$ to the forest, as well as edges from the new vertex to v_S and $v_{S'}$. That is, $T_i = (V(T_{i-1}) \cup \{v_{S\cup S'}\}, E(T_{i-1}) \cup \{\{v_S, v_{S\cup S'}\}, \{v_{S'}, v_{S\cup S'}\}\})$. For the last contraction, instead of adding a new vertex, we only add an edge between v_S and $v_{S'}$. Doing this yields an unrooted contraction tree for the given contraction order. We say that an unrooted contraction tree represents the set of contraction orders from which it can be constructed in this way. If for the last contraction, we added not only a new vertex v_V connected to v_S and $v_{S'}$ but a second vertex r connected v_V, we would have a rooted contraction tree.

We can easily go the other way as well. Suppose we have a rooted contraction tree. Then we can iteratively build a contraction order. We select an arbitrary close pair of non-root leaves, and add to the contraction order the contraction corresponding to the adjacent internal vertex (of the tree). We then remove the two leaves and their adjacent edges. The internal vertex now becomes a leaf, and corresponds to the vertex resulting from contracting the two vertices (of the tensor network) in the new contraction tree. We repeat until only a single edge of the tree remains, corresponding to the completely contracted tensor network and the root. This is the same procedure visualized in Figure 2.4, except that, when the contraction tree is not restricted to be a caterpillar, there may be many more than two pairs of close leaves to choose from at each step.

Given an unrooted contraction tree, we can turn it into a rooted contraction tree by splitting any edge (i.e., removing an edge, adding a new vertex and adding edges between the new vertex and the vertices adjacent to the removed one), and then adding a root vertex and connecting it to the first newly inserted vertex.

Proof of Theorem 1. In a contraction tree, either rooted or unrooted, each internal vertex corresponds to a contraction. In rooted contraction trees, there is a clear directionality; two of the neighbors are "inputs" and the third is "output". However, the congestion of the vertex, the exponential of which gives the time to do the matrix multiplication, is independent of this directionality. Similarly, each edge of a contraction tree corresponds to a tensor that exists at some point in the contraction (specifically, when the edge is adjacent to a leaf). Again the congestion of this edge is independent of its direction, and the size of the tensor is the exponential of the congestion. Without loss of generality, we prove the theorem using rooted contraction trees.

Suppose we have a rooted contraction tree (T, b) of a tensor network $(G, \mathcal{M})$. Each internal vertex $i \in V(T)$ corresponds to a matrix multiplication, which takes time $2^{\mathrm{vc}(i)}$. Each leaf $l \in V(T)$ corresponds to an initial tensor of size $2^{\mathrm{vc}(l)}$, where $\mathrm{vc}(l) = G\left(b^{-1}(l)\right)$. Overall, the total time then is $\sum_{t \in V(T)} 2^{\mathrm{vc}(t)} \leq 2n2^{\mathrm{vc}(T)}$.

The rooted contraction tree gives a partial ordering of its vertices, which represent contractions (or initial tensors). Any topologically sorted linear ordering $(t_1, t_2, \ldots, t_{2n-2})$ of the vertices of the contraction tree can be considered uniquely as a contraction order consistent with the contraction tree, and vice versa. For a given contraction order, consider the intermediate state at some point in the overall contraction procedure. Let t_i be the last tensor contracted and t_{i+1} the next one to contract. Each edge $f \in E(T)$ from $\{t_1, \ldots, t_i\}$ to $\{t_{i+1}, \ldots, t_{2n-2}\}$ corresponds to a tensor that needs to be stored at this point. The size of the tensor is exactly $2^{\mathrm{ec}(f)}$. The size of the next tensor (resulting from the contraction corresponding to t_{i+1}) is $2^{\mathrm{ec}(f')}$, where f' is the edge from t_{i+1} towards the root of T. Using the convention that the weight of an edge $f \in E(T)$ of T is $w(f) = 2^{\mathrm{ec}(f)}$, then the directed, weighted modified cutwidth of a vertex t_{i+1} in a linear ordering $(t_1, \ldots, t_{2n-2})$ of the vertices of T is exactly equal to the space needed to store the remaining tensors to be contracted and make room for the tensor resulting from the next contraction. Once the contraction is done, the memory allocated for the two tensors that were contracted can be freed. For the coarser space bound, we can just pre-allocate memory for every tensor that will arise during the procedure, in total space $\sum_{f \in E(T)} 2^{\mathrm{ec}(f)} \leq 2n2^{\mathrm{ec}(T)}$.

Overall, if we choose a contraction tree T with minimum vertex congestion, i.e., $\mathrm{vc}(G) = \mathrm{vc}(T) \geq \mathrm{ec}(T)$, we get time at most $n2^{\mathrm{vc}(G)}$ and space at most $n2^{\mathrm{vc}(G)}$. If instead we choose a contraction tree T with minimum edge congestion, i.e., $\mathrm{ec}(G) = \mathrm{ec}(T) \geq (2/3)\mathrm{vc}(T)$, we get time at most $n2^{1.5\mathrm{ec}(G)+1}$ and space at most $n2^{\mathrm{ec}(G)}$. Tightness follows from the fact that for any contraction order, we can construct a rooted contraction tree whose properties give the stated bounds. □

Proof of Theorem 2. Suppose we have a rooted contraction tree (T, b) and that $l^* \in V(T)$ is a leaf on a longest path from a leaf to the root using the vertex weight $w(t) = 2^{\mathrm{vc}(t)}$. Call this path from l^* to the root the *critical path* $P*$. The vertices on $P*$, ordered from the leaf to the root, represent a series of contractions. This series of contractions can be done in time $\sum_{t \in V(P^*)} 2^{\mathrm{vc}(t)}$, the vertex-weighted length of the P^*, which by definition is the longest such path. We prove the claim for general contraction trees by induction. The base case is a tensor network of just two tensors, so that there is just a single contraction and the critical path has 3 vertices. The inductive step is that if the claim is true for a contraction tree whose critical path has k vertices, it is true for a contraction tree whose critical path has $k+1$ vertices. Consider the last vertex t on P^* nearest the root. It corresponds to a contraction of a tensor from an earlier contraction $P*$ and a tensor from the remaining subtree of T, i.e., the part of tree not containing P^*. By definition, the length of the critical path of this subtree is no more than the length of the subpath from l^* to t; otherwise P^* would not be the longest path. Therefore, this subtree can be contracted in less time than the earlier parts of P^*. These can be done in parallel, so the overall time is simply that for P^*. □

As shown in Section 2.7, a branch decomposition of $L(G)$ with width k can be efficiently converted into a contraction tree of G with edge congestion $k + \lfloor G)/3 \rfloor$. Similarly, a tree decomposition of $L(G)$ with width $k-1$ can be efficiently converted into a contraction tree of G with vertex congestion k [80]. Thus one way of utilizing these results is to use an existing algorithm for finding tree decompositions or branch decompositions as a starting point. Theorems 1 and 2 can then be used to construct minimum-cost contraction orders in a more precise way than previous results allow. Developing empirically good implementations of algorithms for finding tree decompositions is a particularly active area research [149]. These are already exploited in much recent work on tensor network contraction [27, 38, 153]. The framework presented here can significantly augment the effectiveness of such techniques. For instances with a lot of structure, as typical ones do, the intuitiveness of contraction trees also empowers manual construction of contraction trees.

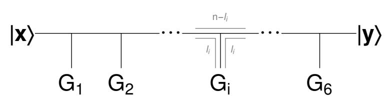

Figure 2.6: Contraction tree representation of the Schrödinger algorithm for computing $\langle x|G_1G_2\cdots G_m|y\rangle$.

There are also techniques for certifying the optimality of tree decompositions and branch decompositions (namely, brambles and tangles) that can be ported to certify the optimality of contraction trees with respect to vertex and edge congestion, respectively. For planar graphs, the exact edge congestion can be computed (non-constructively) in polynomial time [142]. In addition to serving as a lower bound for calculations, the structure of such obstructions may help with understanding the complexity of quantum states as represented by tensor networks.

2.6 Extensions and generalizations

Heretofore, we have assumed that all tensor networks under consideration had no open legs, i.e., that they contract to a single number (0-rank tensor). More generally, we can consider tensor networks with open legs that contract to non-trivial tensors. For such tensors, we treat any open legs as wires to a single "environment" tensor, which we then identify with the root of a rooted contraction tree. For the purposes of minimizing the congestion, the graph will simply have one more vertex. All previous results regarding the costs of contraction then follow exactly as before without modification.

We can also allow tensor networks $(G, \mathcal{M})$ in which G is a hypergraph. Recall how we defined the congestions of a contraction tree (T, b). Each vertex $v \in V(G)$ was identified with a leaf of T through the bijection b. Then each edge $\{u, v\} \in E(G)$ contributed its weight to the congestions of the vertices and edges on the routing (unique path) between $b(u)$ and $b(v)$ in T. For a hyperedge $\{v_1, \ldots, v_k\}$, there is a unique subtree of T connecting the adjacent vertices (which is equal to the union of the paths connecting each pair of edges). Then the hyperedge contributes its weight to the congestions of the vertices and edges on this subtree. The hyperedge corresponds to a so-called "copy" tensor with k legs of the same bond dimension b [16]. The copy tensor is one when all indices have the same value and is zero otherwise. Such a tensor arises, e.g., in a decomposition of a controlled quantum gate.

Decompositions of tensors highlight the main limitation of the present work. While our upper bounds are unconditional, our lower bounds hold only within what we call the matrix multiplication model, in which the only operations allowed are matrix multiplications. This takes advantage only of the topological properties of G and, importantly, not of the properties of the tensor $\mathcal{M}$. However, in many cases of practical interest, the tensors have structure that can be exploited. For example, a tensor corresponding to a quantum gate can be split into two tensors connected by a wire with bond dimension equal to the Schmidt rank across some bipartition of the qubits on which it acts. For gates with less-than-full Schmidt rank, this can help with contraction significantly. Once such a decomposition is made, the sparser graph structure can be exploited by the methods presented here.

Tree-based methods for tensor network contraction are used in state-of-the-art simulations of quantum circuits, where "simulation" here means calculation a single matrix element $\langle \mathbf{x}|C|\mathbf{y}\rangle$ for a pair of basis states $(|\mathbf{x}\rangle, |\mathbf{y}\rangle)$ and the circuit C. In addition to providing a precise analysis of such methods, we can also analyze algorithms not usually expressed in such terms. For example, consider the "Schrödinger" algorithm: a state vector of size 2^n is kept in memory and for each of m gates in sequence. Suppose each gate acts on at most l qubits. Let the circuit be represented as a tensor network with $m+2$ tensors: one for each gate, one for the output $|\mathbf{x}\rangle$, and one for the input $|\mathbf{y}\rangle$. In the corresponding graph G, the vertex $|\mathbf{x}\rangle$ is adjacent to each of the gate vertices that first act on a qubit, with weight equal to the number of qubits that are first acted on by the gate. Similarly for $|\mathbf{y}\rangle$. The Schrödinger algorithm is then a linear contraction order using the vertex ordering $(|\mathbf{x}\rangle, G_1, \ldots, G_m, |\mathbf{y}\rangle)$. Each internal vertex of the contraction tree adjacent to the vertex corresponding to the l_i-local gate G_i has congestion $n + l_i$: $n - l_i$ from the qubits not acted on by the gate, then l_i each from the input and output wires. The total time for the contraction is thus $\sum_{i=1}^{m} 2^{n+l_i} \leq m 2^{n+l} = O(m 2^n)$, where $l = O(1)$ is the maximum locality of a gate. Each internal edge has congestion n, so the contraction can be done using space $O(2^n)$.

An alternative approach is the "Feynman", or path integral, algorithm, which inserts resolutions of the identity after every gate and sums. Now we consider the tensor network corresponding to $\langle \mathbf{x}|C|\mathbf{y}\rangle$ slightly differently. For simplicity, assume all gates are 2-local. Instead of having a single vertex $|x\rangle$ for the input, we have n vertices $|x_i\rangle$, one for each qubit. Similarly, we have n output vertices $\{|y_i\rangle\}$. First, we contract the input vertices $|x_i\rangle$ into the adjacent gate vertices. This leaves $2m$ wires, 2 from each gate to the next or an output. Suppose that instead of contracting the entire tensor network, we remove a single wire and replace it with $|b\rangle\langle b|$ for $b \in \{0,1\}$. The value of the original network is the sum of the values of the reduced networks over $b \in \{0,1\}$. The Feynman algorithm is then to do this for all wires. For each value $\mathbf{b} \in \{0,1\}^{2m}$, we have a tensor network of m tensors and no wires, which we can "contract" in $O(m)$ time and $O(1)$ space. But we need to do this for every $\mathbf{b}$ and sum them up, meaning overall it takes $O(m 4^m)$ time. We need $O(n+m)$ space to keep track of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{b}$. We can generalize this approach to arbitrary tensor networks. First, we remove some set $S \subset E(G)$ of edges, with total weight $W = \sum_{s \in S} w(e)$. There are 2^W values of the corresponding wires, and for each one we contract the reduced tensor network. Let $\tilde{G} = (V, E \setminus S)$ be the reduced network. Overall, for a sequential algorithm, this takes time $\tilde{O}\left(2^{W + \mathrm{vc}(\tilde{G})}\right)$ and space $\tilde{O}\left(W + m 2^{\mathrm{ec}(\tilde{G})}\right)$. Moreover, we consider the cuts S as allowing trivial parallelization, by doing the 2^W contractions of the reduced network in parallel on the same number of processors. This idea was used, for example, by Villalonga et al. to balance time and memory usage in their simulation of grid-based random quantum circuits. Aaronson and Chen [1] show that for carefully chosen cuts that form nested partitions, the contributions W to the time and space from the cuts can be significantly reduced.

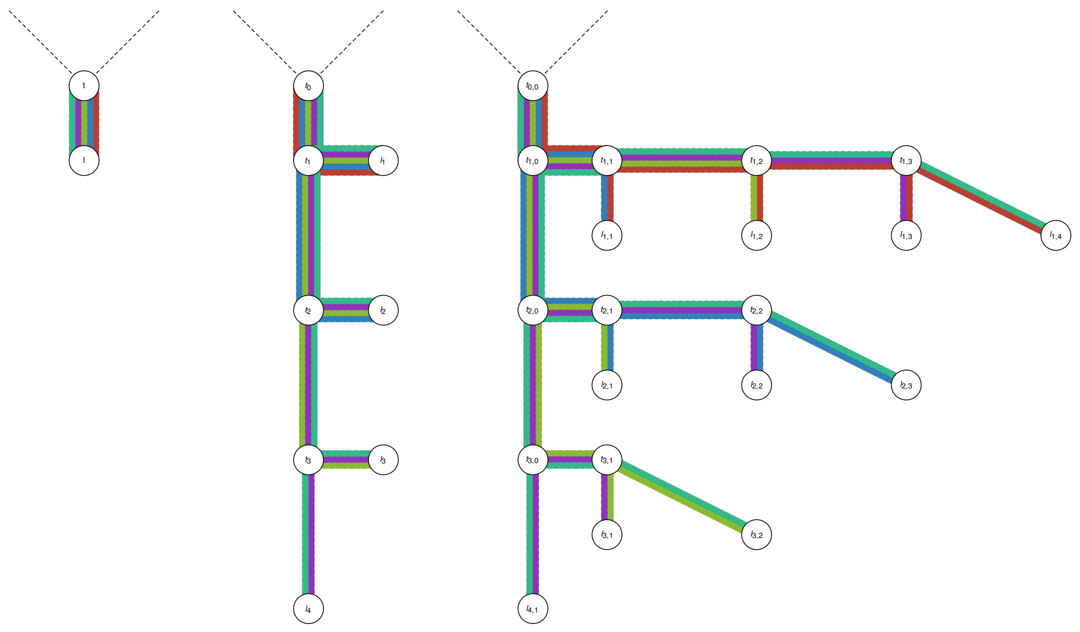

Figure 2.7: From a tree embedding of G to a branch decomposition of $L(G)$. Left: a leaf l_0 and neighboring vertex t_0 of a tree embedding. Middle: Replacement of the leaf l_0 with a caterpillar subtree. Right: Replacement of each leaf with a caterpillar subtree.

2.7 Branchwidth and edge congestion

Proof of Theorem 3. First, we show how to compute a branch decomposition of $L(G)$ with width k given a tree embedding of G with congestion k, implying $\mathrm{bw}(L(G)) \leq \mathrm{ec}(G)$. Suppose we have a tree embedding (T, b) of G with edge congestion k. Let T' be a copy of T and $b' : E(L(G)) \to V(T)$ be a mapping from edges of the line graph to vertices of T. In particular, for an edge $e = \{\{u, v\}, \{v, w\}\}$ of $L(G)$ set $b' : \{\{u, v\}, \{v, w\}\} \mapsto b(v)$, i.e., b' maps adjacent pairs of edges of G to the same leaf mapped to from their common vertex by b. Interpreted as a branch decomposition of $L(G)$, (T', b') has width k, except that b' is not injective. We will now introduce a series of modifications to (T', b') that will turn it into a proper branch decomposition with width k. Note that $b'(e) = b'(f)$ if and only if e and f correspond to the same vertex of G. For each vertex v of G, we will replace the corresponding leaf of T with a subtree whose leaves are one-to-one with the edges of $E(L(G))$ corresponding to the vertex v. Consider a particular vertex v. Let l_0 be the corresponding leaf of T and t_0 its neighbor. Let $(e_1, e_2, \ldots, e_{G(v)})$ be an arbitrary ordering of the edges adjacent to v in G. First, we replace the leaf l_0 with a subcubic caterpillar graph with internal vertices $(t_1, t_2, \ldots, t_{G(v)-2})$ and leaves $(l_1, l_2, \ldots, l_{G(v)-1})$ such that t_i is adjacent to t_{i-1} and l_i for $i \in [G(v) - 2]$ and $t_{G(v)-2}$ is adjacent to $l_{G(v)-1}$. Then we set $b' : \{e_i, e_j\} \mapsto l_{\min(i,j)}$.

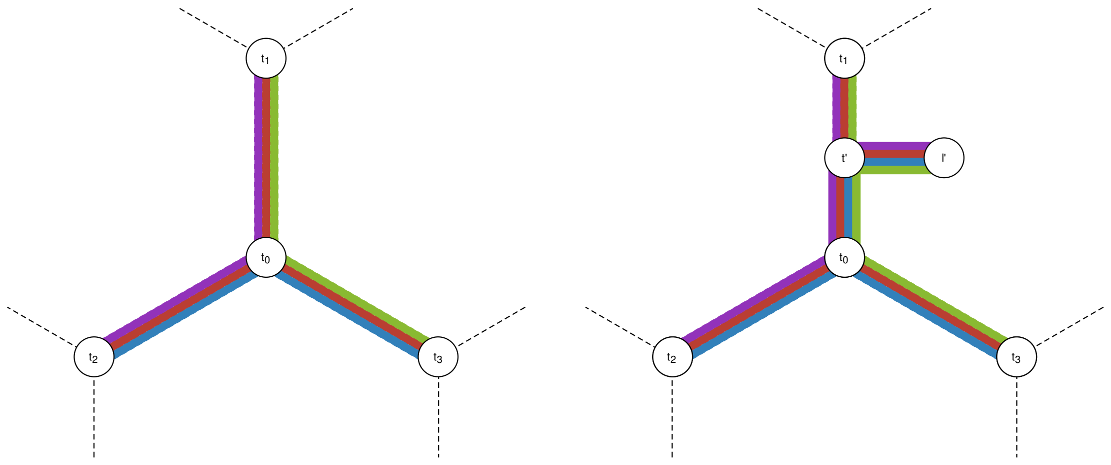

Figure 2.8: From a branch decomposition of $L(G)$ to a tree embedding of G. Left: Part of a branch decomposition. Right: Modified part to form a tree embedding.

At this point $|b'^{-1}(l_i)| = |\{\{e_i, e_j\}|j > i\}| = G(v) - i$. For each l_i we do the following. Relabel its neighbor t_i as $t_{i,0}$. Replace l_i with another subcubic caterpillar graph with internal vertices $\left(t_{i,1}, t_{i,2}, t_{i,G(v)-i-2}\right)$ and leaves $\left(l_{i,1}, l_{i,2}, l_{i,G(v)-i-1}\right)$ such that $t_{i,j}$ is adjacent to $l_{i,j}$ and $t_{i,j-1}$ for $j \in [G(v) - 2]$ and $t_{i,G(v)-2}$ is adjacent to $l_{i,G(v)-1}$. Then set

$$b' : \{e_i, e_j\} \mapsto \begin{cases} l_{i,j-i}, & i < j, \\ l_{j,i-j}, & i > j. \end{cases} \tag{2.2}$$

At this point, (T', b') is a proper branch decomposition of the line graph $L(G)$. What is its width? Let S'_e be the subtree connecting $b'(\{e, f\})$ for all neighbors f of e in $L(G)$. In the part of T' that we didn't change, this coincides with S_e of the tree embedding (T, b). The number of subtrees including the edge $\{t_0, t_{1,0}\}$ of T' is the same as that including the edge $\{t_0, l_0\}$ of T, which is at most the edge congestion of (T, b). In particular, it is exactly $\deg_G(v)$. These are the only subtrees that contain any part of the new parts of the tree T' that we created. The contstruction is shown for a degree 5 vertex in Figure 2.7.

Now, we show how to compute a tree embedding with congestion k from a width-k branch decomposition the line graph, implying $\mathrm{ec}(G) \leq \mathrm{bw}(G)$. Suppose we have a width-k branch decomposition (T, b) of $L(G)$. Let T' be a tree and b' a function from $V(G)$ to $V(T')$. Initially we set $(T', b') = (T, b)$ and iteratively modify it into a tree embedding. For each vertex $v \in V(G)$, the neighboring edges $E_v \subset E(G) = V(L(G))$ form a clique of size $G(v)$. Therefore, there must be some vertex t_0 of T such that S_e contains t_0 for all $e \in E_v$. Let t_1, t_2, t_3 be the three neighbors of t_0 and partition E_v into four (potentially empty) parts: E_0 contains those edges e such that S_e contains all of t_0, t_1, t_2, t_3 and E_i contains those edges e such that S_e does not contain t_i, for $i = 1, 2, 3$. Without loss of generality, assume $w(E_1) \leq w(E_2) \leq w(E_3)$.

Note that $v) = \sum_{i=0}^{4} w(E_i) \geq \sum_{i=1}^{4} w(E_i) \geq 3w(E_1)$. Now, subdivide the edge between t_0 and t_1, introducing a new vertex t', and add a new leaf l' adjacent thereto. For all $e \in E_v$, set $b'(e) = l'$; this leaf will correspond to vertex v in the tree embedding. Note that the congestion of the edge between l' and t' is $v)$, and that the congestion of the edge between t' and t_0 is $w(E_1) \leq v)/3$ more than the congestion of the edge $\{t_0, t_1\}$ that it replaced. If we do this for every vertex, we get a tree embedding whose congestion is at most $G)/3$ more than the width of the branch decomposition we started with. This is illustrated in Figure 2.8. $\square$

It cannot be the case that for every graph G, $\mathrm{bw}(L(G)) = \mathrm{ec}(G)$. Consider, for example, the star graph S_k. Its edge congestion is at least its maximum degree k, but its line graph is the complete graph, whose branchwidth is $\left\lceil \frac{2k}{3} \right\rceil$.

Consider an alternative, what we'll call the *line hypergraph*, denoted $L^*(G)$, with a vertex for each edge of $E(G)$ and a *hyperedge* for each vertex of $V(G)$ (rather than a clique as in the usual line graph). Then it is trivially true that $\mathrm{bw}(L^*(G)) = \mathrm{ec}(G)$.

Chapter 3

Computational complexity of electronic structure

This chapter reproduces Ref. 124 with minor modification. Finding the ground state energy of electrons subject to an external electric field is a fundamental problem in computational chemistry. We prove that this electronic-structure problem, when restricted to a fixed single-particle basis and fixed number of electrons, is QMA-complete. Schuch and Verstraete have shown hardness for the electronic-structure problem with an additional site-specific external *magnetic* field, but without the restriction to a fixed basis [141]. In their reduction, a local Hamiltonian on qubits is encoded in the site-specific magnetic field. In our reduction, the local Hamiltonian is encoded in the choice of spatial orbitals used to discretize the electronic-structure Hamiltonian. As a step in their proof, Schuch and Verstraete show a reduction from the antiferromagnetic Heisenberg Hamiltonian to the Fermi-Hubbard Hamiltonian. We combine this reduction with the fact that the antiferromagnetic Heisenberg Hamiltonian is QMA-hard [131] to observe that the Fermi-Hubbard Hamiltonian on generic graphs is QMA-hard, even when all the hopping coefficients have the same sign. We then reduce from Fermi-Hubbard by showing that an instance of Fermi-Hubbard can be closely approximated by an instance of the Electronic-Structure Hamiltonian in a fixed basis. Finally, we show that estimating the energy of the lowest-energy Slater-determinant state (i.e., the Hartree-Fock state) is NP-complete for the Electronic-Structure Hamiltonian in a fixed basis.

3.1 Introduction

Simulating quantum mechanical systems is one of the most important computational challenges in modern science. Solving this problem, broadly defined, will allow us to probe the foundations of physics, chemistry, and materials science, and will have useful applications to a wide variety of industries. On the other hand, the very properties that make quantum mechanical systems so interesting – such as the exponential growth of the underlying state space and quantum entanglement – also make quantum simulation a particularly difficult computational task.

Finding the means to tame this daunting complexity is an objective that is nearly as old as quantum mechanics itself. Paul Dirac, in a foundational paper from 1929, asserted that "The fundamental laws necessary for the mathematical treatment of a large part of physics and the whole of chemistry are thus completely known, and the difficulty lies only in the fact that application of these laws leads to equations that are too complex to be solved. It therefore becomes desirable that approximate practical methods of applying quantum mechanics should be developed, which can lead to an explanation of the main features of complex atomic systems without too much computation" [55].

Today, nearly a century later, Dirac's quote captures the underlying motivation for a large body of quantum science research. For example, in the context of simulating systems of many electrons, the complexity inherent in the simulation problem has been addressed by approximation methods such as Hartree-Fock and Density Functional Theory [83], as well as by considering simplified quantum models such as the Hubbard and Heisenberg Hamiltonians [9]. Moreover, even in a new, exciting era in which noisy, intermediate-scale quantum computers are being developed that may be well suited to solve certain quantum simulation problems, Dirac's wisdom prevails. Existing quantum algorithms, such as the phase estimation algorithm and the variational quantum eigensolver all obtain approximate solutions to special cases of the quantum simulation problem (see, e.g., [99, 130, 133]).

However, there are also fundamental limitations to these simulation algorithms that stem from quantum computational complexity. Kitaev, building on the classical work of Cook and Levin, proved that a very general quantum simulation problem (approximating the ground state energy of a k-local Hamiltonian) is QMA-complete [45, 98]. QMA is a natural quantum analogue of NP, and QMA-complete problems should not have an efficient quantum algorithm, for essentially the same reasons that NP-complete problems (such as boolean satisfiability) should not have efficient classical algorithms.

Nonetheless, QMA-completeness should not be interpreted as a categorical roadblock, but rather as an important guidepost for the development of future quantum algorithms. In the same way that many practically interesting instances of classical constraint-satisfaction problems have special structural properties that avoid the worst-case hardness implied by NP-completeness results, we study QMA-completeness in order to understand which structural properties reduce the complexity of the simulation problem – and which properties do not – enabling improved quantum simulation algorithms that could potentially exploit this structure.

In this work our goal is understand the computational difficulty of simulating systems of interacting elections. Our main result shows that when restricted to a fixed number of electrons and a fixed single-particle basis, approximating the ground state energy of the electronic structure Hamiltonian is QMA-complete. This can be interpreted as a direct sharpening of Dirac's quote: we conclusively demonstrate that these properties *do not* add enough structure to enable the existence of an efficient quantum simulation algorithm to approximate the ground state energy of such systems.

Section 3.2 gives an overview of our results and techniques. Section 3.3 gives the reduction from the antiferromagnetic Heisenberg Hamiltonian to the Fermi-Hubbard Hamiltonian.

Section 3.4 shows the reduction from Fermi-Hubbard to the Electronic Structure Hamiltonian in a fixed basis. In Section 3.5 we show that finding the lowest energy Hartree-Fock state for the Electronic Structure Hamiltonian in a fixed basis is NP-complete. Some of the technical lemmas used to show the QMA-hardness of the Electronic Structure Hamiltonian in a fixed basis are given in Section 3.6.

3.2 Overview of Results

3.2.1 Formalizing the electronic structure problem

In computational complexity, finding the ground state of a local Hamiltonian acting on qudits is the canonical QMA-complete problem. Local Hamiltonians are interesting not only because they are analogous to classical constraint-satisfaction problems involving a set of low-arity functions, but because physical Hamiltonians are typically local due to the nature of physical forces. The computational complexity of finding ground states of qubit Hamiltonians has been studied extensively, and hardness shown even for Hamiltonians that are physically realistic in the sense, e.g., that the terms are placed on a 2D lattice or all the same up to a positive rescaling [131]. Much less is known about the hardness of local Hamiltonians for indistinguishable particles. In this work, we consider the local Hamiltonian problem for fermionic systems.

The local Hamiltonian problem for systems of indistinguishable particles has two distinctive features. First, the Hamiltonians themselves are invariant under permutations of the particles. Second, the goal is to estimate the lowest-energy of a symmetric (for bosons) or anti-symmetric (for fermions) state. Generic Hamiltonians (i.e., quartic polynomials in the elementary operators with general coefficients) for both types of indistinguishable particles have been shown to be QMA-complete [106, 159], but, as with Hamiltonians on distinguishable particles, we can ask how hard more physically realistic classes of Hamiltonians are. Physically realistic Hamiltonians on indistinguishable particles have special properties that could make them more amenable to computing ground energies. In particular, here we are focused on the computational complexity of the *electronic structure Hamiltonian*

$$H^{(\mathrm{ES})} = -\frac{1}{2}\sum_i \nabla_i^2 + \sum_i V(\mathbf{r}_i) + \frac{1}{2}\sum_{i\neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}, \tag{3.1}$$

which acts on an anti-symmetric state $\psi \in \mathbb{R}^{\eta\times 3}$ of η electrons, where $\mathbf{r}_i$ is the position of the i-th electron in 3-dimensional space. For η electrons and a specified electric potential $V : \mathbb{R}^3 \to \mathbb{R}$, this is the Hamiltonian dictated by the laws of electromagnetism. Of particular interest in chemistry is the *molecular electronic structure Hamiltonian*, in which the external potential

$$V(\mathbf{r}) = -\sum_j \frac{Z_j}{|\mathbf{r} - \mathbf{R}_j|} \tag{3.2}$$

is that of nuclei modelled as classical point particles, each with positive charge Z_j and located at fixed position $\mathbf{R}_j$. In reality, the nuclei are also quantum particles, but they are so much more massive than the electrons that this model (the *Born-Oppenheimer* approximation) is usually a sufficiently accurate approximation to the Hamiltonian of a molecule specified by the nuclear charges and number of electrons. There is a separate optimization procedure to find the lowest-energy configuration of nuclear positions.

Physically, the wavefunction of the electrons is over continuous real space. Computationally, we need to discretize the space of possible wavefunctions in some way in order to have a finite representation of a potential ground state. This leads to the fundamental computational problem of quantum chemistry, estimating the ground state energy of the *electronic structure Hamiltonian in a fixed basis*:

$$H^{(\mathrm{ES})}(\boldsymbol{\phi}, V) = T + V + U = \sum_{\substack{i,j\in[n]\\ \sigma\in\{\pm1\}}} t_{i,j} a_{i,\sigma}^\dagger a_{j,\sigma} + \sum_{\substack{i,j\in[n]\\ \sigma\in\{\pm1\}}} v_{i,j} a_{i,\sigma}^\dagger a_{j,\sigma} + \frac{1}{2}\sum_{\substack{i,j,k,l\in[n]\\ \sigma,\tau\in\{\pm1\}}} u_{i,j,k,l} a_{i,\tau}^\dagger a_{j,\sigma}^\dagger a_{k,\sigma} a_{l,\tau} \tag{3.3}$$

where $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)$ is the single-particle basis with elements $\phi_i : \mathbb{R}^3 \to \mathbb{C}$ and

$$t_{i,j} = -\frac{1}{2}\int d\mathbf{r}\phi_i^*(\mathbf{r})\nabla^2\phi_j(\mathbf{r}), \tag{3.4}$$

$$v_{i,j} = \int d\mathbf{r}\phi_i^*(\mathbf{r})V(\mathbf{r})\phi_j(\mathbf{r}), \tag{3.5}$$

$$u_{i,j,k,l} = \int d\mathbf{r}d\mathbf{s}\phi_i^*(\mathbf{r})\phi_j^*(\mathbf{s})\frac{1}{|\mathbf{r}-\mathbf{s}|}\phi_k(\mathbf{s})\phi_l(\mathbf{r}). \tag{3.6}$$

The indices $i, j, k, l \in [n]$ index spatial orbitals, and $\sigma, \tau \in \{\pm1\}$ indicate the spin. Given the potential $V(\mathbf{r})$ and a fixed set of orbitals, the Hamiltonian shown in (3.3) is then completely determined by the integrals for the kinetic and potential energy shown in Eqs. (3.4) to (3.6). The *Electronic Structure* problem then is to determine whether the ground energy of the resulting Hamiltonian is less than some threshold E or greater than $E + 1/\mathsf{poly}(n)$. This is the version of the problem posed by Whitfield et al. ([160]), who left its hardness as an open problem. We answer here in the affirmative by showing a family of single-particle bases (with zero potential $V(\mathbf{r}) = 0$) that encodes hard problems. In $H^{(\mathrm{ES})}$, the σ and τ indicate the sign of the spin of the spin orbital. For each spatial orbital $\phi_i(\mathbf{r})$, there are two spin orbitals $\phi_{i,\pm1}(\mathbf{r})$. We use ±1 as an index for simplicity, but of course physically the electron's spin has magnitude 1/2.

Definition 11 (Electronic structure in fixed basis set – ESFBS)**.** An instance of electronic structure in a fixed basis set is specified by an external electric field $V : \mathbb{R}^3 \to \mathbb{R}$, a number η of electrons, a basis set $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)$, and thresholds $a < b$, where $b - a \geq 1/\mathsf{poly}(\eta)$. The external potential V and the basis set $\boldsymbol{\phi}$ must be specified concisely (using $\mathsf{poly}(n)$ bits) in a way that allows for efficient ($\mathsf{poly}(n)$-time) calculation of the integrals in Eqs. (3.4) to (3.6). The goal is to determine whether the ground state energy of H_{ES} in the subspace of η electrons spanned by the given basis is at most a or at least b.

Our definition of the problem allows for states with arbitrary total spin, and this freedom will be critical in our construction. One can also consider a variant in which the total spin is fixed, analogous to, for example, the XY problem with fixed magnetization. Our definition of the problem also restricts the states allowed to a finite-dimensional space spanned by a set of fixed single-electron orbitals. By necessity, this is the form of the problem solved in practice by computational chemists. However, for practical purposes, it is desirable that the ground state or ground energy in the chosen basis is close to that in a complete, infinite-dimensional basis. The difference between these two is known as the basis-set error, and bases are typically chosen in order to minimize this error. The basis we use in our construction is artificial in this sense; in the absence of an external potential, there is nothing to to confine the electrons to the subspace of $\mathbb{R}^3$ spanned by the basis. However, the orbitals that we use are still superpositions of Gaussians, a commonly used form in computational chemistry, e.g. the STO-3G basis set [82] with each basis function composed of a fixed superposition of three primitive Gaussians. Indeed, we prove the following theorem that the electronic structure in a fixed basis is QMA-hard by encoding a QMA-hard Hamiltonian in the construction of the basis.

Theorem 4 (ESFBS is QMA-complete, informal)**.** The electronic structure problem in a fixed-basis set and at fixed particle number is QMA-complete.

Our results contribute to a large body of work formally establishing the computational intractability of increasingly physically realistic Hamiltonians. There are still many important problems in computational chemistry whose computational complexity is unknown. For example, even in a fixed basis, does fixing the spin make the problem easier? Does the problem become more tractable if the given orbitals are guaranteed to have small basis-set error? Is the electronic structure problem hard in a complete (infinite-dimensional) basis? If so, is it still hard when the external potential arises solely from a set of positively charged nuclei at fixed positions? We pose two variants of the electronic structure problem whose hardness is an open question. In both cases, the "size" of the problem is the number of electrons.

Electronic structure in a fixed basis with bounded basis-set error: Given an external electric potential V, number of electrons η, thresholds $a < b$, and a basis set $\boldsymbol{\phi}$ with basis-set error $\epsilon(\eta) = 1/\mathsf{poly}(\eta)$ for the given potential V, determine whether the lowest energy of a state in the space spanned by $\boldsymbol{\phi}$ is at most a or greater than $b = a + 1/\mathsf{poly}(\eta)$. The basis-set error is defined as $\langle\tilde{\psi}|H|\tilde{\psi}\rangle - \min_{|\psi\rangle\in\mathbb{R}^{\eta\times 3}} \langle\psi|H|\psi\rangle \leq \epsilon(\eta)$.

The parameters of the *problem* are the promised basis-set error bound $\epsilon(\eta)$, the thresholds $a(\eta)$ and $b(\eta)$, and the family of potentials considered (as a function of η); an instance is specified by simply the number of electrons η, the potential V, and a specification of the basis set $\boldsymbol{\phi}$. This variant entertains the possibility that, while the problem is hard for arbitrary *bases*, it may always be easy for *good* bases (in the sense of having low basis-set error). In practice, chemists always want to use a good basis, and often do, though in general they have no guarantees on the error of the bases that they use. Note that a good basis need not necessarily be complete for the whole space; all that matters is that its span includes a state sufficiently close to the ground state. For example, in Schuch and Verstraete's construction for the QMA-hardness of electronic structure with magnetic fields, the external electric potential V implies a good basis of size $n = \eta$ that captures the ground state but is far from complete. Theoretical and numerical results suggest that for physically realistic external potentials there is a always good basis of size $\mathsf{poly}(\eta)$ [79, 96], though the constant prefactors may be impractically large. Furthermore, there may exist pathological external potentials for which no polynomially large good basis exists.

To account for both the possibility of no good polynomially large basis and the desirability of working in a small basis, we define another variant of the problem that includes finding the basis in which the state is expressed. The formulation attempts to be as general as possible while remaining in QMA. Ideally, we would like to consider all states that can be efficiently represented and whose energy can be efficiently estimated by a quantum computer. To formalize this, we specify some family of parameterized orbitals in which the putative low-energy state can be expressed. For example, the family of bases could consist of all weighted sums of Gaussians. In this case the prover would provide, for each basis element, the centers, the weights, and the exponents of the constituent Gaussians.

Electronic structure in parameterized basis: Given an external electric potential V, number of electrons η, thresholds $a < b$, and a family of basis functions $\{\phi_\theta\}_\theta$, and basis size k, determine whether there exists a basis $\boldsymbol{\phi} = (\phi_{\theta_1}, \dots, \phi_{\theta_k})$ such that the lowest energy of a state in the space spanned by $\boldsymbol{\phi}$ is at most a or greater than $b = a + 1/\mathsf{poly}(\eta)$.

The problem is parameterized by the thresholds $a(\eta)$ and $b(\eta)$ and the family of basis functions $\{\phi_\theta\}_\theta$ allowed; an instance is specified by just the number of electrons η, basis set size k, and potential V. A certificate consists of the classical description of orbitals $\boldsymbol{\phi}$ and a quantum state on $2k$ qubits that is supposed to represent a low-energy of state of η electrons in the basis $\boldsymbol{\phi}$.

Other variants of the electronic structure problem have been considered. Schuch and Verstraete show QMA-hardness for electronic structure with an additional site-specific magnetic field *in a good basis*, which is used to encode an instance of a QMA-hard problem [141]. Their result is thus incomparable to ours; we removed the magnetic field, but also the restriction to a good basis.

There are several related computational problems concerning various ways of representing and working with quantum states. One is N-representability. Note that the 2-electron reduced density matrices (2-RDMs) of the quantum state encode all the information necessary to compute the energy of the electronic structure. The N-representability problem is to determine whether or not for a given set of 2-RDMs there exists a consistent quantum state on the full space. This problem has been shown, under Turing reductions, to be QMA-complete [106]. Another related problem is Density Functional Theory (DFT), which is premised on the fact that the electron density (i.e. the average number of electrons at each point in space) is also sufficient to calculate the energy of the electronic structure Hamiltonian. That is, there exists a universal functional that takes as input the electron density and outputs the energy. However, while such a functional exists, it may not be computationally efficient. Indeed, computing it has been shown to be QMA-hard, also under Turing reductions [161]. For both N-representability and DFT, it remains an open question whether they remain hard when the inputs (2-RDMs and electron densities, respectively) are restricted to the ground states of electronic structure Hamiltonians. Broadbent and Grilo recently proved [35] QMA-completeness of the Consistency of Local Density Matrices problem (i.e., the qudit analog of N-representability) under Karp reductions, but left as an open question whether or not their techniques can be used to show QMA-hardness under Karp reductions of N-representability and the universal functional of DFT.

3.2.2 Implications for VQE and other algorithms

In Chapter 4, we introduce and employ the Variational Quantum Eigensolver (VQE), a variational quantum algorithm for estimating the ground state energy of quantum Hamiltonians, typically applied specifically to electronic structure Hamiltonians. The immediate implication of our QMA-hardness result is that VQE cannot succeed in polynomial time for all instances of the electronic structure problem. VQE is really a meta-algorithm; a particular instantiation entails choosing an ansatz (parameterized set of states) and method for optimizing over that ansatz. There are two potential obstacles to VQE's success: limitations of the ansatz and limitations of the optimization method, with a trade-off between them. If the ansatz is universal (capable of approximating any quantum state with polynomial circuit complexity), then VQE is essentially a model of QCMA, with a classical witness (the parameters) verified by a quantum computer (which prepares the state using the given parameters). Assuming $\mathsf{QMA} \neq \mathsf{QCMA}$, then VQE cannot solve the electronic structure problem in general, even given arbitrary time to find the parameters. Recent work [20] has shown that optimizing the parameters of generic fermionic variational circuits is NP-hard. Our results imply that this must be the case even for electronic structure Hamiltonians, even when the ansatz is simply Slater determinants.

These two obstacles exemplify two aspects of Hamiltonian complexity (how "quantum" the witness is and how hard it is to find) and thus two ways in which quantum computation can be helpful. Importantly, our results only show worst-case hardness. Whether the above implications hold for the more restricted Hamiltonians and ansatzes used in practice remains an open question.

3.2.3 Hubbard Hamiltonians

The proof that Electronic Structure in a Fixed Basis Set (ESFBS) is QMA-hard proceeds in two stages. We first reduce from the antiferromagnetic Heisenberg Hamiltonian to the Fermi-

Hubbard Hamiltonian. Then we reduce from Fermi-Hubbard to ESFBS. This subsection gives an overview of the first reduction.

The Bose-Hubbard and Fermi-Hubbard Hamiltonians are:

$$H^{(\mathrm{BH})} = \sum_{i \in V} U n_i(n_i - 1) + \sum_{\{i,j\} \in E} t_{i,j} \left(b_i^\dagger b_j + \text{h.c.} \right), \quad H^{(\mathrm{FH})} = \sum_{i \in V} U n_{i,+} n_{i,-} + \sum_{\{i,j\} \in E, \sigma \in \{\pm\}} t_{i,j} a_{i,\sigma}^\dagger a_{j,\sigma}, \tag{3.7}$$

where $G = (V, E)$ is the interaction graph, and $a_i a_j^\dagger + a_j^\dagger a_i = b_i b_j^\dagger - b_j^\dagger b_i = \delta_{ij}$. When we refer to the "Hubbard" model without qualification, we mean the *Fermi*-Hubbard model, in which the particles are fermions. Hubbard Hamiltonians are of practical interest because they approximate Hamiltonians of many more complicated condensed-matter and chemical systems. Their solutions are taken to qualitatively describe those of the approximated systems.

Childs et al. [39, 40] show that the Bose-Hubbard Hamiltonian and XY Hamiltonian are QMA-hard with uniform coefficients. In both cases, because the coefficients are uniform, the instance is encoded entirely in the graph, which does not seem embeddable in, say, three spatial dimensions, as we would want for a physically realistic Hubbard Hamiltonian. Schuch and Verstraete [141] show as an intermediate result that the Fermi-Hubbard Hamiltonian on a 2D lattice with a site-specific *magnetic field* is QMA-hard; the instance is encoded entirely in this magnetic field. We show that the magnetic field is not necessary, at the cost of having an arbitrary weighted interaction graph.

Theorem (FH is QMA-complete)**.** The Fermi-Hubbard Hamiltonian with arbitrary coefficients and fixed particle number is QMA-complete, even if all of the tunneling coefficients have the same sign and are bounded by a polynomial in the number of particles.

The proof reduces from the antiferromagnetic Heisenberg Hamiltonian:

$$H^{(\mathrm{Heis})} = \sum_{\{i,j\} \in E} \kappa_{i,j} W_{i,j}, \qquad W = (II + XX + YY + ZZ)/2, \tag{3.8}$$

which is known to be QMA-hard [51, 131]. As in related previous constructions, we fix the number of particles to equal the number of spatial orbitals, i.e., half the number of spin orbitals. The large onsite-repulsion term U penalizes two electrons occupying the same spatial orbital, and so the ground space of the repulsion term has exactly one electron in each spatial orbital. As was done in [106], the spin of the electron in each orbital encodes a logical qubit. With the repulsion term dominating the Hamiltonian, we treat the rest perturbatively. To second order, this yields an antiferromagnetic Heisenberg Hamiltonian on the same graph as a Hubbard Hamiltonian. We go between a qubit Hamiltonian and a fermionic Hamiltonian using the Jordan-Wigner transformation $a_i \leftrightarrow \prod_{j<i} Z_j (X_i + iY_i)/2$. In general, this transforms local fermionic Hamiltonians into non-local qubit Hamiltonians, but with a particular ordering of the spin orbitals, the parity strings $\prod_{j<i} Z_j$ cancel out. In our case, this yields the local Heisenberg Hamiltonian.

3.2.4 Overview of Techniques for Electronic structure

We reduce from an instance of Fermi-Hubbard. The interaction graph has an edge for every pair of fermions with a non-zero interaction term. Given an input Hamiltonian of this form, we create a set of orbitals corresponding to the vertices in the interaction graph. ϕ_i is the orbital for vertex i. Each ϕ_i is a superposition of what we call *primitive* orbitals, which are just Gaussians centered at various points in space. For the most part, these points are spaced out from all the other points by a parameter Γ, which is set to be large. For every edge $\{i, j\}$ in the interaction graph of the Fermi-Hubbard Hamiltonian, there is a pair of primitive orbitals, one in ϕ_i and one in ϕ_j, such that the two primitive orbitals are a distance $\gamma_{i,j}$ apart. The $\gamma_{i,j}$'s are small compared to Γ.

The dominant term that emerges from this construction is the kinetic energy between two Gaussians with exponent α that are separated by a distance of $\gamma_{i,j}$ (with a slight correction due to the fact that the Gaussians are not exactly pairwise orthogonal). Each distance $\gamma_{i,j}$ can then be tuned to obtain the desired coefficient to encode the Fermi-Hubbard Hamiltonian. Each orbital also includes a primitive orbital with exponent $\beta > \alpha$ in order to increase the onsite-repulsion term, ensuring that the ground space for the effective Hamiltonian has exactly one electron per spatial orbital. Thus, each of our orbitals has the form

$$\phi_i(\mathbf{r}) = 2^{-1/2}\phi_{i,0}(\mathbf{r}) + (2d)^{-1/2}\sum_{i=1}^{d}\phi_{i,l}(\mathbf{r}), \tag{3.9}$$

where each $\phi_{i,l}$ is a Gaussian and the parameter d is an upper bound on the degree of the graph. The functions $\phi_{i,0}$ all have some large exponent β and are therefore more concentrated than the functions $\phi_{i,l}$ for $l > 0$ which have a smaller exponent α.

We use two approximation steps which ultimately show that the electronic structure Hamiltonian $H^{(\mathrm{ES})}$ closely approximates the Fermi-Hubbard Hamiltonian $H^{(\mathrm{Hubb})}$.

$$H^{(\mathrm{ES})} \overset{3.4.2}{\to} H^{(\mathrm{round})} \overset{3.4.3}{\to} H^{(\mathrm{main})} \overset{3.4.4}{\propto} H^{(\mathrm{Hubb})}, \tag{3.10}$$

Each step introduces some small error, the bounding of which constitutes the bulk of the technical work in our proof.

The transition from $H^{(\mathrm{ES})}$ to $H^{(\mathrm{round})}$ includes two approximation steps. The first approximation arises from the fact that the orbitals $\boldsymbol{\phi}$ that we use are not perfectly orthonormal. However, there is an orthonormal basis $\tilde{\boldsymbol{\phi}}$ that is very close to $\boldsymbol{\phi}$. We show that the difference is sufficiently small that we can proceed with the coefficients from the nonorthonormal basis but using the elementary operators of the orthonormal basis. There is one exception to this approximation: the overlap of the Gaussians that are relatively close (distance $\gamma_{i,j}$ apart) has a non-negligible effect and requires a slight correction to the corresponding kinetic energy coefficient. In the second approximation step, we drop the interactions of primitive orbitals that are at least a distance of Γ apart, resulting in an expression with many fewer terms. The effect of applying both approximations results in the Hamiltonian $H^{(\mathrm{round})}$. The transition from $H^{(\mathrm{round})}$ to $H^{(\mathrm{main})}$ involves dropping the potential-energy terms that involve more than one primitive orbital. The difference between $H^{(\text{round})}$ and $H^{(\text{main})}$ is an energy offset which is constant for a fixed number of electrons plus an error term which we bound in the proof. We then show that the parameters can be set so that the coefficients of $H^{(\text{main})}$ approximate the Fermi-Hubbard model to within any inverse polynomial.

3.2.5 Product states

Classical algorithms for finding the ground state energy of quantum Hamiltonians are often limited by the fact that the ground state seems to have no concise classical description. For that reason, chemists often try to find the lowest-energy Slater determinant, known as the Hartree-Fock state. Within a fixed basis $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)$, a Slater determinant is a state of the form

$$|\mathrm{SD}(B)\rangle = b_1^\dagger b_2^\dagger \cdots b_\eta^\dagger |\mathbf{0}\rangle , \tag{3.11}$$

where each $b_i = \sum_{j=}^{n} B_{i,j} a_j$ is a sum of annihilation operators in the original basis and the rows of the $\eta \times n$ matrix B are orthonormal.

Definition 12 (Lowest-energy Slater determinant (LESD))**.** Given a local fermionic Hamiltonian in a fixed basis of size n, number η of electrons, and bounds $b > a$, where $b - a = 1/\mathsf{poly}(n)$, determine whether the lowest-energy Slater determinant has energy at most a or at least b. The Slater determinant is specified the matrix $\eta \times n$ B with entries specified by polynomially many bits.

Theorem 5 (informal)**.** LESD for electronic structure Hamiltonians ($H^{(\text{ES})}$ as defined in 3.3) is NP-complete.

Schuch and Verstraete showed that the LESD problem for generic quartic number-preserving fermionic Hamiltonians is NP-hard [141, arXiv version]. We show NP-hardness for the restricted class of such Hamiltonians with coefficients implied by a basis and external potential as in Eqs. (3.3) to (3.6); that is, our Theorems 4 and 5 cover the same class of electronic structure Hamiltonians and differ only in the class of states to be optimized over. Schuch and Verstraete's proof for the QMA-hardness of electronic structure with magnetic fields could likely be extended to the NP-hardness of the Slater determinant version, but neither we nor they have done so.

3.2.6 A Note on Notation

By $[n]$, we mean the set $\{1, 2, \ldots, n\}$. We use $\|\cdot\|$ for the spectral norm of a matrix and the Euclidean norm of a vector. We use $|\cdot|$ for the element-wise scalar norm.

3.3 Fermi-Hubbard Model

We will show that the version of the Fermi-Hubbard Hamiltonian problem described below is QMA-complete. In order for the Fermi-Hubbard model to approximate the antiferromagnetic

Heisenberg from which we are reducing, we need a large onsite-repulsion term u_0 to penalize orbitals with double occupancy. The Fermi-Hubbard problem remains QMA-complete for any u_0 that satisfies the lower bound in the theorem stated below. For the reduction from Fermi-Hubbard to Electronic Structure, we require that the $t^{(\text{Hubb})}_{i,j}$ coefficients are bounded by a polynomial in n, the number of electrons. The hardness result that we prove establishes that Fermi-Hubbard remains hard, even under that constraint.

Theorem 6 (QMA-completeness of Hubbard Hamiltonian with uniform onsite repulsion)**.** There exist constants $p > q > 0$ such that for all $u_0^{(\text{Hubb})} \geq n^{14+3p+2q}$, determining to precision n^{-q} the ground state energy in the n-particle subspace of a Hubbard Hamiltonian

$$H^{(\text{Hubb})} = u_0^{(\text{Hubb})} \sum_{i \in [n]} n_{i,+1} n_{i,-1} + \sum_{\substack{i<j \\ \sigma \in \{\pm 1\}}} t^{(\text{Hubb})}_{i,j} \left(a^\dagger_{i,\sigma} a_{j,\sigma} + a^\dagger_{j,\sigma} a_{i,\sigma} \right) \tag{3.12}$$

subject to $\left| t^{(\text{Hubb})}_{i,j} \right| \leq \sqrt{n^p u_0^{(\text{Hubb})}}$ is QMA-complete.

We will reduce from the antiferromagnetic Heisenberg Hamiltonian problem:

Definition 13 (Antiferromagnetic Heisenberg Hamiltonian)**.** An instance of antiferromagnetic Heisenberg Hamiltonian is defined by an edge-weighted graph $G = (V, E)$ with $\kappa : E \mapsto \mathbb{R}_{\geq 0}$ as

$$H^{(\text{Heis})}(G, w) = \sum_{\{i,j\} \in E} \kappa_{i,j} \left(X_i X_j + Y_i Y_j + Z_i Z_j \right) \tag{3.13}$$

We will require in our reduction that the coefficients $\kappa_{i,j}$ are bounded by a polynomial in the number of qubits. Although not explicitly stated, the following theorem is proven in [131].

Theorem 7 (QMA-completeness of antiferromagnetic Heisenberg Hamiltonian [131])**.** Finding the ground state of an antiferromagnetic Heisenberg Hamiltonian is QMA-complete even when restricted to families of Hamiltonians in which the coefficients are bounded by a polynomial in the number of qubits.

To prove Theorem 6, we show that for sufficiently large $u_0^{(\text{Hubb})}$, the Hubbard model approximates an antiferromagnetic Heisenberg model up to second order in perturbation theory.

We'll treat $U^{(\text{Hubb})} = u_0^{(\text{Hubb})} \sum_i n_{i,+1} n_{i,-1}$ as the penalty term and $T^{(\text{Hubb})} = H^{(\text{Hubb})} - U^{(\text{Hubb})}$ as the perturbation. To convert the fermionic Hamiltonians above to qubit Hamiltonians, we use the Jordan-Wigner transform with the ordering $(1, +1)$, $(1, -1)$, $(2, +1)$, $(2, -1)$, ... For the full Hilbert space $\mathcal{H}$ we'll use a basis of one qubit per spin orbital. For the ground space $\mathcal{H}_0$ of $U^{(\text{Hubb})}$, we'll use a basis of one qubit per spatial orbital, the latter spanning the half-filled subspace of the corresponding pair of spin orbitals. We associate the occupancy of the orbitals of spin $+1$ and -1 with the qubit states $|0\rangle$ and $|1\rangle$, respectively. Let Π_0 be the projector onto $\mathcal{H}_0$ and $\Pi_1 = I - \Pi_0$ the projector onto the orthogonal subspace.

In $\mathcal{H}_0$, $U^{(\text{Hubb})}$ is zero ($U_0^{(\text{Hubb})} = \Pi_0 U^{(\text{Hubb})} \Pi_0 = 0$), and outside it is at least $u_0^{(\text{Hubb})}$. In the half-filling regime, the ground space of $U^{(\text{Hubb})}$ is spanned by those basis states having exactly one electron in each spatial orbital. In $\mathcal{H}_0$, $T^{(\text{Hubb})}$ vanishes. In the notation below, we use a bit to indicate whether an orbital is filled. For edge $\{i, j\}$, the first two bits correspond to orbitals $\phi_{i,+1}$ and $\phi_{i,-1}$ and the last two bits correspond to $\phi_{j,+1}$ and $\phi_{j,-1}$. So the state $|0110\rangle$ has $\phi_{i,-1}$ and $\phi_{j,+1}$ filled. The "excitation" terms are

$$T_{1,0}^{(\text{Hubb})} = \Pi_1 T^{(\text{Hubb})} \Pi_0 = \sum_{\{i,j\}\in E} (-1)^{j-i-1} t_{i,j}^{(\text{Hubb})} [(|1100\rangle + |0011\rangle)(\langle 1001| - \langle 0110|)]_{i,j}. \tag{3.14}$$

With this, using Theorem 8 from the next section, we get

$$H^{(\text{eff})} = -T_{0,1}^{(\text{Hubb})} \left[U_1^{(\text{Hubb})}\right]^{-1} T_{1,0}^{(\text{Hubb})} = \sum_{\{i,j\}\in E} \frac{2\left(t_{i,j}^{(\text{Hubb})}\right)^2}{u_0^{(\text{Hubb})}} (W_{i,j} - 1) = c_{\text{eff}} + \sum_{\{i,j\}\in E} h_{i,j}^{(\text{eff})} W_{i,j}, \tag{3.15}$$

where

$$h_{i,j}^{(\text{eff})} = 2\frac{\left(t_{i,j}^{(\text{Hubb})}\right)^2}{u_0^{(\text{Hubb})}}, \qquad c_{\text{eff}} = -\frac{1}{u_0^{(\text{Hubb})}} \sum_{\{i,j\}\in E} \left(t_{i,j}^{(\text{Hubb})}\right)^2. \tag{3.16}$$

3.3.1 Perturbation Theory

We will use the following formulation of second-order perturbation theory, adapted from a special case of the more general formulation by Bravyi et al. [31].

Theorem 8. [Second-order perturbation theory] Consider a Hamiltonian $H = H^{(\text{pen})} + H^{(\text{pert})}$. Let Π_0 be the projector onto the ground space of $H^{(\text{pen})}$, and $\Pi_1 = 1 - \Pi_0$. Define

$$H^{(\text{eff})} = +H_0^{(\text{pert})} - H_{0,1}^{(\text{pert})} \left(H^{(\text{pen})}\right)^{-1} H_{1,0}^{(\text{pert})}, \tag{3.17}$$

where $A_i = \Pi_i A \Pi_i$ and $A_{i,j} = \Pi_i A \Pi_j$. If $H_0^{(\text{pen})} = 0$ and $H_1^{(\text{pen})} \geq \Delta \geq 2H^{(\text{pert})}$, then

$$\left\| H_{\text{low}} - H^{(\text{eff})} \right\| \leq O\left(\frac{\left\|H^{(\text{pert})}\right\|^3}{\Delta^2}\right), \tag{3.18}$$

where H_{low} is the projection of H onto its eigenspace with eigenvalues at most $\Delta/2$.

3.3.2 Fermi-Hubbard is QMA-Hard

Proof of Theorem 6. There are constants $p, q \geq 0$ such that it is QMA-hard to find the ground state energy to precision n^{-q} of

$$H^{(\mathrm{Heis})} = \sum_{\{i,j\} \in E} \kappa_{i,j} W_{i,j} \tag{3.19}$$

subject to $0 \leq \kappa_{i,j} \leq n^p$. Consider such an instance. We want to choose $u_0^{(\mathrm{Hubb})}$ and $t_{i,j}^{(\mathrm{Hubb})}$ such that

$$H^{(\mathrm{Heis})} = H^{(\mathrm{eff})} - c_{\mathrm{eff}} \tag{3.20}$$

and

$$\left\| H_{\mathrm{low}}^{(\mathrm{Hubb})} - H^{(\mathrm{eff})} \right\| = o(n^{-q}). \tag{3.21}$$

The first constraint, Eq. (3.20), is

$$\kappa_{i,j} = h_{i,j}^{(\mathrm{eff})} = 2 \frac{\left(t_{i,j}^{(\mathrm{Hubb})}\right)^2}{u_0^{(\mathrm{Hubb})}} \tag{3.22}$$

or

$$t_{i,j}^{(\mathrm{Hubb})} = \pm \sqrt{u_0^{(\mathrm{Hubb})} \kappa_{i,j} / 2}. \tag{3.23}$$

Therefore, for any $\kappa_{i,j}$ such that $|\kappa_{i,j}| \leq n^p$ we can choose $t_{i,j}^{(\mathrm{Hubb})}$ such that $\left| t_{i,j}^{(\mathrm{Hubb})} \right| \leq \sqrt{n^p u_0^{(\mathrm{Hubb})}}$ and that Eq. (3.20) is satisfied. To satisfy the second constraint, Eq. (3.21), we use second-order perturbation theory (Theorem 8).

Furthermore, the assumption that $u_0^{(\mathrm{Hubb})} \geq n^{14+3p+2q}$ implies that the condition of Theorem 8 is met:

$$\begin{aligned}
\left\| T^{(\mathrm{Hubb})} \right\| &\leq \sum_{\substack{\{i,j\} \in E \\ \sigma \in \{\pm 1\}}} \left| t_{i,j}^{(\mathrm{Hubb})} \right| && (3.24) \\
&\leq \underbrace{n^2}_{\{i,j\},\sigma} \cdot \underbrace{\sqrt{u_0^{(\mathrm{Hubb})} n^p}}_{t_{i,j}^{(\mathrm{Hubb})}} && (3.25) \\
&= \sqrt{u_0^{(\mathrm{Hubb})}} \sqrt{n^{4+p}} && (3.26) \\
&\leq \sqrt{u_0^{(\mathrm{Hubb})}} \cdot \frac{1}{2} \sqrt{n^{14+3p+2q}} \qquad n \geq 2; p, q \geq 0 && (3.27) \\
&\leq \sqrt{u_0^{(\mathrm{Hubb})}} \cdot \frac{1}{2} \sqrt{u_0^{(\mathrm{Hubb})}} = \frac{1}{2} u_0^{(\mathrm{Hubb})}. && (3.28)
\end{aligned}$$

Theorem 8 then yields

$$\left\| H^{(\mathrm{Hubb})}_{\mathrm{low}} - H^{(\mathrm{eff})} \right\| \leq O\left(\frac{\left\| T^{(\mathrm{Hubb})} \right\|^3}{\left(u_0^{(\mathrm{Hubb})}\right)^2} \right) \leq O\left(\frac{\overbrace{n^6 n^{1.5p} \left(u_0^{(\mathrm{Hubb})}\right)^{1.5}}^{(3.25)}}{\left(u_0^{(\mathrm{Hubb})}\right)^2} \right) \tag{3.29}$$

$$= O\left(\frac{n^6 n^{1.5p}}{\sqrt{u_0^{(\mathrm{Hubb})}}} \right) \leq O\left(\frac{n^6 n^{1.5p}}{\sqrt{n^{14+3p+2q}}} \right) = O\left(n^{-(q+1)}\right) = o\left(n^{-q}\right). \tag{3.30}$$

□

3.4 Electronic structure

Here we prove our main result:

Theorem 4 (QMA-completeness of electronic structure in fixed basis set)**.** Determining the ground state energy of an electronic structure Hamiltonian in a fixed basis set and with fixed particle number to inverse-polynomial precision is QMA-complete.

We'll start by defining a set of n spatial orbitals. Once the orbitals are fixed, the t and u coefficients are determined by the integrals in (3.4) and (3.6), which then yields the physical Hamiltonian

$$H^{(\mathrm{ES})} = T + U = \sum_{\substack{i,j \in [n] \\ \sigma \in \{\pm 1\}}} t_{i,j} a^\dagger_{i,\sigma} a_{j,\sigma} + \frac{1}{2} \sum_{\substack{i,j,k,l \in [n] \\ \sigma,\tau \in \{\pm 1\}}} u_{i,j,k,l} a^\dagger_{i,\sigma} a^\dagger_{j,\tau} a_{k,\tau} a_{l,\sigma} \tag{3.31}$$

in the absence of any external potential ($V = 0$). We show that, when restricted to the subspace with exactly n electrons (with arbitrary spin), this yields an effective Hamiltonian that is close, up to rescaling and shifting, to a Fermi-Hubbard Hamiltonian

$$H^{(\mathrm{Hubb})} = u_0^{(\mathrm{Hubb})} \sum_{i \in [n]} n_{i,+1} n_{i,-1} + \sum_{i<j} t^{(\mathrm{Hubb})}_{i,j} \left(a^\dagger_{i,\sigma} a_{j,\sigma} + a^\dagger_{j,\sigma} a_{i,\sigma} \right) \tag{3.32}$$

with where there are constants $p > q > 0$ such that $u_0^{(\mathrm{Hubb})} \geq n^{14+3p+2q}$ and $\left| t^{(\mathrm{Hubb})}_{i,j} \right| \leq \sqrt{n^p u_0^{(\mathrm{Hubb})}}$ for all edges $\{i, j\}$.

3.4.1 Orbitals

Recall from Section 3.2.4 that our goal is a basis of orbitals such that the electronic structure Hamiltonian in that basis is sufficiently close to a Hubbard Hamiltonian. We define here a set of orbitals that effectively encodes the interaction graph of the Hubbard Hamiltonian. Each orbital represents a vertex of the interaction graph and consists of a superposition of Gaussians centered at various points in space. For the most part, these Gaussians are far apart from each other. If two vertices are connected by an edge, then their corresponding orbitals have two Gaussians that are relatively close to each other. This distance between the Gaussians can be tuned to match the interaction coefficient in the Hubbard Hamiltonian. Let

$$\xi_\alpha(\mathbf{r}) = \left(\frac{2\alpha}{\pi}\right)^{3/4} \exp\left(-\alpha \left\|\mathbf{r}\right\|^2\right) \tag{3.33}$$

be the Gaussian centered at $\mathbf{0} \in \mathbb{R}^3$ with exponent $\alpha > 0$. Each of our orbitals will be a superposition of Gaussians. The centers of these Gaussians will be a set of points $\{\mathbf{x}_{i,l}\}_{i,l}$ in $\mathbb{R}^3$, where $i \in [n]$ and $l \in \{0\} \cup [d]$, and $d \leq n-1$ is an upper bound on the maximum degree of the interaction graph G. Note that although the points are in $\mathbb{R}^3$, the properties we require of them can be satisfied by placing them all along a line; we'll use an arrangement in the 2-dimensional plane for convenience. We require two properties of this set of points:

1. For each edge $\{i, j\}$ in the interaction graph, there is exactly one pair $(l, l') \in [d]^2$ such that $\|\mathbf{x}_{i,l} - \mathbf{x}_{j,l'}\| = \gamma_{i,j} > 0$. Let $\gamma_{\min}$ and $\gamma_{\max}$ be lower and upper bounds on $\gamma_{i,j}$ over $\{i, j\} \in E$.

2. Every other pair of points is at least $\Gamma \gg \gamma_{\max}$ apart (in Euclidean distance).

The important part is the $m = |E|$ pairs of points such that points from different pairs are at least a distance of Γ apart. Each pair of points is associated with an edge $\{i, j\}$ in the interaction graph. The pair of points associated with edge $\{i, j\}$ will be $\gamma_{i,j}$ apart, where $\Gamma \gg \gamma_{i,j}$. In addition, there is a set X of $(d+1)n - 2m$ points each of which is a distance at least Γ from any other point in the construction. The points associated with vertex i in the interaction graph are as follows:

1. $x_{i,0}$ is a point from X.

2. If $p \leq i$) and j is the p-th neighbor of vertex i, then $x_{i,p}$ will be one of the points from the pair associated with edge $\{i, j\}$. (The other point from the pair will belong to vertex j.)

3. If $p > i$), then $x_{i,p}$ is a point from X. (These are just dummy neighbors to ensure that all of the orbitals have the same form.)

With these points, we can define the *primitive* orbitals

$$\phi_{i,p}(\mathbf{r}) = \begin{cases} \xi_\beta\left(\mathbf{r} - \mathbf{x}_{i,0}\right) & p = 0, \\ \xi_\alpha\left(\mathbf{r} - \mathbf{x}_{i,p}\right), & \text{otherwise}, \end{cases} \tag{3.34}$$

where α and β are positive constants to be set later. Ultimately, we will need $\beta \gg \alpha$. The *composite* orbitals that we'll use in the construction will be superpositions of these primitive orbitals:

$$\phi_i(\mathbf{r}) = \frac{1}{\sqrt{2}}\phi_{i,0}(\mathbf{r}) + \frac{1}{\sqrt{2d}}\sum_{l=1}^{d}\phi_{i,l}(\mathbf{r}). \tag{3.35}$$

It will be convenient to be able to refer to the indices of the primitive orbitals that are a distance $\gamma_{i,j}$ apart, corresponding to edge $\{i, j\}$. Define $\mathcal{B}(i, j) = \{(i, p), (j, q)\}$, where j is the p-th neighbor of i and i is the q-th neighbor of j.

We will eventually show that the kinetic energy terms between the primitive orbitals that are separated by only a distance of $\gamma_{i,j}$ will be the dominant terms in the Hamiltonian (besides the onsite-repulsion). We will then tune the $\gamma_{i,j}$ distances so that the coefficients resulting from kinetic energy integrals scale with the $t^{(\text{Hubb})}_{i,j}$ from Eq. (3.32), which are the coefficients in the Fermi-Hubbard Hamiltonian from which we are reducing. The radius β will be chosen to be large enough so that the potential energy coefficients $u_{i,i,i,i}$ effectively result in a $u_0 n_{i,+1} n_{i,-1}$ with a large coefficient u_0.

The construction is illustrated in Figure 3.1 with a small example. The orbitals are strictly positive everywhere, so the overlap $\int d\mathbf{r}\phi_i^*(\mathbf{r})\phi_j(\mathbf{r})$ cannot be exactly zero, but we will show that it's very close. That is, we will show that the orbitals are not perfectly orthonormal, but that they are sufficiently close. For now, we'll proceed as if they are, and address the effect of the nonorthonormality in Section 3.4.2. The purpose of including the $\phi_{i,0}$ component as part of the orbital, which is far away from every other primitive orbital center, is to decouple the scale of the onsite-repulsion term in the Hamiltonian from that of interaction term, which are effected by the one of the components $\phi_{i,l}$ for each orbital ϕ_i. To this end, we will ultimately set $\beta \gg \alpha$. Including the other components $\{\phi_{i,l'}\}_{l' > i)}$ is simply to ease the analysis by making all of the orbitals $\{\phi_i\}_i$ have integrals, over single-electron operators, that are of approximately the same form.

3.4.1.1 Integrals of Operators over Gaussians

Since the composite orbitals are superpositions of primitive orbitals, the expressions for overlap, kinetic energy, and potential energy for the composite orbitals will be linear combinations of the corresponding expression for combinations of primitive orbitals. The following integrals of operators over Gaussians will be useful in expressing these terms for the primitive orbitals.

The overlap of two Gaussians with exponents α and β with centers $\mathbf{x}$ apart:

$$s_{\alpha,\beta}(\|\mathbf{x}\|) = \int d\mathbf{r}\xi_\alpha(\mathbf{r})\xi_\beta(\mathbf{r} - \mathbf{x}) = \left(\frac{2\sqrt{\alpha\beta}}{\alpha + \beta}\right)^{3/2} \exp\left(-\frac{\alpha\beta}{\alpha + \beta}\|\mathbf{x}\|^2\right). \tag{3.36}$$

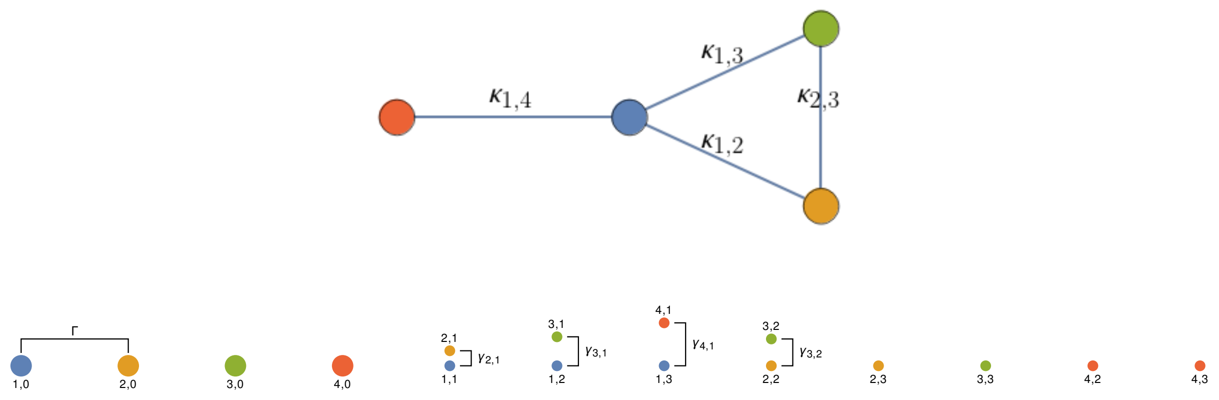

Figure 3.1: The top figure is the interaction graph of a Heisenberg Hamiltonian. The bottom figure shows a possible placement of the primitive orbitals. Larger points represent Gaussians with radius β. Smaller points represent Gaussians with radius α. The orbitals are color coded according to which vertex they belong to from the interaction graph, and thus which composite orbital they contribute to.. For example, the orbital associated with the blue vertex in the interaction graph would be a superposition of the blue Gaussians in the bottom figure. The amplitude of the large blue Gaussian on the left is $1/\sqrt{2}$. The amplitude of each of the smaller blue Gaussians is $1/\sqrt{2d}$, where here $d = 3$.

Due to the rotational invariance of the Gaussians, all of the functions defined in this subsection depend only on the magnitude of their argument, and so we will write, for example, $s\left(\|\mathbf{x}\|\right)$.

We can define the $n(d+1) \times n(d+1)$ matrix S of overlap between the primitive orbitals, where each row and column is indexed by a pair (i,p) corresponding to a primitive orbital:

$$s_{(i,p),(j,q)} = -\frac{1}{2}\int d\mathbf{r}\phi^*_{(i,p)}(\mathbf{r})\phi_{(j,q)}(\mathbf{r}) \tag{3.37}$$

Note that $s_{\alpha,\beta}(\|\mathbf{x}\|)$ denotes the overlap of primitive orbital $\phi_{i,0}$ (whose exponent is β) and $\phi_{j,p>0}$ (whose exponent is α), where $\phi_{i,0}$ and $\phi_{j,p}$ are separated by a distance $\|\mathbf{x}\|$: $s_{(i,0),(j,p)} = s_{\alpha,\beta}(\|\mathbf{x}\|)$. The overlap of two primitive orbitals with the same exponent are denoted by:

$$s_\alpha(\|\mathbf{x}\|) = s_{\alpha,\alpha}(\|\mathbf{x}\|) = \exp\left(-\alpha\left\|\mathbf{x}\right\|^2/2\right), \tag{3.38}$$

$$s_\beta(\|\mathbf{x}\|) = s_{\beta,\beta}(\|\mathbf{x}\|) = \exp\left(-\beta\left\|\mathbf{x}\right\|^2/2\right). \tag{3.39}$$

Therefore $s_{(i,0),(j,0)} = s_\beta(\|\mathbf{x}\|)$, where primitive orbitals $\phi_{(i,0)}$ and $\phi_{(j,0)}$ are a distance $\|\mathbf{x}\|$ apart. Also, $s_{(i,p),(j,q)} = s_\alpha(\|\mathbf{x}\|)$, where $p,q > 0$ and primitive orbitals $\phi_{(i,p)}$ and $\phi_{(j,q)}$ are a distance $\|\mathbf{x}\|$ apart.

The kinetic energy between two Gaussians with exponents α and β with centers $\mathbf{x}$ apart is:

$$t_{\alpha,\beta}(\|\mathbf{x}\|) = -\frac{1}{2}\int d\mathbf{r}\xi_\alpha(\mathbf{r})\nabla^2\xi_\beta(\mathbf{r}-\mathbf{x}) \tag{3.40}$$

$$= 2^{3/2}\frac{(\alpha\beta)^{7/4}}{(\alpha+\beta)^{5/2}}\left(3-2\mu\|\mathbf{x}\|^2\right)\exp\left(-\mu\|\mathbf{x}\|^2\right) \leq \frac{3}{2}\max\{\alpha,\beta\}, \tag{3.41}$$

where $\mu = \alpha\beta/(\alpha+\beta)$, with

$$t_\alpha(\|\mathbf{x}\|) = t_{\alpha,\alpha}(\|\mathbf{x}\|) = \frac{\alpha}{2}\left(3-\alpha\|\mathbf{x}\|^2\right)\exp\left(-\alpha\|\mathbf{x}\|^2/2\right), \tag{3.42}$$

$$t_\beta(\|\mathbf{x}\|) = t_{\beta,\beta}(\|\mathbf{x}\|) = \frac{\beta}{2}\left(3-\beta\|\mathbf{x}\|^2\right)\exp\left(-\beta\|\mathbf{x}\|^2/2\right). \tag{3.43}$$

Define T to be the $n(d+1)\times n(d+1)$ matrix of kinetic energy terms between primitive orbitals. An entry of matrix T is

$$t_{(i,p),(j,q)} = -\frac{1}{2}\int d\mathbf{r}\phi^*_{(i,p)}(\mathbf{r})\nabla^2\phi_{(j,q)}(\mathbf{r}). \tag{3.44}$$

Therefore, $t_{(i,0),(j,0)} = t_\beta(\|\mathbf{x}\|)$, where primitive orbitals $\phi_{(i,0)}$ and $\phi_{(j,0)}$ are a distance $\|\mathbf{x}\|$ apart. Also, $t_{(i,p),(j,q)} = t_\alpha(\|\mathbf{x}\|)$, where $p,q>0$ and primitive orbitals $\phi_{(i,p)}$ and $\phi_{(j,q)}$ are a distance $\|\mathbf{x}\|$ apart.

The potential integrals:

$$u_\alpha^{(\text{Coul})}(\|\mathbf{x}\|) = \int d\mathbf{r}d\mathbf{s}\xi_\alpha(\mathbf{r})^2\xi_\alpha(\mathbf{s}-\mathbf{x})^2\|\mathbf{r}-\mathbf{s}\|^{-1} = \sqrt{\frac{4\alpha}{\pi}}F_0\left(\alpha\|\mathbf{x}\|^2\right) \leq 2\sqrt{\alpha}, \tag{3.45}$$

$$u_\alpha^{(\text{exch})}(\|\mathbf{x}\|) = \int d\mathbf{r}d\mathbf{s}\xi_\alpha(\mathbf{r})\xi_\alpha(\mathbf{r}-\mathbf{x})\xi_\alpha(\mathbf{s})\xi_\alpha(\mathbf{s}-\mathbf{x})\|\mathbf{r}-\mathbf{s}\|^{-1} = \exp\left(-\alpha\|\mathbf{x}\|^2\right)u_\alpha^{(\text{Coul})}(0), \tag{3.46}$$

$$u_\alpha^{(\text{other})}(\|\mathbf{x}\|) = \int d\mathbf{r}d\mathbf{s}\xi_\alpha(\mathbf{r})^2\xi_\alpha(\mathbf{s})\xi_\alpha(\mathbf{s}-\mathbf{x})\|\mathbf{r}-\mathbf{s}\|^{-1} = \exp\left(-\alpha\|\mathbf{x}\|^2/2\right)u_\alpha^{(\text{Coul})}(\mathbf{x}/2), \tag{3.47}$$

where

$$F_k(x) = \int_0^1 e^{-xt^2}t^{2k}dt \tag{3.48}$$

is the Boys function of order k. The analogous definitions for $u_\beta^{(\text{other})}(\mathbf{x})$, $u_\beta^{(\text{other})}(\mathbf{x})$, and $u_\beta^{(\text{other})}(\mathbf{x})$ use Gaussians with exponent β. The potential energy terms on the primitive orbitals are represented by an $n^2(d+1)^2 \times n^2(d+1)^2$ matrix U, where each row and column is indexed by a pair of primitive orbitals $[(i,p),(j,q)]$. The entry in row $[(i,p),(j,q)]$ and column $[(k,r),(l,s)]$ is the potential energy term for orbitals $\phi_{(i,p)}(\mathbf{r})$, $\phi_{(j,q)}(\mathbf{r})$, $\phi_{(k,r)}(\mathbf{r})$, and $\phi_{(l,s)}(\mathbf{r})$:

$$u_{[(i,p),(j,q)],[(k,r),(l,s)]} = \int d\mathbf{r}d\mathbf{s}\frac{\phi^*_{(i,p)}(\mathbf{r})\phi^*_{(j,q)}(\mathbf{s})\phi_{(k,r)}(\mathbf{s})\phi_{(l,s)}(\mathbf{r})}{|\mathbf{r}-\mathbf{s}|}. \tag{3.49}$$

The definitions of the integral functions above correspond to the situation where all four indices (i,p), (j,q), (k,r), and (l,s) denote at most two distinct orbitals with the same exponent. Specifically, for $p,q>0$, where $\|\mathbf{x}\|$ is the distance between $\phi_{(i,p)}$ and $\phi_{(j,q)}$:

$$u_{[(i,p),(j,q)],[(j,q),(i,p)]} = u_\alpha^{(\mathrm{Coul})}(\mathbf{x}), \tag{3.50}$$

$$u_{[(i,p),(j,q)],[(i,p),(j,q)]} = u_\alpha^{(\mathrm{exch})}(\mathbf{x}), \tag{3.51}$$

$$u_{[(i,p),(i,p)],[(i,p),(j,q)]} = u_\alpha^{(\mathrm{other})}(\mathbf{x}). \tag{3.52}$$

3.4.2 Orthonormalizing and rounding

Having constructed our orbitals, we now make two approximations to get a clean, "round" Hamiltonian $H^{(\mathrm{round})}$. First, the orbitals we defined in Eq. (3.35) are slightly nonorthonormal, and so we derive a related orthonormalized basis in which the electronic structure Hamiltonian doesn't change too much. Second, we remove contributions to the Hamiltonian from the electron-electron interaction pairs of primitive orbitals that are far ($>\Gamma$) away from each other. The error of these approximations is quantified by Lemma 1.

The matrix S is defined in (3.37) to be the overlap matrix of the *primitive* orbitals. We can construct a set of orthonormal primitive orbitals by setting:

$$\tilde{\phi}_{i,k}(\mathbf{r}) = \sum_{j,l} \left[S^{-1/2}\right]_{(j,l),(i,k)} \phi_{j,l}(\mathbf{r}) \tag{3.53}$$

and new orthonormal composite orbitals

$$\tilde{\phi}_i(\mathbf{r}) = \frac{1}{\sqrt{2}}\tilde{\phi}_{i,0}(\mathbf{r}) + \frac{1}{\sqrt{2d}}\sum_{l=1}^{d}\tilde{\phi}_{i,l}(\mathbf{r}) \tag{3.54}$$

with annihilation operators $\tilde{a}_{i,\sigma}$ [83]. The Hamiltonian in this orthonormal basis is

$$H^{(\mathrm{ES})} = T^{(\mathrm{ES})} + U^{(\mathrm{ES})} = \sum_{\substack{i,j\in[n]\\ \sigma\in\{\pm1\}}} \tilde{t}_{i,j}\tilde{a}^\dagger_{i,\sigma}\tilde{a}_{j,\sigma} + \frac{1}{2}\sum_{\substack{i,j,k,l\in[n]\\ \sigma,\tau\in\{\pm1\}}} \tilde{u}_{i,j,k,l}\tilde{a}^\dagger_{i,\sigma}\tilde{a}^\dagger_{j,\tau}\tilde{a}_{k,\tau}\tilde{a}_{l,\sigma} \tag{3.55}$$

where

$$\tilde{t}_{i,j} = \int d\mathbf{r}\tilde{\phi}^*_i(\mathbf{r})T\tilde{\phi}_j(\mathbf{r}), \tag{3.56}$$

$$\tilde{u}_{i,j,k,l} = \int d\mathbf{r}d\mathbf{s}\tilde{\phi}^*_i(\mathbf{r})\tilde{\phi}^*_j(\mathbf{s})U\tilde{\phi}_k(\mathbf{s})\tilde{\phi}_l(\mathbf{r}). \tag{3.57}$$

The matrices $\tilde{T}$ and $\tilde{U}$ showing the kinetic and potential energies using the orthonormalized primitive orbitals are analogous to the definitions (3.44) and (3.49). The pair T and $\tilde{T}$ and the pair U and $\tilde{U}$ are related by conjugation by $S^{-1/2}$:

$$\tilde{T} = S^{-1/2}\; T\; S^{-1/2}, \tag{3.58}$$

$$\tilde{U} = (S^{-1/2}\otimes S^{-1/2})\; U\; (S^{-1/2}\otimes S^{-1/2}). \tag{3.59}$$

Since $S \approx I$, the coefficients $\tilde{t}$ and $\tilde{u}$ for the orthonormalized orbitals are close to t and u for the non-orthonormalized orbitals, but the difference needs to be carefully bounded. We will approximate $H^{(\mathrm{ES})}$ by the Hamiltonian $H^{(\mathrm{round})}$ that uses creation and annihilation operators of the orthonormal basis $\tilde{\boldsymbol{\phi}}$ with the original coefficients, subject to two modifications. First, we add a first-order correction to the off-diagonal kinetic coefficients. Second, we remove contributions from pairs of primitive orbitals that are at least Γ apart (which makes many terms vanish completely).

The rounded Hamiltonian is

$$H^{(\mathrm{round})} = T^{(\mathrm{round})} + U^{(\mathrm{round})}. \tag{3.60}$$

The rounded kinetic operator is

$$T^{(\mathrm{round})} = t_{i,i}^{(\mathrm{round})} \sum_{\substack{i\in[n] \\ \sigma\in\{\pm1\}}} \tilde{n}_{i,\sigma} + \sum_{\substack{\{i,j\}\in E \\ \sigma\in\{\pm1\}}} t_{i,j}^{(\mathrm{round})} \left(\tilde{a}_{i,\sigma}^{\dagger}\tilde{a}_{j,\sigma} + \tilde{a}_{j,\sigma}^{\dagger}\tilde{a}_{i,\sigma}\right), \tag{3.61}$$

$$t_{i,i}^{(\mathrm{round})} = c_T = \frac{1}{2}\left(t_\alpha(0) + t_\beta(0)\right), \tag{3.62}$$

$$t_{i,j}^{(\mathrm{round})} = -\frac{\alpha}{4d}\sqrt{f(\omega_{i,j})}, \tag{3.63}$$

where

$$\omega_{i,j} = \alpha\gamma_{i,j}^2, \qquad f(\omega) = \omega^2\exp(-\omega). \tag{3.64}$$

Before getting to the rounded potential operator, let's consider the difference between $T^{(\mathrm{round})}$ and the true kinetic operator $T^{(\mathrm{ES})}$ in Eq. (3.55). Let $\psi_0 = 1/\sqrt{2}$ and $\psi_l = 1/\sqrt{2d}$ for $l > 0$. Since the composite orbitals are superpositions of the primitive orbitals, the kinetic energy term for a pair of composite orbitals is just a linear combination of kinetic energy terms for pairs of primitive orbitals:

$$\tilde{t}_{i,j} = \sum_{p,q} \psi_p\psi_q\tilde{t}_{(i,p),(j,q)}. \tag{3.65}$$

We will eventually show that the kinetic energy contribution for pairs of primitive orbitals that are at least Γ apart will be negligible. Therefore, the only kinetic energy terms that contribute significantly to the sum above are $\tilde{t}_{(i,p),(i,p)}$ and $\tilde{t}_{(i,p),(j,q)}$, where $\{i,j\}$ is an edge and $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$. This means that for edge $\{i,j\}$, there is only one significant term in the sum for $\tilde{t}_{i,j}$. If $\{i,j\}$ is not an edge, then all of the primitive orbitals for composite orbitals i and j are at least Γ apart, and $\tilde{t}_{i,j} \approx 0$. For the diagonal terms $\tilde{t}_{i,i}$, there are $d+1$ significant terms in the sum, corresponding to $\tilde{t}_{(i,p),(i,p)}$ terms. Thus, we will show that

$$\tilde{t}_{i,j} = \sum_{p,q} \psi_p\psi_q\tilde{t}_{(i,p),(j,q)} \approx \begin{cases} \sum_p \psi_p^2\tilde{t}_{(i,p),(i,p)}, & i = j, \\ \frac{1}{2d}\tilde{t}_{(i,p),(j,q)} & \{i,j\} \in E, \\ 0, & \{i,j\} \notin E, \end{cases} \tag{3.66}$$

where $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$. We would now like to approximate each $\tilde{t}_{(i,p),(j,q)}$ with $t_{(i,p),(j,q)}$, which is the kinetic energy term for a pair of simple Gaussians. This turns out to be a sufficiently accurate approximation for $\tilde{t}_{(i,p),(i,p)}$. Note that

$$t_{i,i}^{(\text{round})} = \frac{1}{2}\left(t_\alpha(0) + t_\beta(0)\right) = \sum_p \psi_p^2 t_{(i,p),(i,p)}.$$

However, for edge $\{i,j\}$, where $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$, primitive orbitals $\phi_{(i,p)}$ and $\phi_{(j,q)}$ are only $\gamma_{i,j}$ apart. In this case, there is sufficient overlap between the orbitals that the effect of orthonormalizing the orbitals has a significant impact on the kinetic energy between the pair. Therefore, instead of setting $t_{i,j}^{(\text{round})}$ to be $(1/2d)t_{(i,p),(j,q)}$, we use a slightly corrected expression as defined by the function f. For comparison:

$$\frac{1}{2d} t_{(i,p),(j,q)} = \frac{\alpha}{4d}(3 - \omega_{i,j}) \exp(-\omega_{i,j}/2), \tag{3.67}$$

$$t_{i,j}^{(\text{round})} = -\frac{\alpha}{4d}\sqrt{f(\omega_{i,j})} = \frac{-\alpha\omega_{i,j}}{4d} \exp(-\omega_{i,j}/2). \tag{3.68}$$

For the potential operator, the coefficients u are a sufficiently good approximation for the $\tilde{u}$. We will show that we can also drop potential energy terms that involve any two primitive orbitals that are a distance at least Γ apart. Thus, we only need to include terms $u_{[(i,p),(j,q)],[(k,r),(l,s)]}$, where the indices (i,p), (j,q), (k,r), and (l,s) are all the same or all come from the set $\mathcal{B}(i,j)$ for some edge $\{i,j\}$. Thus, $u_{i,j,k,l}^{(\text{round})}$ will be 0, except when i, j, k, l are all equal or are all endpoints of the same edge. The rounded potential operator is

$$U^{(\text{round})} = \frac{1}{2} \sum_{\substack{(i,j,k,l)\in B \\ \sigma\in\pm 1}} u_{i,j,k,l}^{(\text{round})} \tilde{a}_{i,\sigma}^\dagger \tilde{a}_{j,\tau}^\dagger \tilde{a}_{k,\tau} \tilde{a}_{l,\sigma}, \tag{3.69}$$

where

$$B = \bigcup_{\{i,j\}\in E} \{i,j\}^4 \tag{3.70}$$

is the set of all 4-tuples of indices such that they are all the same or there are two distinct indices corresponding to an edge in the graph. For example, $(i,i,i,i), (i,j,j,i) \in B$ but

$(i,i,j,k),(i,k,k,i) \notin B$ for $\{i,j\} \in E$ and $\{i,k\} \notin E$. The coefficients are defined as

$$c_U^{(\text{round})} = u_{i,i,i,i}^{(\text{round})} = \frac{1}{4}u_{(i,0),(i,0),(i,0),(i,0)} + \frac{1}{4d^2}\sum_{p\in[d]} u_{(i,p),(i,p),(i,p),(i,p)} \tag{3.71}$$

$$= \frac{1}{4}u_\beta^{(\text{Coul})}(0) + \frac{1}{4d}u_\alpha^{(\text{Coul})}(0), \tag{3.72}$$

$$u_{i,j,j,i}^{(\text{round})} = u_{j,i,i,j}^{(\text{round})} = \frac{1}{4d^2}u_{(i,p),(j,q),(j,q),(i,p)} = \frac{1}{4d^2}u_\alpha^{(\text{Coul})}(\gamma_{i,j}), \tag{3.73}$$

$$u_{i,i,j,j}^{(\text{round})} = u_{j,j,i,i}^{(\text{round})} = u_{i,j,i,j}^{(\text{round})} = u_{j,i,j,i}^{(\text{round})} = \frac{1}{4d^2}u_{(i,p),(i,p),(j,q),(j,q)} = \frac{1}{4d^2}u_\alpha^{(\text{exch})}(\gamma_{i,j}), \tag{3.74}$$

$$u_{i,i,i,j}^{(\text{round})} = u_{i,i,j,i}^{(\text{round})} = u_{i,j,i,i}^{(\text{round})} = u_{j,i,i,i}^{(\text{round})} \tag{3.75}$$

$$u_{j,j,j,i}^{(\text{round})} = u_{j,j,i,j}^{(\text{round})} = u_{j,i,j,j}^{(\text{round})} = u_{i,j,j,j}^{(\text{round})} = \frac{1}{4d^2}u_{(i,p),(i,p),(i,p),(j,q)} = \frac{1}{4d^2}u_\alpha^{(\text{other})}(\gamma_{i,j}), \tag{3.76}$$

with $u_{i,j,k,l}^{(\text{round})} = 0$ for $(i,j,k,l) \notin B$. The following lemma bounds the difference between $H^{(\text{ES})}$ and $H^{(\text{round})}$.

Lemma 1. *If $\beta \geq \alpha \geq 1$, $\omega_{\min} \geq 4$, $\Gamma \geq 640n^{18}\beta^3$, and $\alpha\Gamma^2 \geq 12\log\beta + 80\log n + 4\omega_{\min} + 24$, then*

$$\left\|H^{(\text{ES})} - H^{(\text{round})}\right\| \leq 3n^2\alpha f(\omega_{\min}) + \frac{1}{20n^2} + 8n^4\sqrt{\alpha}\cdot\exp(-\omega_{\min}/2), \tag{3.77}$$

where $\omega_{\min} = \alpha\gamma_{\min}^2$.

The matrices T, $\tilde{T}$, S, and $S^{-1/2}$ are all close to block diagonal. Blocks are either single entries on the diagonal (corresponding to primitive orbitals that are a distance at least Γ from all other primitive orbitals) or a 2×2 sub-matrix corresponding to an edge $\{i,j\}$. Suppose that $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$. For any $n(d+1)\times n(d+1)$ matrix A, let $A_{i,j}$ denote the 2×2 sub-matrix of A indexed by the elements of $\mathcal{B}(i,j)$:

$$A_{i,j} = \begin{pmatrix} a_{(i,p),(i,p)} & a_{(i,p),(j,q)} \\ a_{(j,q),(i,p)} & a_{(j,q),(j,q)} \end{pmatrix}.$$

We refer to all of the $A_{i,j}$ blocks collectively as the *edge blocks* of A. The proof of Lemma 1 uses the fact that the off-diagonal terms of T outside of the $T_{i,j}$ blocks are small. The same is true for $\tilde{T}$ and $R = S^{-1/2}$.

$\tilde{U}$ and U are also related by conjugation by $S^{-1/2}\otimes S^{-1/2}$. We will show that $\tilde{U}$ and U are also close to block diagonal. We define $U_{i,j}$ to be the 4×4 sub-matrix of U corresponding to the intersections of the four rows and four columns indexed by:

$$[(i,p),(i,p)],\quad [(i,p),(j,q)],\quad [(j,q),(i,p)],\quad [(j,q),(j,q)]$$

The proof of Lemma 1 uses the fact that the off-diagonal terms of U outside of the $U_{i,j}$ blocks are small. The same is true for $\tilde{U}$. We refer to all of the $U_{i,j}$ blocks collectively as the *edge blocks*.

Lemma 1 is proved in Section 3.6. Outside of this subsection, all creation and annihilation operators are those of the orthonormalized basis $\tilde{\boldsymbol{\phi}}$; in other words, we'll drop the tildes.

3.4.3 Getting the main Hamiltonian

With the rounded Hamiltonian $H^{(\text{round})}$ in hand, we make one final approximation to get to the main Hamiltonian $H^{(\text{main})}$ that we will later show is close to a Hubbard Hamiltonian. Specifically, we remove the "off-diagonal" Coulomb interaction terms. The error of this approximation is bounded by Lemma 2.

The main Hamiltonian is

$$H^{(\text{round})} = H^{(\text{main})} + H^{(\text{approx})} + n \cdot c_T, \tag{3.78}$$

$$H^{(\text{main})} = c_U^{(\text{main})} \sum_i n_{i,+1} n_{i,-1} + \sum_{\substack{\{i,j\} \in E \\ \sigma \in \{\pm 1\}}} t_{i,j}^{(\text{round})} \left(a_{i,\sigma}^\dagger a_{j,\sigma} + a_{j,\sigma}^\dagger a_{i,\sigma} \right), \tag{3.79}$$

where $c_U^{(\text{main})} = u_\beta^{(\text{Coul})}(0)/4$. The difference $H^{(\text{round})} - H^{(\text{main})} - n \cdot c_T$ contains two types of terms, both of whose coefficients are $O(\sqrt{\alpha})$: the smaller part of the onsite terms $c_U^{(\text{round})}$, and the offsite terms corresponding to edges in the interaction graph. The following lemma bounds the contribution from this difference..

Lemma 2.

$$\left\| H^{(\text{round})} - H^{(\text{main})} - n c_T \right\| \leq 30 n^2 \sqrt{\alpha}. \tag{3.80}$$

Proof of Lemma 2. First, recall that we're restricting to the fixed-particle number subspace, in which the diagonal part $c_T \sum_{i,\sigma} n_{i,\sigma}$ of $T^{(\text{round})}$ is the constant $n \cdot c_T$. That is, $T^{(\text{round})} = T^{(\text{main})} + n \cdot c_T$. Let

$$B_2 = B \setminus \{(i,i,i,i) : i \in [n]\} \tag{3.81}$$

be the subset of B whose elements contain two distinct indices (corresponding to an edge).

$$\left\|H^{(\text{round})} - H^{(\text{main})} - n\cdot c_T\right\| = \left\|U^{(\text{round})} - U^{(\text{main})}\right\| \tag{3.82}$$

$$= \left\| \frac{1}{2} \sum_{\substack{(i,j,k,l)\in B\\ \sigma,\tau\in\{\pm1\}}} u^{(\text{round})}_{i,j,k,l} a^\dagger_{i,\sigma} a^\dagger_{j,\tau} a_{k,\tau} a_{l,\sigma} - c^{(\text{main})}_U \sum_{i\in[n]} n_{i,+1} n_{i,-1} \right\| \tag{3.83}$$

$$\le \sum_{i\in[n]} \left|c^{(\text{round})}_U - c^{(\text{main})}_U\right| + \frac{1}{2} \sum_{\substack{(i,j,k,l)\in B_2\\ \sigma,\tau\in\{\pm1\}}} u^{(\text{round})}_{i,j,k,l} \tag{3.84}$$

$$= n\cdot \frac{1}{4d} u^{(\text{Coul})}_\alpha(0) + \frac{1}{2} \sum_{\substack{(i,j,k,l)\in B_2\\ \sigma,\tau\in\{\pm1\}}} u^{(\text{round})}_{i,j,k,l} \tag{3.85}$$

$$\le n\cdot \frac{1}{4d} u^{(\text{Coul})}_\alpha(0) + \frac{1}{2}\cdot \underbrace{4}_{\sigma,\tau} \cdot \underbrace{14\cdot \binom{n}{2}}_{B_2} \cdot u^{(\text{Coul})}_\alpha(0) \tag{3.86}$$

$$\le 15n^2 u^{(\text{Coul})}_\alpha(0) = 15n^2 \cdot \frac{2}{\sqrt{\pi}}\sqrt{\alpha} \le 30n^2\sqrt{\alpha}. \tag{3.87}$$

□

3.4.4 Hardness of estimating ground state energy

Now, we're ready to prove the main theorem.

Proof of Theorem 4. Membership in QMA is straightforward. For hardness, we reduce from the Fermi-Hubbard model. Recall Theorem 6: for some p, q and all $u_0^{(\text{Hubb})} \ge n^{14+3p+2q}$, finding the ground state to precision n^{-q} of

$$H^{(\text{Hubb})} = u_0^{(\text{Hubb})} \sum_{i\in[n]} n_{i,+1} n_{i,-1} + \sum_{i<j} t^{(\text{Hubb})}_{i,j} \left(a^\dagger_{i,\sigma} a_{j,\sigma} + a^\dagger_{j,\sigma} a_{i,\sigma}\right) \tag{3.88}$$

subject to $\left|t^{(\text{Hubb})}_{i,j}\right| \le \sqrt{n^p u_0^{(\text{Hubb})}}$ is QMA-complete. In the preceding sections, we showed that, using our choice of single-electron orbitals, the electronic structure Hamiltonian is close to

$$H^{(\text{ES})} \approx H^{(\text{main})} + n\cdot c_T = c^{(\text{main})}_U n_{i,+1} n_{i,-1} + \sum_{i<j} t^{(\text{round})}_{i,j} \left(a^\dagger_{i,\sigma} a_{j,\sigma} + a^\dagger_{j,\sigma} a_{i,\sigma}\right) + n\cdot c_T. \tag{3.89}$$

To prove the theorem, it suffices to show that for any Hubbard Hamiltonian satisfying the conditions of Theorem 6, we can set the parameters $\alpha, \beta, \{\gamma_{i,j}\}_{i,j}, \Gamma$ such that

$$\rho H^{(\text{Hubb})} = H^{(\text{main})} \tag{3.90}$$

and

$$\left\|H^{(\mathrm{main})} - H^{(\mathrm{ES})} - n \cdot c_T\right\| = o\left(\rho n^{-q}\right) \tag{3.91}$$

for some $\rho \in \mathbb{R}$. With this, finding the ground state of $H^{(\mathrm{ES})}$ to precision $O(\rho n^{-q})$ would allow us to find the ground state of $H^{(\mathrm{Hubb})}$ to precision $O(n^{-q})$, and so the former must be QMA-hard. We'll base our parameterization on four constants independent of n:

$$a = \log_n \alpha, \qquad b = \log_n \beta, \qquad r = \log_n \rho, \qquad g = -\log_n \sqrt{f(\omega_0)} = -\frac{1}{2}\log_n f(\omega_0), \tag{3.92}$$

where ω_0 is a lower bound on $\omega_{i,j}$ to be set later. The first three immediately set α, β, and ρ, respectively. Equating $n^r H^{(\mathrm{Hubb})}$ and $H^{(\mathrm{main})}$ requires

$$n^r u_0^{(\mathrm{Hubb})} = c_U^{(\mathrm{main})} = \frac{1}{2\sqrt{\pi}}\sqrt{\beta}, \tag{3.93}$$

$$n^r t_{i,j}^{(\mathrm{Hubb})} = -\frac{\alpha}{4d}\sqrt{f(\omega_{i,j})}. \tag{3.94}$$

Coefficient ranges If we set

$$\boxed{b = 30 + 6p + 4q + 2r} \tag{3.95}$$

then

$$u_0^{(\mathrm{Hubb})} = \frac{\sqrt{\beta}}{2\sqrt{\pi}n^r} \geq \frac{1}{4}\frac{\sqrt{\beta}}{n^r} = 4^{-1}n^{0.5b-r} = \frac{n}{4}n^{14+3p+2q} \geq n^{14+3p+2q} \qquad n \geq 4 \tag{3.96}$$

satisfies the lower bound in the statement of Theorem 6.

If we set

$$\boxed{g = -\frac{1}{2}p + a - \frac{1}{4}b - \frac{3}{2} - \frac{1}{2}r \geq 1}, \tag{3.97}$$

then for $n \geq 9$,

$$\frac{\alpha}{4d}\sqrt{f(\omega_0)} \geq \frac{\alpha}{4n}\sqrt{f(\omega_0)} = \frac{1}{4}n^{a-g-1} = \frac{\sqrt{n}}{4}n^{\frac{1}{2}r+\frac{1}{2}p+\frac{1}{4}b} \tag{3.98}$$

$$\geq \frac{3}{4}n^{\frac{1}{2}r+\frac{1}{2}p+\frac{1}{4}b} \tag{3.99}$$

$$\geq \frac{1}{\sqrt{2\sqrt{\pi}}}n^{\frac{1}{2}r+\frac{1}{2}p+\frac{1}{4}b} = n^r\sqrt{n^p\frac{n^{-r}\sqrt{\beta}}{2\sqrt{\pi}}} = \rho\sqrt{n^p u_0^{(\mathrm{Hubb})}}, \tag{3.100}$$

and thus for any $t_{i,j}^{(\mathrm{Hubb})} \leq \sqrt{n^p u_0^{(\mathrm{Hubb})}}$ there is some $\omega_{i,j} \geq \omega_0$ that satisfies Eq. (3.94).

Bounding the difference between ES and Hubbard The difference between the electronic structure Hamiltonian and the main Hamiltonian is

$$\begin{aligned}\left\|H^{(\text{main})}+n\cdot c_T-H^{(\text{ES})}\right\| &\leq \left\|H^{(\text{main})}+n\cdot c_T-H^{(\text{round})}\right\|+\left\|H^{(\text{round})}-H^{(\text{ES})}\right\| && (3.101)\\ &\leq 30n^2\sqrt{\alpha}+\frac{1}{20n^2}+3n^2\alpha f(\omega_{\min})+8n^4\sqrt{\alpha}\exp(-\omega_{\min}/2) && (3.102)\\ &\leq 30n^2\sqrt{\alpha}+\frac{1}{20n^2}+3n^2\alpha f(\omega_0)+8n^4\sqrt{\alpha}\sqrt{f(\omega_0)} && (3.103)\\ &= O\left(n^{2+\frac{1}{2}a}+n^{-2}+n^{2+a-2g}+n^{4+\frac{1}{2}a-g}\right) && (3.104)\\ &= O\left(n^{4+\frac{1}{2}a}+n^{2+a-2g}\right). && (3.105)\end{aligned}$$

Therefore, to satisfy Eq. (3.91), it would suffice to have

$$\boxed{4+\frac{1}{2}a<r-q} \tag{3.106}$$

and

$$2+a-2g<r-q. \tag{3.107}$$

Plugging Eq. (3.97) into the latter yields

$$\boxed{p+q+5<a-\frac{1}{2}b}. \tag{3.108}$$

Parameter setting In summary, our constraints are

$$\begin{aligned}30+6p+4q&=b-2r, && (3.109)\\ \frac{1}{2}p+\frac{5}{2}&<a-\frac{1}{4}b-\frac{1}{2}r, && (3.110)\\ 4+q&<-\frac{1}{2}a+r, && (3.111)\\ p+q+5&<a-\frac{1}{2}b. && (3.112)\end{aligned}$$

The following settings satisfy all the required constraints:

$$\begin{aligned}a&=18p+12q+90, && (3.113)\\ b&=\frac{5}{3}a, && (3.114)\\ r&=\frac{2}{3}a, && (3.115)\\ g&=\frac{1}{4}a-\frac{1}{2}p-\frac{3}{2}. && (3.116)\end{aligned}$$

□

3.5 Hardness of finding lowest-energy Slater determinant

In this section, we show that finding the lowest-energy Slater determinant (i.e., Hartree-Fock state) of an electronic structure Hamiltonian is NP-hard. This is a natural complement to our QMA-hardness result, in that Slater determinants are the most natural class of fermionic states that are efficiently representable and manipulable classically. The proof has much in common with that of Theorem 4. We start with the same parameterized construction of orbitals described in Section 3.4.1, and then orthonormalize and round them as in Section 3.4.2 to get the Hamiltonian $H^{(\text{round})}$. We then diverge from the QMA-hardness proof by setting the parameters in a different regime. Specifically, we set the exponents α and β large enough that the Hamiltonian becomes essentially classical (diagonal). The proof concludes by showing that this classical Hamiltonian can express an NP-hard problem such as independent set.

Theorem 5. Determining the lowest-energy Slater determinant of an electronic structure Hamiltonian in a fixed basis and with fixed particle number to inverse-polynomial precision is NP-complete.

Proof. To start, we'll set $\gamma_{i,j} = \gamma$ for all $\{i, j\} \in E$. We show that the parameters α, β, γ, Γ can be set such that the electronic structure approximates a diagonal Hamiltonian

$$H^{(\text{ES})} - n \cdot c_T \approx H^{(\text{class})} = u_1^{(\text{class})} \sum_i n_{i,+1} n_{i,-1} + u_2^{(\text{class})} \sum_{\substack{\{i,j\} \in E \\ \sigma,\tau \in \{\pm 1\}}} n_{i,\sigma} n_{j,\tau}, \tag{3.117}$$

where

$$u_1^{(\text{class})} = c_U^{(\text{round})}, \qquad\qquad u_2^{(\text{class})} = \frac{1}{4d^2} u_\alpha^{(\text{Coul})}(\gamma). \tag{3.118}$$

For a diagonal Hamiltonian, there is always a computational basis state of lowest energy. Because basis states are a special case of Slater-determinants, finding the lowest-energy Slater-determinant for diagonal Hamiltonians is equivalent to finding the ground state.

For sufficiently large $u_1^{(\text{class})} > 4n^2 u_2^{(\text{class})}$, the ground space of $H^{(\text{class})}$ in the k-electron subspace for $k \le n$ will have at most one electron in each spatial orbital, and the ground state energy is

$$h(\mathbf{n}) = u_2^{(\text{class})} \sum_{\{i,j\} \in E} n_i n_j, \tag{3.119}$$

where $n_i = n_{i,+1} + n_{i,-1}$ is the occupancy of the i-th spatial orbital.

The state space is spanned by vectors $\mathbf{n}$ such that $\sum_i n_i = k$, which we can interpret as representing a subset $S \subset V$ of vertices with size $|S| = k$. The classical function $h(\mathbf{n})$ is then proportional to the number of edges with both endpoints in the set S. In other words, if $h(\mathbf{n}) = 0$, then the set S is an independent set of size k; otherwise $h(\mathbf{n}) \ge u_2^{(\text{class})}$. Therefore, if $u_1^{(\text{class})}$ is sufficiently larger than $u_2^{(\text{class})}$, then finding the lowest-energy Slater-determinant of $H^{(\text{class})}$ in the k-electron subspace to precision $u_2^{(\text{class})}$ is as hard as determining if a graph has an independent set of size k.

To finish the proof, we just need to set the parameters such that

$$u_1^{(\text{class})} > 4n^2 u_2^{(\text{class})}, \tag{3.120}$$

$$\left\| H^{(\text{ES})} - H^{(\text{class})} \right\| < \frac{1}{2} u_2^{(\text{class})}. \tag{3.121}$$

Let $\gamma = 1$, leaving α, β, and Γ to be set. The first constraint is satisfied by $\beta \geq 16n^4$:

$$\tag{3.122}$$

$$\begin{aligned} u_1^{(\text{class})} &= c_U^{(\text{round})} \geq \frac{1}{4} u_\beta^{(\text{Coul})}(0) && (3.123)\\ &= \frac{1}{4}\sqrt{\frac{4\beta}{\pi}} > \frac{1}{4}\sqrt{\beta} && (3.124)\\ &\geq n^2 && (3.125)\\ &\geq \frac{n^2}{d^2} \operatorname{erf}\left(\sqrt{\alpha\gamma^2}\right) && (3.126)\\ &= 4n^2 u_2^{(\text{class})}. && (3.127) \end{aligned}$$

For the second constraint, if $\alpha \geq 1$, then

$$\frac{1}{2} u_2^{(\text{class})} \geq \frac{1}{2}\frac{1}{4d^2} \operatorname{erf} 1 \geq \frac{1}{8n^2} \cdot \frac{1}{2} \geq \frac{1}{16n^2}. \tag{3.128}$$

Lemma 3. For $\alpha \geq 1$, $\gamma_{i,j} = \gamma \geq 1$,

$$\left\| H^{(\text{round})} - H^{(\text{class})} \right\| \leq 14\alpha n^2 e^{-\alpha\gamma^2/4}. \tag{3.129}$$

Proof of Lemma 3. The classical Hamiltonian $H^{(\text{class})}$ has no kinetic component, and so we need to bound the entirety of the non-constant kinetic component of the rounded Hamiltonian $H^{(\text{round})}$:

$$\begin{aligned} \left\| T^{(\text{round})} - n \cdot c_T \right\| &= \left\| \sum_{\substack{\{i,j\}\in E \\ \sigma\in\{\pm 1\}}} t_{i,j}^{(\text{round})} \right\| \left(a_{i,\sigma}^\dagger a_{j,\sigma} + a_{j,\sigma}^\dagger a_{i,\sigma} \right) && (3.130)\\ &\leq 4 \sum_{\{i,j\}\in E} \left| t_{i,j}^{(\text{round})} \right| && (3.131)\\ &\leq \frac{\alpha}{d} n^2 \sqrt{f(\omega)} \leq \alpha^2 n^2 \gamma^2 e^{-\alpha\gamma^2/2}. && (3.132) \end{aligned}$$

For the potential difference, define

$$B_3 = B_2 \setminus \{(i,j,j,i) : \{i,j\} \in E\}, \tag{3.133}$$

i.e. the indices of potential terms that are *not* Coulomb (which are exactly those included in $H^{(\text{class})}$). Then

$$
\begin{aligned}
&\left\|U^{(\text{round})} - H^{(\text{class})}\right\| && (3.134)\\
&= \left\| \frac{1}{2} \sum_{\substack{(i,j,k,l)\in B\\ \sigma,\tau\in\{\pm1\}}} u^{(\text{round})}_{i,j,k,l} a^\dagger_{i,\sigma} a^\dagger_{j,\tau} a_{k,\tau} a_{l,\sigma} - u_1^{(\text{class})} \sum_{i\in[n]} n_{i,+1} n_{i,-1} - u_2^{(\text{class})} \sum_{\substack{\{i,j\}\in E\\ \sigma,\tau\in\{\pm1\}}} n_{i,\sigma} n_{j,\tau}, \right\| && (3.135)\\
&\le \frac{1}{2} \sum_{\substack{(i,j,k,l)\in B_3\\ \sigma,\tau\in\{\pm1\}}} u^{(\text{round})}_{i,j,k,l} && (3.136)\\
&\le \frac{1}{2} \cdot \underbrace{4}_{\sigma,\tau} \cdot \underbrace{12 \cdot \binom{n}{2}}_{B_3} \cdot \frac{1}{4d^2} 2\sqrt{\alpha} \exp\left(-\alpha\gamma^2/2\right) && (3.137)\\
&\le 6\sqrt{\alpha} n^2 \exp\left(-\alpha\gamma^2/2\right). && (3.138)
\end{aligned}
$$

Putting them together,

$$
\begin{aligned}
\left\|H^{(\text{round})} - H^{(\text{class})}\right\| &\le \left\|T^{(\text{round})} - n\cdot c_T\right\| + \left\|U^{(\text{round})} - H^{(\text{class})} - n\cdot c_T\right\| && (3.139)\\
&\le 7\alpha n^2(\alpha\gamma^2) e^{-\alpha\gamma^2/2} \le 14\alpha n^2 e^{-\alpha\gamma^2/4} && (3.140)
\end{aligned}
$$

□

Together, Lemmas 1 and 3 imply that for $\gamma = 1$, $\beta \ge \alpha > 74 + 48\log n$, $\Gamma \ge 640 n^{18}\beta^3$, and $\alpha\Gamma^2 \ge 12\log\beta + 80\log n + 4\alpha + 24$,

$$
\begin{aligned}
\left\|H^{(\text{ES})} - H^{(\text{class})} - n\cdot c_T\right\| &\le \left\|H^{(\text{ES})} - H^{(\text{round})}\right\| + \left\|H^{(\text{round})} - H^{(\text{class})} - n\cdot c_T\right\| && (3.141)\\
&\le 3n^2\alpha f(\omega_{\min}) + \frac{1}{20n^2} + 8n^4\sqrt{\alpha}\cdot\exp(-\omega_{\min}/2) + 14\alpha n^2 e^{-\alpha\gamma^2/4} && (3.142)\\
&= 3n^2\alpha^2 e^{-\alpha/2} + \frac{1}{20n^2} + 8n^4\sqrt{\alpha}\cdot\exp(-\alpha/2) + 14\alpha n^2 e^{-\alpha/4} && (3.143)\\
&\le \frac{1}{20n^2} + 100n^4 e^{-\alpha/8} \le \frac{1}{20n^2} + \frac{1}{80n^2} = \frac{1}{16n^2} && (3.144)
\end{aligned}
$$

where used the fact that for $x \ge 0$, $\max\{x^2 e^{-x/2}, \sqrt{x}e^{-x/2}, xe^{-x/4}\} \le 4e^{-x/8}$. For sufficiently large n, it suffices to set $\beta = \alpha = n$, $\gamma = 1$, and $\Gamma = n^{32}$. □

3.6 Proof of Lemma 1

The proof of Lemma 1 uses the following technical lemmas that quantify the statement that matrices $\tilde{T}$, S, and $\tilde{U}$ are approximately block diagonal. It will be convenient to refer only to the entries along the diagonal or inside the edge blocks. For an $n(d+1)\times n(d+1)$ matrix or an $n^2(d+1)^2\times n^2(d+1)^2$ matrix A, let $A^{(\text{block})}$ to denote the matrix obtained by replacing all of the off-diagonal entries of A outside the edge blocks with 0. Define

$$S^{(\text{neg})} = S - S^{(\text{block})}, \tag{3.145}$$

$$R = S^{-1/2}, \tag{3.146}$$

$$R^{(\text{aprx})} = (S^{(\text{block})})^{-1/2}, \tag{3.147}$$

$$R^{(\text{neg})} = R - R^{(\text{aprx})}. \tag{3.148}$$

Note that, because $S^{(\text{block})}$ is block diagonal, $R^{(\text{aprx})}$ is also block diagonal. However $R^{(\text{block})}\neq R^{(\text{aprx})}$. The matrix $R^{(\text{neg})}$, unlike $S^{(\text{neg})}$, has non-zero entries even on the diagonal and within the blocks, though these are small.

The first lemma bounds $\max(|R^{(\text{neg})}|)$, where $\max(|A|)$ is defined to be the maximum of the absolute values of the entries in matrix A.

Lemma 4. If $\alpha\Gamma^2 \geq 4\log n + 2\omega_{\min} + 2$ and $\omega_{\min}\geq 4$, then

$$r^{(\text{neg})}_{\max} = \max\left(\left|R^{(\text{neg})}\right|\right) \leq n^2 \exp\left[-(\alpha\Gamma^2 - \omega_{\min})/2\right]. \tag{3.149}$$

Corollary 1. For $\alpha\Gamma^2 \geq 4\log n + 2\omega_{\min} + 2$ and $\omega_{\min}\geq 2$,

$$\begin{aligned} r^{(\text{neg})}_{\max} &\leq n^2\exp\left[-\left(\alpha\Gamma^2 - \omega_{\min}\right)/2\right] \\ &\leq n^2 \exp\left[-\left(4\log n + 2\omega_{\min} + 2 - \omega_{\min}\right)/2\right] \leq \exp(-1) \leq 1/2. \end{aligned} \tag{3.150}$$

The entries of matrix S are just the overlap of normalized Gaussians, so the diagonal is all ones. Block $S_{i,j}$ corresponding to edge $\{i,j\}$, where $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$ is

$$S_{i,j} = \begin{pmatrix} 1 & \epsilon_{i,j} \\ \epsilon_{i,j} & 1 \end{pmatrix}, \tag{3.151}$$

$$\epsilon_{i,j} = s_{(i,p),(j,q)} = s_\alpha(\gamma_{i,j}) = \exp(-\omega_{i,j}/2), \qquad \text{where } \omega_{i,j} = \alpha\gamma_{i,j}^2. \tag{3.152}$$

The entry $s_{(i,p),(i,p)}$ is not contained in an edge block if and only if $p=0$ or $p>i$. In this case, the orbital $\phi_{i,p}$ is at least a distance Γ away from every other primitive orbital, and the block for $s_{(i,p),(i,p)}$ is just the single element on the diagonal. For these primitive orbitals, we have

$$s_{(i,p),(i,p)} = r^{(\text{aprx})}_{(i,p),(i,p)} = 1.$$

The edge blocks of $R^{(\text{aprx})}$ can be computed exactly as

$$R_{i,j}^{(\text{aprx})} = (S_{i,j})^{-1/2} \tag{3.153}$$

$$= \frac{1}{2}\begin{pmatrix} \frac{1}{\sqrt{1+\epsilon_{i,j}}} + \frac{1}{\sqrt{1-\epsilon_{i,j}}} & \frac{1}{\sqrt{1+\epsilon_{i,j}}} - \frac{1}{\sqrt{1-\epsilon_{i,j}}} \\ \frac{1}{\sqrt{1+\epsilon_{i,j}}} - \frac{1}{\sqrt{1-\epsilon_{i,j}}} & \frac{1}{\sqrt{1+\epsilon_{i,j}}} + \frac{1}{\sqrt{1-\epsilon_{i,j}}} \end{pmatrix}. \tag{3.154}$$

The following lemma bounds the error from just taking the leading term in $\epsilon_{i,j}$. Note that the 2×2 matrix $R_{i,j}^{(\text{aprx})}$ has identical on-diagonal entries and identical off-diagonal entries. Let $\textsc{On}(R_{i,j}^{(\text{aprx})})$ refer to the value of the on-diagonal entries and let $\textsc{Off}(R_{i,j}^{(\text{aprx})})$ refer to the value of the off-diagonal entries. The matrix $R_{i,j}^{(\text{aprx})}T_{i,j}R_{i,j}^{(\text{aprx})}$ has the same symmetries, so we can define $\textsc{On}$ and $\textsc{Off}$ for those matrices as well.

Lemma 5. For $\omega_{\min} \geq 4$ and $\{i,j\} \in E$ where $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$,

$$1 \leq \textsc{On}(r^{(\text{aprx})}) \leq 1 + \epsilon_{i,j}^2, \tag{3.155}$$

$$-\frac{\epsilon_{i,j}}{2} - \epsilon_{i,j}^3 \leq \textsc{Off}(r^{(\text{aprx})}) \leq -\frac{\epsilon_{i,j}}{2}, \tag{3.156}$$

$$t_\alpha(0) \leq \textsc{On}\left(R_{i,j}^{(\text{aprx})}T_{i,j}R_{i,j}^{(\text{aprx})}\right) \leq t_\alpha(0) + \alpha\omega_{i,j}\epsilon_{i,j}^2, \tag{3.157}$$

$$-\frac{\alpha}{2}\sqrt{f(\omega_{i,j})}(1+4\epsilon_{i,j}^2) \leq \textsc{Off}\left(R_{i,j}^{(\text{aprx})}T_{i,j}R_{i,j}^{(\text{aprx})}\right) \leq -\frac{\alpha}{2}\sqrt{f(\omega_{i,j})}, \tag{3.158}$$

$$\max\left(\left|\left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2}U_{i,j}\left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} - U_{i,j}\right|\right) \leq 16\sqrt{\alpha}\epsilon_{i,j}. \tag{3.159}$$

Corollary 2. For $\omega_{\min} \geq 4$,

$$r_{\max}^{(\text{aprx})} = \max\left(\left|R^{(\text{aprx})}\right|\right) \leq 3/2. \tag{3.160}$$

Corollary 3. For $\alpha\Gamma^2 \geq 4\log n + \omega_{\min} + 2$ and $\omega_{\min} \geq 4$,

$$r_{\max} = \max(|R|) \leq r_{\max}^{(\text{aprx})} + r_{\max}^{(\text{neg})} \leq 2. \tag{3.161}$$

Define $T^{(\text{neg})} = T - T^{(\text{block})}$. Similarly, define $U^{(\text{neg})} = U - U^{(\text{block})}$. The following lemma bounds these coefficients.

Lemma 6. For $\beta \geq \alpha \geq 1$ and $\alpha\Gamma^2 \geq 64$,

$$t_{\max} = \max(|T|) \leq \frac{3}{2}\beta, \tag{3.162}$$

$$t_{\max}^{(\text{neg})} = \max\left(\left|T^{(\text{neg})}\right|\right) \leq \beta\exp\left(-\alpha\Gamma^2/4\right), \tag{3.163}$$

$$u_{\max} = \max(|U|) \leq 2\beta^3, \tag{3.164}$$

$$u_{\max}^{(\text{neg})} = \max\left(\left|U^{(\text{neg})}\right|\right) \leq 2\beta^3/\Gamma. \tag{3.165}$$

Proofs of the technical lemmas follow the proof of Lemma 1. Note that the conditions of the technical lemmas (and corollaries) are implied by the conditions of Lemma 1.

Lemma 1 (restated)**.** If $\beta \geq \alpha \geq 1$, $\omega_{\min} \geq 4$, $\Gamma \geq 640 n^{18}\beta^3$, and $\alpha\Gamma^2 \geq 12\log\beta + 80\log n + 4\omega_{\min} + 24$, then

$$\left\|H^{(\mathrm{ES})} - H^{(\mathrm{round})}\right\| \leq 3n^2\alpha f(\omega_{\min}) + \frac{1}{20n^2} + 8n^4\sqrt{\alpha}\cdot\exp(-\omega_{\min}/2), \tag{3.77}$$

where $\omega_{\min} = \alpha\gamma_{\min}^2$.

Proof of Lemma 1. We will bound the kinetic and potential parts separately, starting with the former.

Define $\tilde{T}^{(\mathrm{aprx})} = R^{(\mathrm{aprx})}T^{(\mathrm{block})}R^{(\mathrm{aprx})}$, and recall that $\tilde{T} = RTR$. The first task is to bound the error of approximating $\tilde{T}$ by $\tilde{T}^{(\mathrm{aprx})}$:

$$\begin{aligned}
\max\left(\left|\tilde{T} - \tilde{T}^{(\mathrm{aprx})}\right|\right) &= \max\left(\left|RTR - R^{(\mathrm{aprx})}T^{(\mathrm{block})}R^{(\mathrm{aprx})}\right|\right) && (3.166)\\
&\leq \max\left(\left|RTR - RT^{(\mathrm{block})}R\right|\right) && (3.167)\\
&\quad + \max\left(\left|RT^{(\mathrm{block})}R - R^{(\mathrm{aprx})}T^{(\mathrm{block})}R^{(\mathrm{aprx})}\right|\right).
\end{aligned}$$

We will bound each term from (3.168) separately. We will use the fact that if A and B are $m\times m$ matrices, then $\max(|AB|) \leq m\cdot\max(|A|)\cdot\max(|B|)$. Since the matrices R and T are $n(d+1)\times n(d+1)$ matrices and $d+1 \leq n$, we will pick up a factor of at most n^2 every time this rule is applied.

$$\begin{aligned}
\max\left(\left|RTR - RT^{(\mathrm{block})}R\right|\right) &= \max\left(\left|R(T - T^{(\mathrm{block})})R\right|\right) && (3.168)\\
&\leq n^4(r_{\max})^2\max(|T - T^{(\mathrm{block})}|) = n^4(r_{\max})^2 t_{\max}^{(\mathrm{neg})} && (3.169)\\
&\leq 4n^2\beta\exp(-\alpha\Gamma^2/4). && (3.170)
\end{aligned}$$

The last inequality uses the bound from (3.161) that $r_{\max} \leq 2$ and from (3.163) that $t_{\max}^{(\mathrm{neg})} \leq \beta\exp(-\alpha\Gamma^2/4)$. To bound the second term from (3.168), recall that $R = R^{(\mathrm{aprx})} + R^{(\mathrm{neg})}$.

$$\begin{aligned}
&\max\left(\left|RT^{(\mathrm{block})}R - R^{(\mathrm{aprx})}T^{(\mathrm{block})}R^{(\mathrm{aprx})}\right|\right) && (3.171)\\
&= \max\left(\left|\left[R^{(\mathrm{aprx})} + R^{(\mathrm{neg})}\right]T^{(\mathrm{block})}\left[R^{(\mathrm{aprx})} + R^{(\mathrm{neg})}\right] - R^{(\mathrm{aprx})}T^{(\mathrm{block})}R^{(\mathrm{aprx})}\right|\right) && (3.172)\\
&= \max\left(\left|R^{(\mathrm{aprx})}T^{(\mathrm{block})}R^{(\mathrm{neg})} + R^{(\mathrm{neg})}T^{(\mathrm{block})}R^{(\mathrm{aprx})} + R^{(\mathrm{neg})}T^{(\mathrm{block})}R^{(\mathrm{neg})}\right|\right) && (3.173)\\
&\leq n^4 t_{\max} r_{\max}^{(\mathrm{neg})}\left(2r_{\max}^{(\mathrm{aprx})} + r_{\max}^{(\mathrm{neg})}\right) && (3.174)\\
&\leq n^4 \underbrace{\frac{3}{2}\beta}_{(3.162)} \underbrace{\exp\left[-(\alpha\Gamma^2 - \omega_{\min})/2\right]}_{(3.149)} \left(2\underbrace{\frac{3}{2}}_{(3.160)} + \underbrace{\frac{1}{2}}_{(3.150)}\right) && (3.175)\\
&\leq 6n^4\beta\exp\left[-(\alpha\Gamma^2 - \omega_{\min})/2\right] \leq 6n^4\beta\exp(-\alpha\Gamma^2/4) && (3.176)
\end{aligned}$$

The last inequality is implied by the assumptions of the lemma, specifically that $\alpha\Gamma^2 \geq 2\omega_{\min}$. Putting together the bounds from (3.170) and (3.176) we get that

$$\max\left(\left|\tilde{T} - \tilde{T}^{(\text{aprx})}\right|\right) \leq 10n^4\beta\exp(-\alpha\Gamma^2/4). \tag{3.177}$$

The next step is to use the approximation for the kinetic-energy terms for the primitive orbitals to get the kinetic-energy term for the composite orbitals. Recall that composite orbital ϕ_i is a superposition of $\phi_{i,p}$:

$$\phi_i = \sum_{p=0}^{d} \psi_p \phi_{i,p},$$

where $\psi_0 = 1/\sqrt{2}$ and $\psi_{p>0} = 1/\sqrt{2d}$. Therefore, the kinetic-energy terms for the composite orbitals are just superpositions of the kinetic-energy terms for the primitive orbitals:

$$t_{i,j} = \sum_{p,q} \psi_p \psi_q t_{(i,p),(j,q)}.$$

We can apply this principle to $\tilde{T}$ and $\tilde{T}^{(\text{aprx})}$ as well:

$$\tilde{t}_{i,j} = \sum_{p,q} \psi_p \psi_q \tilde{t}_{(i,p),(j,q)} \qquad \text{and} \qquad \tilde{t}^{(\text{aprx})}_{i,j} = \sum_{p,q} \psi_p \psi_q \tilde{t}^{(\text{aprx})}_{(i,p),(j,q)}.$$

Using the bound from (3.177):

$$|\tilde{t}_{i,j} - \tilde{t}^{(\text{aprx})}_{i,j}| \leq \sum_{p,q} \psi_p \psi_q |\tilde{t}_{(i,p),(j,q)} - \tilde{t}^{(\text{aprx})}_{(i,p),(j,q)}| \tag{3.178}$$

$$\leq \frac{1}{2}(d+1)^2 \max\left(\left|\tilde{T} - \tilde{T}^{(\text{aprx})}\right|\right) \leq 5n^6\beta\exp(-\alpha\Gamma^2/4). \tag{3.179}$$

The next task is to bound $|\tilde{t}^{(\text{aprx})}_{i,j} - t^{(\text{round})}_{i,j}|$. We will consider three separate cases. In each case, we will show that

$$\left|\tilde{t}^{(\text{aprx})}_{i,j} - t^{(\text{round})}_{i,j}\right| \leq \alpha f(\omega_{\min}). \tag{3.180}$$

Recall that $\tilde{T}^{(\text{aprx})} = R^{(\text{aprx})} T^{(\text{block})} R^{(\text{aprx})}$, so matrix $\tilde{T}^{(\text{aprx})}$ is block diagonal. This means that $\tilde{t}^{(\text{aprx})}_{(i,p),(j,q)} = 0$ unless $(i,p) = (j,q)$ or $\{i,j\} \in E$ and $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$. This will considerably simplify the sum

$$\tilde{t}^{(\text{aprx})}_{i,j} = \sum_{p,q} \psi_p \psi_q \tilde{t}^{(\text{aprx})}_{(i,p),(j,q)}. \tag{3.181}$$

Case 1. Diagonal element: $i = j$.

First note that if $p = 0$ or $p > i$, then the block containing (i, p) is just the single entry on the diagonal. In this case, $r^{(\text{aprx})}_{(i,p),(i,p)} = 1$ and $\tilde{t}^{(\text{aprx})}_{(i,p),(i,p)} = t_{(i,p),(i,p)}$.

Thus, when $i = j$, the sum (3.181) simplifies to

$$\tilde{t}^{(\text{aprx})}_{i,i} = \frac{1}{2} t_{(i,0),(i,0)} + \frac{1}{2d} \sum_{j:\{i,j\}\in E} \mathrm{ON}\left(R^{(\text{aprx})}_{i,j} T_{i,j} R^{(\text{aprx})}_{i,j}\right) + \frac{1}{2d} \sum_{p>i} t_{(i,p),(i,p)}. \tag{3.182}$$

The function t is defined in (3.42) so that $t_\beta(0) = t_{(i,0),(i,0)}$ and $t_\alpha(0) = t_{(i,p),(i,p)}$ for $p > 0$. Thus,

$$\tilde{t}^{(\text{aprx})}_{i,i} = \frac{1}{2} t_\beta(0) + \frac{1}{2d} \sum_{j:\{i,j\}\in E} \mathrm{ON}\left(R^{(\text{aprx})}_{i,j} T_{i,j} R^{(\text{aprx})}_{i,j}\right) + \frac{1}{2d} \sum_{p>i} t_\alpha(0). \tag{3.183}$$

Recall from (3.62) that the diagonal coefficients of $T^{(\text{round})}$ are

$$t^{(\text{round})}_{i,i} = c_T = \frac{1}{2}\left(t_\beta(0) + t_\alpha(0)\right) = \frac{1}{2} t_\beta(0) + \frac{1}{2d} \sum_{p>0} t_\alpha(0). \tag{3.184}$$

Therefore the difference between $\tilde{t}^{(\text{aprx})}_{i,j}$ and $t^{(\text{round})}_{i,j}$ is

$$\begin{aligned}
\left|\tilde{t}^{(\text{aprx})}_{i,i} - t^{(\text{round})}_{i,i}\right| &= \left|\frac{1}{2d} \sum_{j:\{i,j\}\in E} \mathrm{ON}\left(R^{(\text{aprx})}_{i,j} T_{i,j} R^{(\text{aprx})}_{i,j}\right) - t_\alpha(0)\right| && (3.185)\\
&\le \frac{1}{2d} \sum_{j:\{i,j\}\in E} \left|\mathrm{ON}\left(R^{(\text{aprx})}_{i,j} T_{i,j} R^{(\text{aprx})}_{i,j}\right) - t_\alpha(0)\right| && (3.186)\\
&\le \frac{1}{2d} \sum_{j:\{i,j\}\in E} \alpha \omega_{i,j} \epsilon^2_{i,j} \qquad \text{by (3.157)} && (3.187)\\
&\le \frac{\alpha}{2d} \sum_{j:\{i,j\}\in E} f(\omega_{i,j}) && (3.188)\\
&\le \frac{\alpha}{2d} \cdot d \cdot f(\omega_{\min}) \qquad \omega_{\min} \ge 2 && (3.189)\\
&= \frac{\alpha}{2} f(\omega_{\min}) \le \alpha f(\omega_{\min}). && (3.190)
\end{aligned}$$

Note that since $\omega_{\min} \ge 2$ (by the assumptions of the lemma), the function $f(\omega) = \omega^2 \exp(-\omega)$ is maximized at $\omega_{\min}$.

Case 2. Off-diagonal element corresponding to edge: $\{i, j\} \in E$.

In this case, there is exactly one p and exactly one q such that (i,p) and (j,q) are in the same block, where $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$. Thus, the summation in Eq. (3.181) has only one non-zero term:

$$\tilde{t}^{(\text{aprx})}_{i,j} = \frac{1}{2d} \textsc{Off}\left(R^{(\text{aprx})}_{i,j} T_{i,j} R^{(\text{aprx})}_{i,j}\right). \tag{3.191}$$

Recall from (3.63) that $t^{(\text{round})}_{i,j} = -\frac{\alpha}{4d}\sqrt{f(\omega_{i,j})}$.

Therefore

$$\begin{aligned}
\left|\tilde{t}^{(\text{aprx})}_{i,j} - t^{(\text{round})}_{i,j}\right| &= \frac{1}{2d}\left|\textsc{Off}\left(R^{(\text{aprx})}_{i,j} T_{i,j} R^{(\text{aprx})}_{i,j}\right) - \left(-\frac{\alpha}{2}\sqrt{f(\omega_{i,j})}\right)\right| && (3.192)\\
&\le \frac{1}{2d}\cdot\frac{\alpha}{2}\sqrt{f(\omega_{i,j})}\cdot 4\epsilon^2_{i,j} \qquad \text{by (3.158)} && (3.193)\\
&= \frac{\alpha}{d}\omega_{i,j}\exp(-3\omega_{i,j}/2) && (3.194)\\
&\le \alpha\omega_{i,j}\exp(-\omega_{i,j}) && (3.195)\\
&\le \alpha f(\omega_{\min}). && (3.196)
\end{aligned}$$

Again, we are using the fact that since $\omega_{\min} \ge 2$, the function $f(\omega) = \omega^2\exp(-\omega)$ is maximized at $\omega_{\min}$.

Case 3. Off-diagonal element corresponding to non-edge: $\{i,j\} \notin E$.

In this case, (i,p) and (j,q) are in different blocks for all p,q, and so the summation in Eq. (3.181) is empty. That is, $\tilde{t}^{(\text{aprx})}_{i,j} = 0$. Recall that $t^{(\text{round})}_{i,j}$ is also zero for $\{i,j\} \notin E$.

Finally, we can combine the bound for $\left|\tilde{t}_{i,j} - \tilde{t}^{(\text{aprx})}_{i,j}\right|$ from (3.179) and the bound for $\left|\tilde{t}^{(\text{aprx})}_{i,j} - t^{(\text{round})}_{i,j}\right|$ from (3.180):

$$\begin{aligned}
\left\|T^{(\text{ES})} - T^{(\text{round})}\right\| &\le \left\|\sum_{\substack{i,j\\ \sigma}}\left(\tilde{t}_{i,j} - t^{(\text{round})}_{i,j}\right)\tilde{a}^{\dagger}_{i,\sigma}\tilde{a}_{j,\sigma}\right\| && (3.197)\\
&\le \sum_{\substack{i,j\\ \sigma}}\left|\tilde{t}_{i,j} - t^{(\text{round})}_{i,j}\right|\left\|\tilde{a}^{\dagger}_{i,\sigma}\tilde{a}_{j,\sigma}\right\| = 2\sum_{i,j}\left|\tilde{t}_{i,j} - t^{(\text{round})}_{i,j}\right| && (3.198)\\
&\le 2\sum_{i,j}\left|\tilde{t}_{i,j} - \tilde{t}^{(\text{aprx})}_{i,j}\right| + 2\sum_{i,j}\left|\tilde{t}^{(\text{aprx})}_{i,j} - t^{(\text{round})}_{i,j}\right| && (3.199)\\
&\le 10n^8\beta\exp(-\alpha\Gamma^2/4) + 2n^2\alpha f(\omega_{\min}). && (3.200)
\end{aligned}$$

We can apply the conditions of the lemma to simplify this expression. The lower bound on $\alpha\Gamma^2$ implies that $\exp(-\alpha\Gamma^2/4) \le (10n^6\beta)^{-1}\cdot\exp(-\omega_{\min})$. Using the assumptions that $\alpha \ge 1$ and $\omega_{\min} \ge 1$:

$$10n^8\beta\exp(-\alpha\Gamma^2/4) \le n^2\exp(-\omega_{\min}) \le n^2\alpha(\omega_{\min})^2\exp(-\omega_{\min}) = n^2\alpha f(\omega_{\min}).$$

Recall that $f(\omega) = \omega^2 \exp(-\omega)$. The final bound for the kinetic-energy difference is

$$\left\|T^{(\mathrm{ES})} - T^{(\mathrm{round})}\right\| \leq 3n^2 \alpha f(\omega_{\min}). \tag{3.201}$$

Next, we consider the terms for the potential energy. As with the kinetic-energy terms, we will approximate $\tilde{U} = (R \otimes R)U(R \otimes R)$ by

$$\tilde{U}^{(\mathrm{aprx})} = (R^{(\mathrm{aprx})} \otimes R^{(\mathrm{aprx})})U^{(\mathrm{block})}(R^{(\mathrm{aprx})} \otimes R^{(\mathrm{aprx})}). \tag{3.202}$$

The matrices are now $n^2(d+1)^2 \times n^2(d+1)^2$. We will use the fact that if A and B are $m \times m$ matrices, then $\max(|AB|) \leq m \max(|A|) \cdot \max(|B|)$. Since $d+1 \leq n$, we pick up a factor of at most n^4 every time this principle is applied. We will bound $\max(|\tilde{U} - \tilde{U}^{(\mathrm{aprx})}|)$ in two stages. First we bound

$$\max\left(\left|\tilde{U} - (R \otimes R)U^{(\mathrm{block})}(R \otimes R)\right|\right) = \max\left(\left|(R \otimes R)U(R \otimes R) - (R \otimes R)U^{(\mathrm{block})}(R \otimes R)\right|\right) \tag{3.203}$$

$$= \max\left(\left|(R \otimes R)(U - U^{(\mathrm{block})})(R \otimes R)\right|\right) \tag{3.204}$$

$$\leq n^8 (r_{\max})^4 u_{\max}^{(\mathrm{neg})} \tag{3.205}$$

$$\leq n^8 \underbrace{2^4}_{3.161} \underbrace{2\beta^3/\Gamma}_{3.165} = 32 n^8 \beta^3/\Gamma. \tag{3.206}$$

The next step is to bound

$$\max\left(\left|(R \otimes R)U^{(\mathrm{block})}(R \otimes R) - \tilde{U}^{(\mathrm{aprx})}\right|\right) \tag{3.207}$$

$$= \max\left(\left|(R \otimes R)U^{(\mathrm{block})}(R \otimes R) - (R^{(\mathrm{aprx})} \otimes R^{(\mathrm{aprx})})U^{(\mathrm{block})}(R^{(\mathrm{aprx})} \otimes R^{(\mathrm{aprx})})\right|\right). \tag{3.208}$$

If we substitute $R = R^{(\mathrm{neg})} + R^{(\mathrm{aprx})}$ in to the expression $(R \otimes R)U^{(\mathrm{block})}(R \otimes R)$ and expand the product, we get the sum of 2^4 terms:

$$(R \otimes R)U^{(\mathrm{block})}(R \otimes R) = \sum_{a,b,c,d \in \{\mathrm{neg},\mathrm{aprx}\}} (R^{(\mathrm{a})} \otimes R^{(\mathrm{b})})U^{(\mathrm{block})}(R^{(\mathrm{c})} \otimes R^{(\mathrm{d})}). \tag{3.209}$$

In bounding the difference from (3.208), we are left with the terms in which a, b, c, d are not all equal to "aprx", so every remaining term will have at least one factor of $R^{(\text{neg})}$:

$$|(R \otimes R)U^{(\text{block})}(R \otimes R) - \tilde{U}^{(\text{aprx})}| \leq n^8 u_{\max} \sum_{x=0}^{3} \binom{4}{x} \left(r_{\max}^{(\text{aprx})}\right)^x \left(r_{\max}^{(\text{neg})}\right)^{4-x} \tag{3.210}$$

$$\leq n^8 u_{\max} \cdot 15 \cdot r_{\max}^{(\text{neg})} \left[\max\left\{r_{\max}^{(\text{aprx})}, r_{\max}^{(\text{neg})}\right\}\right]^3 \tag{3.211}$$

$$\leq n^8 \underbrace{2\beta^3}_{(3.164)} \cdot 15 \cdot \underbrace{n^2 \exp\left[-(\alpha\Gamma^2 - \omega_{\min})/2\right]}_{(3.149)} \cdot \underbrace{\left(\frac{3}{2}\right)^3}_{(3.160,3.150)} \tag{3.212}$$

$$\leq 102 n^{10} \beta^3 \exp\left[-(\alpha\Gamma^2 - \omega_{\min})/2\right] \tag{3.213}$$

$$\leq 102 n^{10} \beta^3 \exp(-\alpha\Gamma^2/4). \tag{3.214}$$

The last inequality uses the assumption from the lemma that $\alpha\Gamma^2 \geq 2\omega_{\min}$. Putting the two bounds from (3.206) and (3.214) together, we get that:

$$\max\left(\left|\tilde{U} - \tilde{U}^{(\text{aprx})}\right|\right) \tag{3.215}$$

$$\leq \max\left(\left|\tilde{U} - (R \otimes R)U^{(\text{block})}(R \otimes R)\right|\right) + \max\left(\left|(R \otimes R)U^{(\text{block})}(R \otimes R) - \tilde{U}^{(\text{aprx})}\right|\right) \tag{3.216}$$

$$\leq 32 n^8 \beta^3/\Gamma + 102 n^{10} \beta^3 \exp(-\alpha\Gamma^2/4). \tag{3.217}$$

Since the composite orbitals are superpositions of the primitive orbitals, the potential-energy terms for the composite orbitals can be expressed as linear combinations of the potential-energy terms for the primitive orbitals. Therefore

$$u_{i,j,k,l} = \sum_{p,q,r,s,\in[d+1]} \psi_p \psi_q \psi_r \psi_s u_{[(i,p),(j,q)][(l,r),(l,s)]}, \tag{3.218}$$

where the amplitudes ψ are defined to be $\psi_0 = 1/\sqrt{2}$ and $\psi_{p>0} = 1/\sqrt{2d}$. The same definition for $\tilde{u}_{i,j,k,l}$ and $\tilde{u}_{i,j,k,l}^{(\text{aprx})}$ can be applied using the potential-energy terms for the primitive orbitals defined in $\tilde{U}$ and $\tilde{U}^{(\text{aprx})}$. We can apply the bound from (3.217) to bound the difference in the potential-energy terms for the composite orbitals:

$$|\tilde{u}_{i,j,k,l} - \tilde{u}_{i,j,k,l}^{(\text{aprx})}| = \left|\sum_{p,q,r,s,\in[d+1]} \psi_p \psi_q \psi_r \psi_s \left(\tilde{u}_{[(i,p),(j,q)][(l,r),(l,s)]} - \tilde{u}_{[(i,p),(j,q)][(l,r),(l,s)]}^{(\text{aprx})}\right)\right| \tag{3.219}$$

$$\leq \frac{1}{4}(d+1)^4 \max\left(\left|\tilde{U} - \tilde{U}^{(\text{aprx})}\right|\right) \tag{3.220}$$

$$\leq \frac{1}{4} n^4 (32 n^8 \beta^3/\Gamma + 102 n^{10} \beta^3 \exp(-\alpha\Gamma^2/4)) \tag{3.221}$$

$$\leq 8 n^{12} \beta^3/\Gamma + 26 n^{14} \beta^3 \exp(-\alpha\Gamma^2/4). \tag{3.222}$$

We can now apply the assumptions of the lemma to simplify the above expression. The assumption that $\Gamma \geq 640n^{18}\beta^3$ implies that $8n^{12}\beta^3/\Gamma \leq 1/(80n^6)$. The assumption that $\alpha\Gamma^2 \geq 12\log\beta + 80\log n + 4\omega_{\min} + 24 \geq 12\log\beta + 80\log n + 40$ implies that $26n^{14}\beta^3\exp(-\alpha\Gamma^2/4) \leq 1/(80n^6)$. Therefore

$$8n^{12}\beta^3/\Gamma + 26n^{14}\beta^3\exp(-\alpha\Gamma^2/4) \leq \frac{1}{40n^6}. \tag{3.223}$$

The next task is to bound $|\tilde{u}^{(\text{aprx})}_{i,j,k,l} - u^{(\text{round})}_{i,j,k,l}|$. Since

$$\tilde{U}^{(\text{aprx})} = (R^{(\text{aprx})} \otimes R^{(\text{aprx})})U^{(\text{block})}(R^{(\text{aprx})} \otimes R^{(\text{aprx})})$$

is block diagonal, many of the terms in the sum (3.218) will be zero. We consider three cases. In each case, we will show that

$$|\tilde{u}^{(\text{aprx})}_{i,j,k,l} - \tilde{u}^{(\text{round})}_{i,j,k,l}| \leq 4\sqrt{\alpha} \cdot \exp\left(-\omega_{\min}/2\right). \tag{3.224}$$

Case 1. Onsite term $i = j = k = l$.

Note that the entry in row $[(i,p),(i,q)]$ and row $[(i,r),(i,s)]$ is outside of a block unless $p = q = r = s$. If $p = 0$ or $p > i$, then the block containing $[(i,p),(i,p)]$ is just the single entry on the diagonal. In this case, $r^{(\text{aprx})}_{(i,p),(i,p)} \otimes r^{(\text{aprx})}_{(i,p),(i,p)} = 1$ and the diagonal element at $[(i,p),(i,p)]$ is the same for $\tilde{U}^{(\text{aprx})}$ and U. If $p = 0$, then primitive orbital $\phi_{i,0}$ is a Gaussian of width β and the diagonal term of U at $[(i,0),(i,0)]$ is as in Eq. (3.45) defined as $u^{(\text{Coul})}_{\beta}(0)$. For $p > i$, then primitive orbital $\phi_{i,p}$ is a Gaussian of width α and the diagonal term of U at $[(i,p),(i,p)]$ is as in Eq. (3.45) defined as $u^{(\text{Coul})}_{\alpha}(0)$.

Thus, when $i = j = k = l$, the sum (3.218) simplifies to

$$\tilde{u}^{(\text{aprx})}_{i,i,i,i} = \frac{1}{4d^2}\sum_{0<p\leq i}\left(\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2} U_{i,j}\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2}\right)_{(i,p),(i,p),(i,p),(i,p)} \tag{3.225}$$

$$+\frac{1}{4}u^{(\text{Coul})}_{\beta}(0) + \frac{1}{4d^2}\sum_{p>i} u^{(\text{Coul})}_{\alpha}(0). \tag{3.226}$$

Recall that $u^{(\text{round})}_{i,i,i,i}$ is defined in (3.72) to be

$$c^{(\text{round})}_{U} = \frac{1}{4}u^{(\text{Coul})}_{\beta}(0) + \frac{1}{4d}u^{(\text{Coul})}_{\alpha}(0) = \frac{1}{4}u^{(\text{Coul})}_{\beta}(0) + \frac{1}{4d^2}\sum_{p\in[d]} u^{(\text{Coul})}_{\alpha}(0) \tag{3.227}$$

Therefore,

$$\left|\tilde{u}^{(\text{aprx})}_{i,i,i,i} - u^{(\text{round})}_{i,i,i,i}\right| \tag{3.228}$$

$$= \frac{1}{4d^2}\left|\sum_{0<p\leq i}\left(\left(\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2}U_{i,j}\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2}\right)_{(i,p),(i,p),(i,p),(i,p)}\right) - u^{(\text{Coul})}_{\alpha}(0)\right| \tag{3.229}$$

$$= \frac{1}{4d^2}\left|\sum_{0<p\leq i}\left(\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2}U_{i,j}\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2} - U_{i,j}\right)_{(i,p),(i,p),(i,p),(i,p)}\right| \tag{3.230}$$

$$\leq \frac{1}{4d^2}\cdot d\cdot \max\left(\left|\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2}U_{i,j}\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2} - U_{i,j}\right)\right| \tag{3.231}$$

$$\leq \frac{1}{4}\underbrace{16\sqrt{\alpha}\epsilon_{\max}}_{(3.159)} = 4\sqrt{\alpha}\cdot\exp(-\omega_{i,j}/2) \leq 4\sqrt{\alpha}\cdot\exp(-\omega_{\min}/2). \tag{3.232}$$

The last inequality uses the assumption of the lemma that $\omega_{\min} \geq 2$.

Case 2. All indices within block corresponding to edge $\{i,j\} \in E$.

Let $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$. Consider, for example, the term $\tilde{u}^{(\text{aprx})}_{i,j,j,i}$. The sum in (3.218) has only one non-zero term corresponding to row $[(i,p),(j,q)]$ and column $[(j,q),(i,p)]$. So

$$\tilde{u}^{(\text{aprx})}_{i,j,j,i} = \frac{1}{4d^2}\left(\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2}U_{i,j}\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2}\right)_{[(i,p),(j,q)],[(j,q),(i,p)]} \tag{3.233}$$

Recall that

$$u^{(\text{round})}_{i,j,j,i} = \frac{1}{4d^2}u^{(\text{Coul})}_{\alpha}(\gamma_{i,j}) = \frac{1}{4d^2}u_{[(i,p),(j,q)],[(j,q),(i,p)]} \tag{3.234}$$

The first equality comes from the definition of $u^{(\text{round})}$ in (3.73) and the second comes from the definition of $u^{(\text{Coul})}_{\alpha}(\gamma_{i,j})$ in (3.45). The entry in row $[(i,p),(j,q)]$ and column $[(j,q),(i,p)]$ is inside the block corresponding to edge $\{i,j\}$. Therefore,

$$\left|\tilde{u}^{(\text{aprx})}_{i,j,j,i} - u^{(\text{round})}_{i,j,j,i}\right| = \frac{1}{4d^2}\max\left(\left|\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2}U_{i,j}\left(R^{(\text{block})}_{i,j}\right)^{\otimes 2} - U_{i,j}\right|\right) \tag{3.235}$$

$$\leq \frac{1}{4}\underbrace{16\sqrt{\alpha}\epsilon_{i,j}}_{(3.159)} = 4\sqrt{\alpha}\cdot\exp(-\omega_{i,j}/2) \leq 4\sqrt{\alpha}\cdot\exp(-\omega_{\min}/2). \tag{3.236}$$

The same bound holds for $\left|\tilde{u}^{(\text{aprx})}_{i,i,j,i} - u^{(\text{round})}_{i,i,j,i}\right|$, $\left|\tilde{u}^{(\text{aprx})}_{j,i,j,i} - u^{(\text{round})}_{j,i,j,i}\right|$, et cetera for $\{i,j\} \in E$.

Case 3. At least one pair of indices corresponding to non-edge $\{i,j\} \notin E$.

In this case, both $\tilde{u}^{(\mathrm{aprx})}_{i,j,k,l}$ and $u^{(\mathrm{round})}_{i,j,k,l}$ are zero. All together,

$$\begin{aligned}
\left\|U^{(\mathrm{ES})} - U^{(\mathrm{round})}\right\| &\leq \left\| \frac{1}{2} \sum_{\substack{i,j,k,l \\ \sigma,\tau}} \left(\tilde{u}_{i,j,k,l} - u^{(\mathrm{round})}_{i,j,k,l} \right) \tilde{a}^\dagger_{i,\sigma} \tilde{a}^\dagger_{j,\sigma} \tilde{a}_{k,\sigma} \tilde{a}_{l,\sigma} \right\| && (3.237) \\
&\leq \frac{1}{2} \sum_{\substack{i,j,k,l \\ \sigma,\tau}} \left| \tilde{u}_{i,j,k,l} - u^{(\mathrm{round})}_{i,j,k,l} \right| \left\| \tilde{a}^\dagger_{i,\sigma} \tilde{a}^\dagger_{j,\sigma} \tilde{a}_{k,\sigma} \tilde{a}_{l,\sigma} \right\| && (3.238) \\
&= \frac{1}{2} \cdot \underbrace{4}_{\sigma,\tau} \sum_{i,j,k,l} \left| \tilde{u}_{i,j,k,l} - u^{(\mathrm{round})}_{i,j,k,l} \right| && (3.239) \\
&\leq 2 \sum_{i,j,k,l} \left| \tilde{u}_{i,j,k,l} - \tilde{u}^{(\mathrm{aprx})}_{i,j,k,l} \right| + 2 \sum_{i,j,k,l} \left| \tilde{u}^{(\mathrm{aprx})}_{i,j,k,l} - u^{(\mathrm{round})}_{i,j,k,l} \right| && (3.240) \\
&\leq 2n^4 \underbrace{\left(\frac{1}{40n^6} \right)}_{(3.223)} + 2n^4 \cdot \underbrace{4\sqrt{\alpha} \cdot \exp(-\omega_{\min}/2)}_{(3.224)} && (3.241) \\
&= \frac{1}{20n^2} + 8n^4 \sqrt{\alpha} \cdot \exp(-\omega_{\min}/2) && (3.242)
\end{aligned}$$

Eqs. (3.201) and (3.242) imply the lemma:

$$\begin{aligned}
\left\|H^{(\mathrm{ES})} - H^{(\mathrm{round})}\right\| &\leq \left\|T^{(\mathrm{ES})} - T^{(\mathrm{round})}\right\| + \left\|U^{(\mathrm{ES})} - U^{(\mathrm{round})}\right\| && (3.243) \\
&\leq 3n^2 \alpha f(\omega_{\min}) + \frac{1}{20n^2} + 8n^4 \sqrt{\alpha} \cdot \exp(-\omega_{\min}/2). && (3.244)
\end{aligned}$$

□

3.6.1 Proof of Lemma 4

Proof of Lemma 4. Define $\epsilon_{\max}$ to be the largest off-diagonal element of S and ϵ_{neg} to be the largest entry of $S^{(\mathrm{neg})}$, which is the largest entry of S outside of an edge block:

$$\begin{aligned}
\epsilon_{\max} &= \max_{(i,p) \neq (j,q)} s_{(i,p),(j,q)} = s_\alpha(\gamma_{\min}) = \exp(-\omega_{\min}/2), && (3.245) \\
\epsilon_{\mathrm{neg}} &= \max\left(S^{(\mathrm{neg})}\right) \leq s_\alpha(\Gamma) = \exp(-\alpha\Gamma^2/2). && (3.246)
\end{aligned}$$

Let $S^{(\mathrm{block})} = I + S^{(\mathrm{OD})}$; $S^{(\mathrm{OD})}$ has at most one entry per row or column, and that entry is between 0 and $\epsilon_{\max}$. Using the Taylor expansion

$$M^{-1/2} = \sum_{k=0}^{\infty} (-2)^{-k} \frac{(2k-1)!!}{k!} (M - I)^k \qquad (3.247)$$

of a matrix M around the identity I, we have

$$\begin{aligned} R^{(\mathrm{neg})} &= R - R^{(\mathrm{aprx})} && (3.248)\\ &= (S)^{-1/2} - \left(S^{(\mathrm{block})}\right)^{-1/2} && (3.249)\\ &= \left(I + S^{(\mathrm{OD})} + S^{(\mathrm{neg})}\right)^{-1/2} - \left(I + S^{(\mathrm{OD})}\right)^{-1/2} && (3.250)\\ &= \sum_{k=0}^{\infty} (-2)^{-k} \frac{(2k-1)!!}{k!} \left[\left(S^{(\mathrm{OD})} + S^{(\mathrm{neg})}\right)^k - \left(S^{(\mathrm{OD})}\right)^k\right]. && (3.251) \end{aligned}$$

Entrywise,

$$\begin{aligned} &\left[\left(S^{(\mathrm{OD})} + S^{(\mathrm{neg})}\right)^k\right]_{(i_0,l_0),(i_k,l_k)} && (3.252)\\ &= \sum_{(i_1,l_1),\ldots,(i_{k-1},l_{k-1})} \left(S^{(\mathrm{OD})} + S^{(\mathrm{neg})}\right)_{(i_0,l_0),(i_1,l_1)} \cdots \left(S^{(\mathrm{OD})} + S^{(\mathrm{neg})}\right)_{(i_{k-1},l_{k-1}),(i_k,l_k)} && (3.253)\\ &= \sum_{0<\left\|\mathbf{x}_{i_{k'},l_{k'}} - \mathbf{x}_{i_{k'+1},l_{k'+1}}\right\|} s_{(i_0,l_0),(i_1,l_1)} \cdots s_{(i_{k-1},l_{k-1}),(i_k,l_k)}, && (3.254) \end{aligned}$$

where the summation excludes the diagonal entries. (Recall that I, $S^{(\mathrm{OD})}$, and $S^{(\mathrm{neg})}$ have disjoint support.) Similarly,

$$\left[\left(S^{(\mathrm{OD})}\right)^k\right]_{(i_0,l_0),(i_k,l_k)} = \sum_{0<\left\|\mathbf{x}_{i_{k'},l_{k'}} - \mathbf{x}_{i_{k'+1},l_{k'+1}}\right\|<\Gamma} s_{(i_0,l_0),(i_1,l_1)} \cdots s_{(i_{k-1},l_{k-1}),(i_k,l_k)}, \qquad (3.255)$$

where the summation excludes both the diagonal and anything outside of the blocks. The difference between Eq. (3.254) and Eq. (3.255) is the summation in Eq. (3.254) restricted to when at least one of the neighboring pairs is at least Γ separated. Each term with exactly x pairs separated by at least Γ contributes at most $\epsilon_{\max}^{k-x}\epsilon_{\mathrm{neg}}^{x}$. There are $\binom{k}{x}$ places in the sequence that these pairs can occur. For each factor contributing more than ϵ_{neg} there is at most one index value $(i_{k'}, l_{k'})$, and for each factor contributing at most ϵ_{neg} there are at most

$n(d+1)-1 \leq 2n^2$ indices. Therefore

$$\left[\left(S^{(\mathrm{OD})}+S^{(\mathrm{neg})}\right)^k-\left(S^{(\mathrm{OD})}\right)^k\right]_{(i_0,l_0),(i_k,l_k)} \tag{3.256}$$

$$\leq \sum_{x=1}^{k}\binom{k}{x}\left(2n^2\epsilon_{\mathrm{neg}}\right)^x\epsilon_{\max}^{k-x} \tag{3.257}$$

$$= \sum_{x=0}^{k-1}\binom{k}{x}\left(2n^2\epsilon_{\mathrm{neg}}\right)^{k-x}\epsilon_{\max}^{x} = \left(2n^2\epsilon_{\mathrm{neg}}\right)^k\sum_{x=0}^{k-1}\frac{k}{k-x}\binom{k-1}{x}\left(\frac{\epsilon_{\max}}{2n^2\epsilon_{\mathrm{neg}}}\right)^x \tag{3.258}$$

$$\leq \left(2n^2\epsilon_{\mathrm{neg}}\right)^k\sum_{x=0}^{k-1}k\binom{k-1}{x}\left(\frac{\epsilon_{\max}}{2n^2\epsilon_{\mathrm{neg}}}\right)^x = k\left(2n^2\epsilon_{\mathrm{neg}}\right)^k\left(1+\frac{\epsilon_{\max}}{2n^2\epsilon_{\mathrm{neg}}}\right)^{k-1} \tag{3.259}$$

$$= k\left(2n^2\epsilon_{\mathrm{neg}}\right)\left(2n^2\epsilon_{\mathrm{neg}}+\epsilon_{\max}\right)^{k-1} \tag{3.260}$$

$$\leq k\left(2n^2\epsilon_{\mathrm{neg}}\right)(2\epsilon_{\max})^{k-1} = kn^2\frac{\epsilon_{\mathrm{neg}}}{\epsilon_{\max}}(2\epsilon_{\max})^k, \tag{3.261}$$

where we used the fact that

$$2n^2\epsilon_{\mathrm{neg}} = 2n^2e^{-\alpha\Gamma^2/2} \leq e\cdot n^2e^{-\alpha\Gamma^2/2} \leq e\cdot n^2e^{-(4\log n+\omega_{\min}+2)/2} = e^{-\omega_{\min}/2} = \epsilon_{\max} \tag{3.262}$$

by assumption. Returning to the expression in Eq. (3.251), the norm of each entry of $R^{(\mathrm{neg})}$ then is

$$\left|r^{(\mathrm{neg})}_{(i_0,l_0),(i_k,l_k)}\right| = \left|\sum_{k=0}^{\infty}(-2)^{-k}\frac{(2k-1)!!}{k!}\left[\left(S^{(\mathrm{OD})}+S^{(\mathrm{neg})}\right)^k-\left(S^{(\mathrm{OD})}\right)^k\right]_{(i_0,l_0),(i_k,l_k)}\right| \tag{3.263}$$

$$\leq \sum_{k=0}^{\infty}(2)^{-k}\frac{(2k-1)!!}{k!}\left|\left[\left(S^{(\mathrm{OD})}+S^{(\mathrm{neg})}\right)^k-\left(S^{(\mathrm{OD})}\right)^k\right]_{(i_0,l_0),(i_k,l_k)}\right| \tag{3.264}$$

$$\leq n^2\frac{\epsilon_{\mathrm{neg}}}{\epsilon_{\max}}\sum_{k=1}^{\infty}(2)^{-k}\frac{(2k-1)!!}{k!}k(2\epsilon_{\max})^k \tag{3.265}$$

$$= n^2\frac{\epsilon_{\mathrm{neg}}}{\epsilon_{\max}}\sum_{k=1}^{\infty}\frac{(2k-1)!!}{(k-1)!}\epsilon_{\max}^k \tag{3.266}$$

$$= n^2\frac{\epsilon_{\mathrm{neg}}}{\epsilon_{\max}}\frac{\epsilon_{\max}}{(1-2\epsilon_{\max})^{3/2}} \tag{3.267}$$

$$\leq n^2\frac{\epsilon_{\mathrm{neg}}}{\epsilon_{\max}} \qquad \text{for } \omega_{\min}\geq 4 \tag{3.268}$$

$$\leq n^2\exp\left[-(\alpha\Gamma^2-\omega_{\min})/2\right]. \tag{3.269}$$

□

3.6.2 Proof of Lemma 5

Proof of Lemma 5. In this proof, we'll use the following form of Taylor's Theorem.

Theorem 9 (Taylor's Theorem with remainder in Lagrange form [5])**.** Let f be a $(n+1)$-times differentiable function in the region $[0,1]$. Then for every $x \in [0,1]$ there is some $c \in [0,x]$ such that

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(0)}{k!} x^k + \frac{f^{(n+1)}(c)}{(n+1)!} x^{n+1}. \tag{3.270}$$

Note that $\omega_{\min} \geq 2$ implies that $\epsilon_{i,j} = e^{-\omega_{i,j}/2} \leq e^{-\omega_{\min}/2} \leq 1/e$.

We'll start with the bounds on the entries of $R^{(\text{aprx})}$. Given the derivatives

$$\frac{d}{d\epsilon}\left(\frac{1}{\sqrt{1+\epsilon}} \pm \frac{1}{\sqrt{1-\epsilon}}\right) = \frac{1}{2}\left(-(1+\epsilon)^{-3/2} \pm (1-\epsilon)^{-3/2}\right), \tag{3.271}$$

$$\frac{d^2}{d\epsilon^2}\left(\frac{1}{\sqrt{1+\epsilon}} \pm \frac{1}{\sqrt{1-\epsilon}}\right) = \frac{3}{4}\left((1+\epsilon)^{-5/2} \pm (1-\epsilon)^{-5/2}\right), \tag{3.272}$$

$$\frac{d^3}{d\epsilon^3}\left(\frac{1}{\sqrt{1+\epsilon}} \pm \frac{1}{\sqrt{1-\epsilon}}\right) = \frac{15}{8}\left(-(1+\epsilon)^{-7/2} \pm (1-\epsilon)^{-7/2}\right), \tag{3.273}$$

Theorem 9 implies that

$$\frac{1}{\sqrt{1+\epsilon}} + \frac{1}{\sqrt{1-\epsilon}} = 2 + 0 + \frac{1}{2}\frac{3}{4}\left((1+\epsilon')^{-5/2} + (1-\epsilon')^{-5/2}\right)\epsilon^2, \tag{3.274}$$

$$\frac{1}{\sqrt{1+\epsilon}} - \frac{1}{\sqrt{1-\epsilon}} = 0 - \epsilon + 0 - \frac{1}{3!}\frac{15}{8}\left((1+\epsilon')^{-7/2} + (1-\epsilon')^{-7/2}\right)\epsilon^3 \tag{3.275}$$

for some $\epsilon' \in [0,\epsilon]$. For $0 \leq \epsilon \leq 1/e$, we have

$$2 \leq \frac{1}{\sqrt{1+\epsilon}} + \frac{1}{\sqrt{1-\epsilon}} \leq 2 + 2\epsilon^2, \tag{3.276}$$

$$-\epsilon - 2\epsilon^3 \leq \frac{1}{\sqrt{1+\epsilon}} - \frac{1}{\sqrt{1-\epsilon}} \leq -\epsilon. \tag{3.277}$$

Dividing by 2 and substituting $\epsilon = \epsilon_{i,j}$ implies Eqs. (3.155) and (3.156).

Now, let's turn to the entries of $R_{i,j}^{(\text{aprx})} T_{i,j} R_{i,j}^{(\text{aprx})}$. Let $\mathcal{B}(i,j) = \{(i,p),(j,q)\}$. To make the notation more concise within this proof, we will refer to the diagonal elements of $R_{i,j}^{(\text{aprx})}$ as $r_{ON} = r_{(i,p),(i,p)}^{(\text{aprx})} = r_{(j,q),(j,q)}^{(\text{aprx})}$ and the off-diagonal elements as $r_{OFF} = r_{(i,p),(j,q)}^{(\text{aprx})} = r_{(j,q),(i,p)}^{(\text{aprx})}$. Note that

$$(r_{ON})^2 + (r_{OFF})^2 = \frac{1}{4}\left(\left(\frac{1}{\sqrt{1+\epsilon_{i,j}}} + \frac{1}{\sqrt{1-\epsilon_{i,j}}}\right)^2 + \left(\frac{1}{\sqrt{1+\epsilon_{i,j}}} - \frac{1}{\sqrt{1-\epsilon_{i,j}}}\right)^2\right) \tag{3.278}$$

$$= \frac{1}{2}\left(\frac{1}{1+\epsilon_{i,j}} + \frac{1}{1-\epsilon_{i,j}}\right) = \frac{1}{1-\epsilon_{i,j}^2} \tag{3.279}$$

and

$$r_{ON} \cdot r_{OFF} = \frac{1}{4}\left(\frac{1}{\sqrt{1+\epsilon_{i,j}}} + \frac{1}{\sqrt{1-\epsilon_{i,j}}}\right)\left(\frac{1}{\sqrt{1+\epsilon_{i,j}}} - \frac{1}{\sqrt{1-\epsilon_{i,j}}}\right) \tag{3.280}$$

$$= \frac{1}{4}\left(\frac{1}{1+\epsilon_{i,j}} - \frac{1}{1-\epsilon_{i,j}}\right) = -\frac{\epsilon_{i,j}}{2(1-\epsilon_{i,j}^2)}. \tag{3.281}$$

The diagonal entries of $T_{i,j}$ are $t_{(i,p),(i,p)} = t_{(j,q),(j,q)} = t_\alpha(0)$. and the off-diagonal entries are $t_{(i,p),(j,q)} = t_{(j,q),(i,p)} = t_\alpha(\gamma_{i,j})$. Then the diagonal of $R_{i,j}^{(\text{aprx})} T_{i,j} R_{i,j}^{(\text{aprx})}$ entry is

$$\text{ON}\left(R_{i,j}^{(\text{aprx})} T_{i,j} R_{i,j}^{(\text{aprx})}\right) \tag{3.282}$$

$$= \begin{pmatrix} r_{ON} & r_{OFF} \end{pmatrix} \begin{pmatrix} t_\alpha(0) & t_\alpha(\gamma_{i,j}) \\ t_\alpha(\gamma_{i,j}) & t_\alpha(0) \end{pmatrix} \begin{pmatrix} r_{ON} \\ r_{OFF} \end{pmatrix} \tag{3.283}$$

$$= t_\alpha(0)\left((r_{ON})^2 + (r_{OFF})^2\right) + 2t_\alpha(\gamma_{i,j}) r_{ON} r_{OFF} \tag{3.284}$$

$$= \frac{t_\alpha(0)}{1-\epsilon_{i,j}^2} - \frac{t_\alpha(\gamma_{i,j})\epsilon_{i,j}}{1-\epsilon_{i,j}^2} \tag{3.285}$$

$$= \left(t_\alpha(0) - s_\alpha(\gamma_{i,j}) t_\alpha(\gamma_{i,j})\right) \frac{1}{1-\epsilon_{i,j}^2} \tag{3.286}$$

$$= \left(\frac{3}{2}\alpha - \frac{1}{2}\alpha(3-\omega_{i,j})\exp(-\omega_{i,j})\right) \frac{1}{1-\epsilon_{i,j}^2} \tag{3.287}$$

$$= \left(\frac{3}{2}\alpha(1-\epsilon_{i,j}^2) + \frac{1}{2}\alpha\omega_{i,j}\epsilon_{i,j}^2\right) \frac{1}{1-\epsilon_{i,j}^2} \tag{3.288}$$

$$= t_\alpha(0) + \frac{\alpha\omega_{i,j}\epsilon_{i,j}^2}{2(1-\epsilon_{i,j}^2)} \tag{3.289}$$

and the off-diagonal entry of $R_{i,j}^{(\text{aprx})} T_{i,j} R_{i,j}^{(\text{aprx})}$ is

$$\begin{aligned}
&\textsc{Off}\left(R_{i,j}^{(\text{aprx})} T_{i,j} R_{i,j}^{(\text{aprx})}\right) &(3.290)\\
&= \begin{pmatrix} r_{ON} & r_{OFF} \end{pmatrix} \begin{pmatrix} t_\alpha(0) & t_\alpha(\gamma_{i,j}) \\ t_\alpha(\gamma_{i,j}) & t_\alpha(0) \end{pmatrix} \begin{pmatrix} r_{OFF} \\ r_{ON} \end{pmatrix} &(3.291)\\
&= 2t_\alpha(0) \cdot r_{ON} \cdot r_{OFF} + t_\alpha(\gamma_{i,j}) \left((r_{ON})^2 + (r_{OFF})^2 \right) &(3.292)\\
&= -\frac{t_\alpha(0)\epsilon_{i,j}}{1-\epsilon_{i,j}^2} + \frac{t_\alpha(\gamma_{i,j})}{1-\epsilon_{i,j}^2} &(3.293)\\
&= \left(t_\alpha(\gamma_{i,j}) - s_\alpha(\gamma_{i,j}) t_\alpha(0)\right) \frac{1}{1-\epsilon_{i,j}^2} &(3.294)\\
&= \left(\frac{1}{2}\alpha(3-\omega_{i,j}) \exp(-\omega_{i,j}/2) - \frac{3}{2}\alpha \exp(-\omega_{i,j}/2) \right) \frac{1}{1-\epsilon_{i,j}^2} &(3.295)\\
&= -\frac{1}{2}\alpha\omega_{i,j} \exp(-\omega_{i,j}/2) \frac{1}{1-\epsilon_{i,j}^2} &(3.296)\\
&= -\frac{1}{2}\alpha \sqrt{f(\omega_{i,j})} \frac{1}{1-\epsilon_{i,j}^2}. &(3.297)
\end{aligned}$$

Let's look at this factor $(1-\epsilon^2)^{-1}$. It's always at least 1, and its first two derivatives are

$$\frac{d}{d\epsilon}\left(\frac{1}{1-\epsilon^2}\right) = \frac{2\epsilon}{(1-\epsilon^2)^2}, \tag{3.298}$$

$$\frac{d^2}{d\epsilon^2}\left(\frac{1}{1-\epsilon^2}\right) = \frac{2(1+3\epsilon^2)}{(1-\epsilon^2)^3}. \tag{3.299}$$

By Theorem 9,

$$\frac{1}{1-\epsilon^2} = 1 + \frac{(1+3\epsilon'^2)}{(1-\epsilon'^2)^3}\epsilon^2 \tag{3.300}$$

for some $0 \le \epsilon' \le \epsilon$. For $\epsilon \le 1/e$, we have

$$1 \le \frac{1}{1-\epsilon^2} \le 1 + 4\epsilon^2 \le 2. \tag{3.301}$$

Combining Eqs. (3.289), (3.297) and (3.301) implies Eqs. (3.157) and (3.158).

Finally, we turn to the bound for

$$\max \left| \left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} U_{i,j} \left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} - U_{i,j} \right|$$

Each entry of $\left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2}$ is a product of two terms from $\{r_{ON}, r_{OFF}\}$, and only the diagonal terms are $(r_{ON})^2$. For notational ease, we will index the four rows and columns of $U_{i,j}$ by

$\{0,1,2,3\}$. For $a,b \in \{0,1,2,3\}$, we will denote the entry in row a and column b by $U_{i,j}[a,b]$. Now consider a particular entry in row a and column b of $\left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} U_{i,j} \left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2}$. This entry is the sum of 16 terms, each of which is a product of one entry from $U_{i,j}$ and four factors from $\{r_{ON}, r_{OFF}\}$. The only term that has four factors of (r_{ON}) is $(r_{ON})^4 U_{i,j}[a,b]$ because the two factors of $(r_{ON})^2$ must come from diagonal entries of $\left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2}$. The other 15 terms all have at least one factor of r_{OFF}. Also since $|r_{OFF}| < |r_{ON}|$, each of these other terms is at most $|r_{OFF}||r_{ON}|^3 \cdot \max(U_{i,j})$. Therefore we have:

$$\left| \left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} U_{i,j} \left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} [a,b] - U_{i,j}[a,b] \right| \tag{3.302}$$

$$\leq \left| (r_{ON})^4 U_{i,j}[a,b] + 15 \cdot |r_{OFF}||r_{ON}|^3 \max(U_{i,j}) - U_{i,j}[a,b] \right| \tag{3.303}$$

The maximum entry in $U_{i,j}$ is $u_\alpha^{(\text{Coul})}(0)$. Therefore

$$\max \left| \left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} U_{i,j} \left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} - U_{i,j} \right| \leq u_\alpha^{(\text{Coul})}(0) \left[(r_{ON})^4 - 1 + 15 \cdot |r_{OFF}| \cdot |r_{ON}|^3 \right]$$

Using the bounds from Eqs. (3.155) and (3.156), we know that $|r_{OFF}| \leq \epsilon_{i,j}(1/2 + \epsilon_{i,j}^2)$ and $|r_{ON}| \leq 1 + \epsilon_{i,j}^2$. Also $\epsilon_{i,j} = \exp(\omega_{i,j}/2)$ and since by assumption $\omega_{i,j} \geq 4$, $\epsilon \leq 1/4$.

$$\max \left| \left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} U_{i,j} \left(R_{i,j}^{(\text{aprx})}\right)^{\otimes 2} - U_{i,j} \right| \tag{3.304}$$

$$\leq u_\alpha^{(\text{Coul})}(0) \left[(r_{ON})^4 - 1 + 15|r_{OFF}| \cdot |r_{ON}|^3 \right] \tag{3.305}$$

$$\leq u_\alpha^{(\text{Coul})}(0) \left[(1+\epsilon_{i,j}^2)^4 - 1 + 15\epsilon_{i,j}(1/2 + \epsilon_{i,j}^2) \cdot (1+\epsilon_{i,j}^2)^3 \right] \tag{3.306}$$

$$\leq u_\alpha^{(\text{Coul})}(0) \left[2\epsilon_{i,j} + 15 \left(\frac{9}{16}\right) \cdot \left(\frac{17}{16}\right)^3 \epsilon_{i,j} \right] \tag{3.307}$$

$$\leq u_\alpha^{(\text{Coul})}(0)\epsilon_{i,j} \cdot 12 = \frac{2}{\sqrt{\pi}} 13\sqrt{\alpha}\epsilon_{i,j} \leq 16\sqrt{\alpha}\epsilon_{i,j} \tag{3.308}$$

□

3.6.3 Proof of Lemma 6

Proof of Lemma 6. We'll start with the kinetic coefficient bounds. Each $t_{(i,p),(j,q)}$ coefficient has one of the following three forms (from Eq. (3.41)), depending on whether orbitals $\phi_{i,p}$

and $\phi_{j,q}$ have exponent α or β, and where x is the distance between the two orbitals:

$$t_{\alpha,\beta}(x) = 2^{3/2}\frac{(\alpha\beta)^{7/4}}{(\alpha+\beta)^{5/2}}\left(3-2\mu x^2\right)\exp\left(-\mu x^2\right) \qquad \text{where } \mu = \alpha\beta/(\alpha+\beta) \tag{3.309}$$

$$t_{\alpha}(x) = \frac{\alpha}{2}\left(3-\alpha x^2\right)\exp\left(-\alpha x^2/2\right) \tag{3.310}$$

$$t_{\beta}(x) = \frac{\beta}{2}\left(3-\beta x^2\right)\exp\left(-\beta x^2/2\right) \tag{3.311}$$

Consider the prefactor:

$$2^{3/2}\frac{(\alpha\beta)^{7/4}}{(\alpha+\beta)^{5/2}} = \frac{\sqrt{\alpha\beta}}{2}\left[\frac{\sqrt{\alpha\beta}}{\left(\frac{\alpha+\beta}{2}\right)}\right]^{5/2} \leq \frac{\sqrt{\alpha\beta}}{2} \tag{3.312}$$

The inequality follows from the fact that the geometric mean of two positive numbers is no more than their arithmetic mean. Therefore, since $\beta \geq \alpha$, the maximum prefactor for $t_\alpha(x)$, $t_\beta(x)$, or $t_{\alpha,\beta}(x)$ is

$$\max\left\{\frac{\alpha}{2}, \frac{\beta}{2}, \frac{\sqrt{\alpha\beta}}{2}\right\} = \frac{\beta}{2}. \tag{3.313}$$

The part of the function t that depends on x is

$$\bar{t}_\mu(x) = (3-2\mu x^2)\exp(\mu x^2),$$

where $\mu = \alpha\beta/(\alpha+\beta)$ or $\alpha/2$ or $\beta/2$. Note that $\bar{t}_\mu(x)$ changes sign once, from positive to negative, at $2\mu x^2 = 3$. Its derivative,

$$\bar{t}'_\mu(x) = 2^{3/2}\mu x(4\mu x^2 - 10)\exp\left(-\mu x^2\right), \tag{3.314}$$

vanishes only at the origin and $2\mu x^2 = 5$, where it goes from negative to positive. Therefore,

$$\max_{x\geq 0}|\bar{t}_\mu(x)| = \max\left\{\bar{t}_\mu(0), -\bar{t}_\mu(\sqrt{5/2\mu})\right\} = \bar{t}_\mu(0) = 3. \tag{3.315}$$

Putting this together with the bound on the prefactor from (3.313), we get that $t_{\max} \leq \frac{3}{2}\beta$.

$\mu \geq \alpha/2$, and therefore $2\mu\Gamma^2 \geq \alpha\Gamma^2$. Since, by assumption, $\alpha\Gamma^2 \geq 5$, we know that $\bar{t}_\mu(x)$ is monotonic for $x \geq \Gamma$. Therefore

$$\max_{x\geq\Gamma}|\bar{t}_\mu(x)| = |\bar{t}_\mu(\Gamma)| \tag{3.316}$$

$$= \left|3-2\mu\Gamma^2\right|\exp\left(-\mu\Gamma^2\right) \tag{3.317}$$

$$\leq 2\mu\Gamma^2\exp\left(-\mu\Gamma^2\right) \tag{3.318}$$

$$\leq 2\exp\left(-\mu\Gamma^2/2\right) \leq 2\exp\left(-\alpha\Gamma^2/4\right), \tag{3.319}$$

where in getting to the last line we used that $xe^{-x} \leq e^{-x/2}$. Putting this together with the bound on the prefactor from (3.313), we get that $t_{\max}^{(\text{neg})} \leq \beta \exp\left(-\alpha\Gamma^2/4\right)$.

Bounding the potential integrals will be easier because the integrand is strictly positive. Each potential integral corresponds to four Gaussians with centers $\mathbf{x}_1$ through $\mathbf{x}_4$ and exponents ζ_1 through ζ_4:

$$\int d\mathbf{r}d\mathbf{s}\xi_{\zeta_1}(\mathbf{r}-\mathbf{x}_1)\xi_{\zeta_2}(\mathbf{s}-\mathbf{x}_2)\frac{1}{\|\mathbf{r}-\mathbf{s}\|}\xi_{\zeta_3}(\mathbf{s}-\mathbf{x}_3)\xi_{\zeta_4}(\mathbf{r}-\mathbf{x}_4) \tag{3.320}$$

$$= \prod_{i=1}^{4}\left(\frac{2\zeta_i}{\pi}\right)^{3/4}\int d\mathbf{r}d\mathbf{s}\exp\left[-\zeta_1\|\mathbf{r}-\mathbf{x}_1\|^2-\zeta_2\|\mathbf{s}-\mathbf{x}_2\|^2\right. \tag{3.321}$$

$$\left.-\zeta_3\|\mathbf{s}-\mathbf{x}_3\|^2-\zeta_4\|\mathbf{r}-\mathbf{x}_4\|^2\right]\frac{1}{\|\mathbf{r}-\mathbf{s}\|} \tag{3.322}$$

$$\leq \left(\frac{2\beta}{\pi}\right)^3\int d\mathbf{r}d\mathbf{s}\exp\left[-\alpha\left(\|\mathbf{r}-\mathbf{x}_1\|^2+\|\mathbf{s}-\mathbf{x}_2\|^2\right.\right. \tag{3.323}$$

$$\left.\left.+\|\mathbf{s}-\mathbf{x}_3\|^2+\|\mathbf{r}-\mathbf{x}_4\|^2\right)\right]\frac{1}{\|\mathbf{r}-\mathbf{s}\|} \tag{3.324}$$

$$= \left(\frac{2\beta}{\pi}\right)^3\int d\mathbf{r}d\mathbf{s}\exp\left[-\alpha\left(2\left\|\mathbf{r}-\frac{\mathbf{x}_1+\mathbf{x}_4}{2}\right\|^2+\frac{1}{2}\|\mathbf{x}_1-\mathbf{x}_4\|^2\right.\right. \tag{3.325}$$

$$\left.\left.+2\left\|\mathbf{s}-\frac{\mathbf{x}_2+\mathbf{x}_3}{2}\right\|^2+\frac{1}{2}\|\mathbf{x}_2-\mathbf{x}_3\|^2\right)\right]\frac{1}{\|\mathbf{r}-\mathbf{s}\|} \tag{3.326}$$

$$= \left(\frac{\beta}{\alpha}\right)^3\exp\left[-\frac{\alpha}{2}\left(\|\mathbf{x}_1-\mathbf{x}_4\|^2+\|\mathbf{x}_2-\mathbf{x}_3\|^2\right)\right]\left(\frac{2\alpha}{\pi}\right)^3 \tag{3.327}$$

$$\times\int d\mathbf{r}d\mathbf{s}\exp\left[-\alpha\left(2\|\mathbf{r}\|^2+2\left\|\mathbf{s}-\frac{\mathbf{x}_2+\mathbf{x}_3-\mathbf{x}_1-\mathbf{x}_4}{2}\right\|^2\right)\right]\frac{1}{\|\mathbf{r}-\mathbf{s}\|} \tag{3.328}$$

$$= \left(\frac{\beta}{\alpha}\right)^3\exp\left[-\frac{\alpha}{2}\left(\|\mathbf{x}_1-\mathbf{x}_4\|^2+\|\mathbf{x}_2-\mathbf{x}_3\|^2\right)\right]u_\alpha^{(\text{Coul})}\left(\frac{\mathbf{x}_2+\mathbf{x}_3-\mathbf{x}_1-\mathbf{x}_4}{2}\right) \tag{3.329}$$

$$\leq \beta^3\alpha^{-3}\sqrt{\frac{4\alpha}{\pi}} \leq 2\beta^3\alpha^{-5/2} \leq 2\beta^3, \tag{3.330}$$

and so $u_{\max} \leq 2\beta^3$. To bound $u_{\max}^{(\text{neg})}$, consider the above when at least one pair of the points $\mathbf{x}_1$ through $\mathbf{x}_4$ are at least Γ apart. If $\|\mathbf{x}_1-\mathbf{x}_4\| \geq \Gamma/2$, then the integral is at most

$$\beta^3\alpha^{-3}\exp\left[-\frac{\alpha}{2}\|\mathbf{x}_1-\mathbf{x}_4\|^2\right]u_\alpha^{(\text{Coul})}(0) \leq \beta^3\alpha^{-3}\exp\left(-\alpha\Gamma^2/8\right)2\sqrt{\alpha} \leq 2\beta^3\exp\left(-\alpha\Gamma^2/8\right) \tag{3.331}$$

and similarly for $\|\mathbf{x}_2-\mathbf{x}_3\| \leq \Gamma/2$. If neither of these are the case then at least one of $\mathbf{x}_1, \mathbf{x}_4$ must be at least Γ away from at least one of $\mathbf{x}_2, \mathbf{x}_3$. Without loss of generality, suppose

$\|\mathbf{x}_1 - \mathbf{x}_2\| \geq \Gamma$. That $\|\mathbf{x}_1 - \mathbf{x}_4\| \leq \Gamma/2$ implies

$$\left\| \frac{\mathbf{x}_1 + \mathbf{x}_4}{2} - \mathbf{x}_1 \right\| \leq \frac{\Gamma}{4} \tag{3.332}$$

and similarly for $\mathbf{x}_2$ and $\mathbf{x}_3$. Then

$$\left\| \frac{\mathbf{x}_2 + \mathbf{x}_3 - \mathbf{x}_1 - \mathbf{x}_4}{2} \right\| \tag{3.333}$$

$$= \left\| \left(\underbrace{\frac{\mathbf{x}_2 + \mathbf{x}_3}{2} - \mathbf{x}_2}_{\leq \Gamma/4} \right) - \left(\underbrace{\frac{\mathbf{x}_1 + \mathbf{x}_4}{2} - \mathbf{x}_1}_{\leq \Gamma/4} \right) + \left(\underbrace{\mathbf{x}_2 - \mathbf{x}_1}_{\geq \Gamma} \right) \right\| \tag{3.334}$$

$$\geq \Gamma - \frac{\Gamma}{4} - \frac{\Gamma}{4} = \frac{\Gamma}{2} \tag{3.335}$$

and the potential integral is at most

$$\beta^3 u_\alpha^{(\mathrm{Coul})}(\Gamma/2) = \beta^3 \sqrt{\frac{4\alpha}{\pi}} F_0(\alpha\Gamma^2/4) \leq \beta^3 \sqrt{\frac{4\alpha}{\pi}} \sqrt{\frac{\pi}{4}} \frac{1}{\sqrt{\alpha\Gamma^2/4}} = 2\beta^3 \frac{1}{\Gamma}. \tag{3.336}$$

Together with Eq. (3.330), this yields

$$u_{\max}^{(\mathrm{neg})} \leq \max\left\{2\beta^3 \exp(-\alpha\Gamma^2/8), 2\beta^3\Gamma^{-1}\right\} = 2\beta^3 \max\{\exp(-\alpha\Gamma^2/8), \Gamma^{-1}\} \tag{3.337}$$

For $\alpha\Gamma^2 \geq 64$ (a condition of the lemma),

$$\frac{1}{\exp(-\alpha\Gamma^2/8)} = \exp(\alpha\Gamma^2/8) = \sum_{k=0}^{\infty} \frac{1}{k!} \left(\frac{\alpha}{8}\Gamma^2\right)^k \tag{3.338}$$

$$\leq \frac{\alpha}{8}\Gamma^2 \leq \frac{\alpha^2}{8}\Gamma^2 = \sqrt{\frac{\alpha\Gamma^2}{64}} \cdot \Gamma \geq \Gamma \tag{3.339}$$

and so

$$u_{\max}^{(\mathrm{neg})} \leq 2\beta^3/\Gamma. \tag{3.340}$$

□

Part II

Application of quantum algorithms

Chapter 4

Generalized swap networks and their application

This chapter reproduces Ref. 125 with minor modifications. Sections 4.9 and 4.10 include a summary and excerpt of Refs. 91, 114, respectively. The practical use of many types of near-term quantum computers requires accounting for their limited connectivity. One way of overcoming limited connectivity is to insert swaps in the circuit so that logical operations can be performed on physically adjacent qubits, which we refer to as solving the "routing via matchings" problem. We address the routing problem for families of quantum circuits defined by a hypergraph wherein each hyperedge corresponds to a potential gate. Our main result is that any unordered set of k-qubit gates on distinct k-qubit subsets of n logical qubits can be ordered and parallelized in $O(n^{k-1})$ depth using a linear arrangement of n physical qubits; the construction is completely general and achieves optimal scaling in the case where gates acting on all $\binom{n}{k}$ sets of k qubits are desired. We highlight two classes of problems for which our method is particularly useful. First, it applies to sets of mutually commuting gates, as in the (diagonal) phase separators of Quantum Alternating Operator Ansatz (Quantum Approximate Optimization Algorithm) circuits. For example, a single level of a QAOA circuit for Maximum Cut can be implemented in linear depth, and a single level for 3-SAT in quadratic depth. Second, it applies to sets of gates that do not commute but for which compilation efficiency is the dominant criterion in their ordering. In particular, it can be adapted to Trotterized time-evolution of fermionic Hamiltonians under the Jordan-Wigner transformation, and also to non-standard mixers in QAOA. Using our method, a single Trotter step of the electronic structure Hamiltonian in an arbitrary basis of n orbitals can be done in $O(n^3)$ depth while a Trotter step of the unitary coupled cluster singles and doubles method can be implemented in $O(n^2\eta)$ depth, where η is the number of electrons.

4.1 Introduction

The state of experimental quantum computing is rapidly advancing towards "quantum supremacy" [26], i.e., the point at which quantum computers will be able to perform certain specialized tasks that are infeasible for even the largest classical supercomputers. Beyond this technical milestone, however, lies another: *useful* quantum supremacy, in which quantum computers can solve problems whose answers are of interest independently of how they were achieved. The combination of efficient quantum algorithms [74, 145] and scalable error correction [67] makes such progress likely in the long term, barring fundamental surprises. In the near term, we have so-called Noisy Intermediate-Scale Quantum (NISQ) devices [133], capable perhaps of outperforming classical devices on certain problems, but with extremely constrained resources. Many types of such devices (e.g., superconducting quantum processors) will have limited connectivity. For the most part, existing quantum algorithms assume an abstract device with arbitrary connectivity, i.e., the ability to do a two-qubit gate between any pair of qubits. In theory, this suffices given that circuits can be compiled to any concrete family of devices with polynomial overhead in qubits and gates [34]. In practice, polynomial overheads matter and can be the crucial difference between being feasible and infeasible on NISQ devices.

The overall goal of compilation within the quantum circuit model is to take a quantum algorithm and implement it (maybe approximately) on a concrete piece of quantum computing hardware. There are many approaches to this, but perhaps the most straightforward is to transform the desired quantum circuit into an executable one in two steps: 1) *decomposition* of the constituent gates into (maybe approximately) equivalent sub-circuits consisting of "native" gates, and 2) what we call *routing via matchings* of the circuit [42]. Our focus here is on the routing problem, in which the logical qubits are dynamically assigned to physical qubits in a way that allows the desired logical gates to be implemented while respecting the restricted connectivity of the actual hardware. In general, it may be necessary to use swap gates to change this assignment of logical qubits to physical qubits throughout the execution of the circuit.

In the past several years, there has been a blossoming of tools for addressing variants of this routing problem, which are variously called "quantum circuit placement", "qubit mapping", "qubit allocation", or "quantum circuit compilation" (though the latter term generally encompasses much more). Prior work, however, has taken one of two approaches. First, of theoretical interest, is to show how *any* quantum circuit can be converted "efficiently" (i.e., with polynomial overhead) into one in which gates act only locally in some hardware graph [34, 87, 113]. The second is an instance-specific approach, in which the problem is solved anew for each logical circuit [15, 28, 48, 86, 104, 105, 109, 140, 147, 151, 162, 165]. We propose and instantiate a new instance-independent approach, in which the routing is done for a family of instances, with little-to-no compilation necessary for each instance; the per-instance compilation time is therefore effectively amortized to nil. This approach, which finds solutions for families of instances, interpolates between the two approaches above and seeks to balance the time to solution and the quality of solution. The family-specific routing can be found either algorithmically or, as is done here, manually. Algorithms useful in the instance-specific approach, where quality of solution is prioritized over time to solution, may (but not necessarily) differ significantly from those useful in the instance-independent approach, wherein the prioritization is reversed. On the other hand, for many problem families, there is an instance with maximal structure on which instance-specific algorithms can be run, thus obtaining compilations that can be used for the whole family. In general, these instance-specific approaches work best on sparser cases, and on dense instances will return inferior compilations to the ones given here.

In many quantum algorithms for quantum chemistry it is the case that all circuits of a given size for a particular problem have the same structure with respect to a partial ordering of the operations, and the only instance-specific aspect is the parameters (e.g. rotation angles) of the gates. Furthermore, the implementation of these gates on hardware often has the same properties (e.g. fidelity and duration) regardless of the parameters. In such cases, the instance of the compilation problem is effectively independent of the instance of the application problem. Compare this with implementing QAOA on hardware in which gate durations are independent of their parameters, in which case the routing problem for a given problem instance is the same regardless of the variational parameters, but differs significantly for different problem instances. In cases in which gate durations vary, an upper bound on (or average over) the range of durations can be used to obtain instance-independent compilations. Thus, the distinction between instance-specific and instance-independent approaches is somewhat subjective and contextual, but we merely aim to emphasize that there is an under-explored regime in the trade-off between quality of solution and computation time in approaches to the quantum circuit routing problem.

An alternative approach for variational algorithms in general is to obviate the compilation problem by using an ansatz that is based on the connectivity of the target hardware [61, 95] and less so on the target application. By efficiently compiling application-specific circuits to constrained hardware, our methods combine the efficiency of this approach with respect to physical resources with the advantages of an application-specific ansatz (e.g., fewer variational parameters).

A method was recently proposed for implementing a Trotter step of a fermionic Hamiltonian containing $\binom{n}{2}$ terms, where n is the number of orbitals, using a circuit of depth n with only linear connectivity [94, 100]. Using fermionic swap gates [47], Kivlichan et al. were able to change the mapping between fermionic modes and physical qubits while preserving anti-symmetry [100]. By constructing a network of these gates such that, at some point in the circuit, each pair of orbitals is assigned to some pair of neighboring qubits, they were able to guarantee that they could implement each of the terms in the Trotter decomposition of the Hamiltonian by acting only locally on said pair of qubits, and that they could implement $n/2$ such terms in parallel. In this work, we generalize their approach and describe a way to construct networks of fermionic swap gates acting on n qubits such that each k-tuple of fermionic modes is mapped to adjacent physical qubits at some point during the circuit. The circuits that we construct have an asymptotically optimal depth of $O(n^{k-1})$ while only assuming linear connectivity.

For fermionic Hamiltonians, Motta et al. take a different approach that exploits the fact that many Hamiltonians of practical interest are low-rank [121]. For unitary coupled cluster and full-rank generic chemical Hamiltonians, their methods achieve the same scaling as ours, as summarized in Table 4.1. Our methods provide an alternative Trotter order, whose relative value will need to be studied empirically. For a Trotter step of the Hamiltonian for real molecular systems, empirical data indicate that their low-rank methods can achieve $O(n^2)$ depth.

The question of how to optimally implement a collection of k-qubit operators is not confined to the simulation of fermionic quantum systems. Another promising use is in the application of the Quantum Alternating Operator Ansatz (Quantum Approximate Optimization Algorithm, or QAOA) to Constraint Satisfaction Problems (CSPs) over Boolean domains. This approach was taken for the Maximum Cut problem using existing linear swap networks [50]; our methods can address k-CSP for any k in $O(n^{k-1})$ depth.

Our main contributions are:

- Formalizing a variant of the quantum circuit routing problem in a way that abstracts away details of particular devices and focuses on their geometry, which is shared by a wide class of devices;
- Making explicit and general the equivalence between swap networks that change the mapping of logical qubits to physical qubits and those that change the mapping between fermionic modes and physical qubits;
- Explicit constructions for several important classes of problems, as summarized in Table 4.1, using modular primitives that can be applied to new problems; and
- Providing tools for lower bounding the depth of solutions to the routing problem, in particular by connecting it with prior work on acquaintance time and graph minors.

This chapter is organized as follows. In Section 4.2, we more formally describe the quantum circuit routing problem and our approach thereto. In Section 4.3, we introduce generalized swap networks that will be used in the constructions of later sections. In Section 4.4, we introduce some specific quantum simulation tasks related to fermionic Hamiltonians, as well as the Quantum Alternating Operator Ansatz (QAOA), which yield families of circuits that can be routed using our methods. In Section 4.5, we present our main result, showing how to achieve optimal scaling when routing (with an arbitrary ordering) circuits consisting of a k-qubit gate for each possible set of k qubits. In Section 4.6 we describe families of instances arising from the Unitary Coupled Cluster method and how to efficiently route them. A reader familiar with either QAOA or quantum simulation of fermionic Hamiltonians and interested in quickly learning some useful techniques may do so in sections 4.3, 4.5, and 4.6. In Section 4.7, we discuss the instance-independent approach in the context of quantum annealing. In Section 4.8, we show how to lower bound the depth of a solution to the circuit routing problem. In Section 4.9, we introduce a Non-orthogonal Variational Quantum

Instance family:	k-CSP	UCCGSD	UCCSD	UpCCGSD
Depth:	$\Theta(n^{k-1})$	$\Theta(n^3)$	$\Theta(\eta n^2)$	$\Theta(n)$

Table 4.1: Main results. k-CSP indicates depth for a single round of QAOA. Remaining columns indicate depth for a single Trotter step of the coupled cluster operator or of a similarly structured variational ansatz. All are optimal up to constant prefactors for arbitrary connectivity. See Section 4.5 for details regarding k-CSP and Section 4.6 regarding Unitary Coupled Cluster.

Eigensolver; we used swap networks to compile the circuits in our numerical experiments. In Section 4.10, we construct swap networks for block-diagonal Hamiltonians.

4.2 Model

We consider hardware consisting of a line of n qubits and suppose that we are able to implement any k-qubit gate in time τ_k on any set of k qubits that are adjacent. This is an abstraction of the more physical model in which only 1- and 2-qubit gates can be directly implemented; τ_k for $k > 2$ is thus some linear combination of τ_1 and τ_2 that indicates an upper bound on the cost of compiling any k-qubit gate. When considering a specific piece of hardware, this model is relatively coarse; different gates on different sets of physical qubits may require vastly different times to implement. However, this level of abstraction allows for significant generality without too great a loss of precision. Accordingly, for a specific piece of hardware, our constructions should be considered as a starting point, with low-level optimizations likely to improve the constant factors significantly. For example, the line of qubits on which the swap networks are defined can be embedded in a "castellated" manner in a $2 \times (n/2)$ lattice, as shown in Figure 4.1. The availability of the additional qubit adjacency can enable more efficient decomposition of higher-locality gates.

The problem we would like to solve is as follows: given a set of k-qubit gates G on n qubits, implement them in some order on the hardware described above. In particular, we focus on the swap-network paradigm. That is, we start with an initial assignment of logical qubits to physical qubits and insert a sequence of 2-qubit swap gates to move the logical qubits around so that for every gate in G the logical qubits on which it acts are physically adjacent at some point in the process. As discussed in Sections 4.4.1 and 4.6, the routing problem thus defined is equivalent to the problem of using fermionic swap gates to change the ordering of a Jordan-Wigner string to enable the implementation of gates locally. Without loss of generality, we assume that there is at most one gate in G acting on any set of qubits, and that for any gate $g \in G$ acting on a set of qubits S there is no other gate $g' \in G$ acting on a subset of qubits $S' \subset S$. This is a convenient abstraction, rather than a restriction. An instance of the routing problem is thus specified as a hypergraph, with vertices corresponding to logical qubits and hyperedges corresponding to logical gates. We focus on k-complete

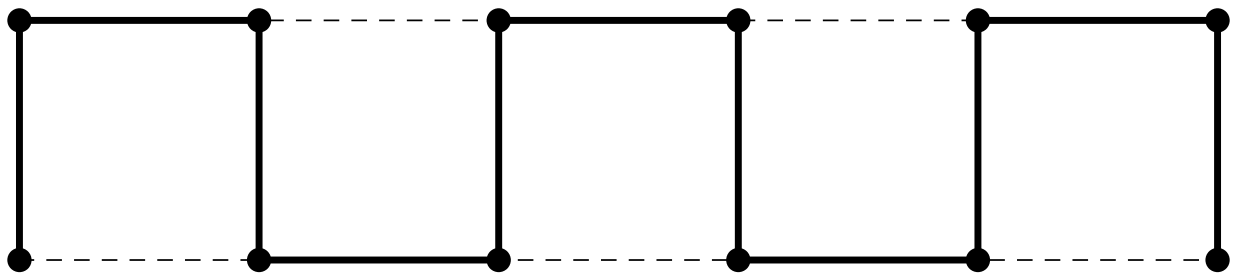

Figure 4.1: "Castellated" mapping of the Jordan-Wigner string into a $2 \times (n/2)$ square lattice. While all swapping is done along the Jordan-Wigner string, mapping the string to the lattice in this way allows for potentially more efficient decomposition of the 4-qubit gates by making available a fourth adjacency.

hypergraphs, ones in which for every subset of k vertices, there is an edge connecting them; $|G| = \binom{n}{k}$. Results for complete hypergraphs give worst case bounds for the general problem. A more general variant is the more typically considered problem in which one wants to enforce a temporal partial ordering on the logical gates.

In general, near-term hardware will have greater connectivity than a line; nevertheless, it will likely contain a line as a subset, so that our constructions give a baseline. Even with greater connectivity, our scaling is optimal when the number of gates is $\Omega(n^k)$. Let $m = |G|$ be the number of gates, ν the number of physical qubits, and n the number of logical qubits. At most ν gates can be implemented at a time, so the circuit depth must be at least m/ν. For $m = \Omega(n^k)$ and $\nu = O(n)$, this implies a minimal depth of $\Omega(n^{k-1})$, which our construction provides. Because our focus is on resource-constrained near-term hardware, we shall assume that the number of physical qubits is equal to the number of logical qubits.

4.3 Swap networks

Henceforth, by "swap gate", we shall mean either the standard swap gate (when considering a mapping of logical qubits to physical qubits) or the fermionic swap gate (when considering a mapping of fermionic modes to physical qubits); for circuit routing, everything is exactly the same in both cases except for the "interpretation". A *swap network* is a circuit consisting entirely of swap gates. We define a *2-complete linear swap network*, a notion we shall generalize shortly, to be a swap network in which all pairs of logical qubits are linearly adjacent at some point in the circuit and in which all swap gates act on linearly adjacent physical qubits. Such networks ensure that, in the linear architecture described in Sec 4.2, there is an opportunity to add, for each pair of logical qubits, a 2-qubit gate acting on those logical qubits (or fermionic modes as the case may be) at some point in the circuit. We call such opportunities *acquaintance opportunities.* They are not part of the swap network, but we shall often draw them as empty boxes in circuit diagrams to illustrate acquaintance

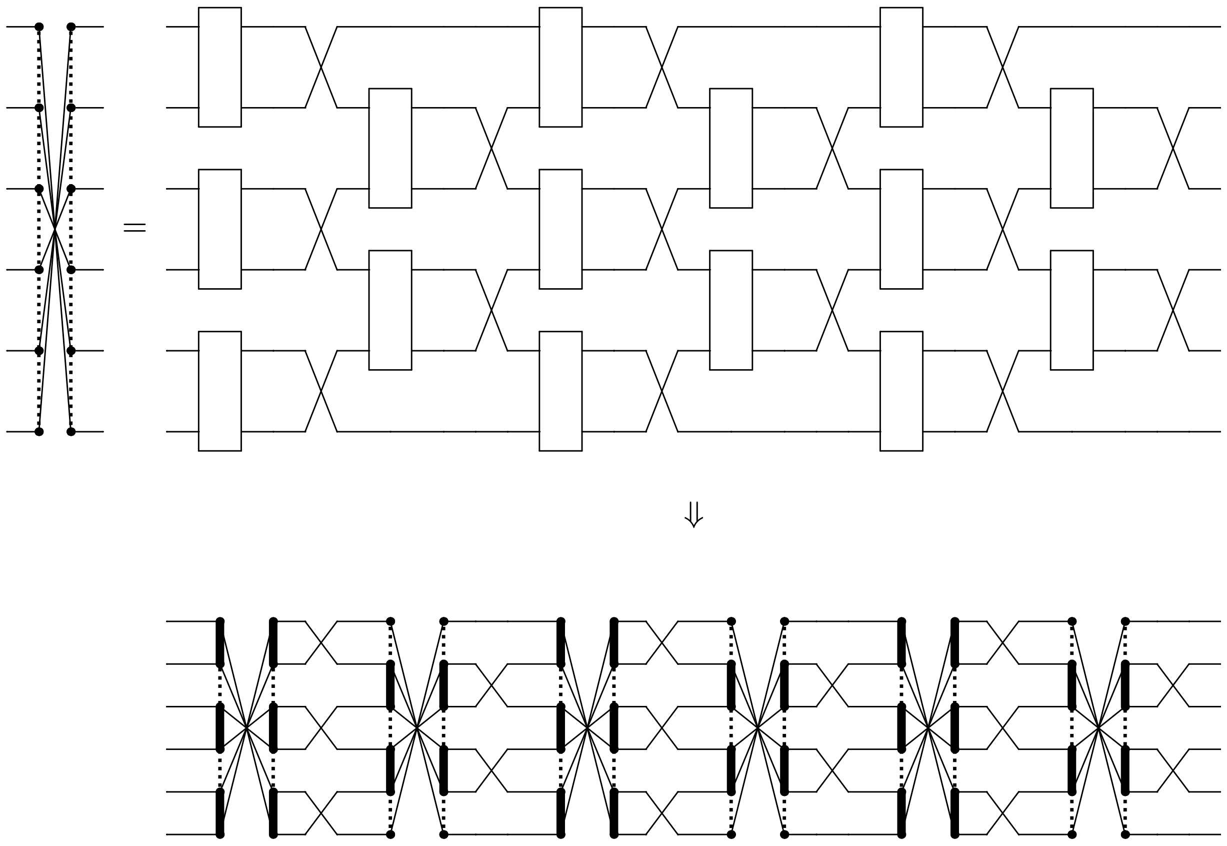

Figure 4.2: Top: Notation (left) and decomposition (right) for the canonical 2-complete linear (2-CCL) swap network for acquainting all pairs of qubits, annotated with empty boxes showing acquaintance opportunities. The circuit has depth n, and contains $n(n-1)/2$ swap gates. Bottom: The canonical 3-complete linear (3-CCL) swap network that acquaints all triples of qubits. It is formed by replacing each layer i of acquaintance opportunities in the 2-CCL swap network with a $\mathcal{P}_i$-swap network; the qubits to be acquainted in the former are the parts in the partition $\mathcal{P}_i$. The 3-local acquaintance opportunities are not shown. Part of this swap network is shown in more detail in the top of Figure 4.5, with acquaintance opportunities indicated. (Each layer of 1-swaps will be cancelled out by the final layer in the expansion of the preceding 2-swap network; we include them here to make the recursive step clear.)

properties of swap networks, as in Figure 4.2. We shall say that a set of logical qubits that has at least one such acquaintance opportunity is "acquainted" by the network, or that the swap network "acquaints" those qubits.

Before generalizing this notion, we review the construction of Kivlichan et al. [100] for implementing a 2-local gate on every pair of logical qubits in depth n, using $\binom{n}{2}$ swap gates. The swap network underlying this construction is what we shall call the *canonical* 2-complete linear swap network. Let the physical qubits be labeled 1 through n, and partition the pairs of adjacent qubits into two sets based on the parity of their larger index: even pairs $\{\{1,2\},\{3,4\},\ldots\}$ and odd pairs $\{\{2,3\},\{4,5\},\ldots,\}$. Note that the pairs in each partition are mutually disjoint. We define the *canonical 2-complete linear* (2-CCL) swap network as n alternating layers of swaps on the even pairs and odd pairs, as illustrated in the top half of Figure 4.2. The overall effect of the 2-CCL swap network is to reverse the ordering of the logical qubits. In doing so, it directly swaps every pair of logical qubits. This construction has the attractive property that each acquaintance opportunity precedes a swap gate on the same two qubits, so any added gate that acts on a pair of logical qubits can be combined with the swap of those two qubits, with the result that in depth n we can execute a 2-qubit gate between every pair of logical qubits.

One direction for generalization is to $(\mathcal{S},\mathcal{A})$-swap networks, where $\mathcal{S}$ is a subset of all pairs of qubits and $\mathcal{A}$ is an architecture, such as a 2D grid. The set $\mathcal{S}$ captures the pairs of qubits to which we want to apply 2-qubit gates at a given stage in a circuit. We shall not discuss this generalization further, other than to note that our results can be used to provide bounds for $(\mathcal{S},\mathcal{A})$-swap networks. Because in the present work we shall present only swap networks acting on a line, we shall often leave that aspect implicit in the terminology and refer simply, e.g., to a "2-complete swap network".

Instead, we are interested in generalizing to *k-complete swap networks*, networks in which the elements of every set of k logical qubits are adjacent at some point, so that a k-qubit gate (or set of 1- and 2-qubit gates making up the k-qubit gate) could be applied thereto. To support the construction of k-complete swap networks in Sec.4.5, here we introduce a generalization of a 2-complete swap network that swaps elements of a partition of qubits, rather than individual logical qubits: a *complete $\mathcal{P}$-swap network*, where $\mathcal{P}$ is an ordered partition of the physical qubits such that each part contains only contiguous qubits, contains only swap operators that swap parts of the partition. In this way, a complete $\mathcal{P}$-swap network has the property that every part in the partition is adjacent to every other part in the partition at some point in the network.

In constructing $\mathcal{P}$-swap networks, it will be useful to swap pairs of sets of qubits using what we call a *(k_1,k_2)-swap gate*, or, more generally, a *generalized swap gate.* The (k_1,k_2)-swap gate swaps a set of k_1 logical qubits with a set of k_2 logical qubits, while preserving the ordering within each set, i.e., it permutes a sequence of logical qubits from $(i_1,\ldots,i_{k_1},i_{k_1+1},\ldots,i_{k_1+k_2})$ to $(i_{k_1+1},\ldots,i_{k_1+k_2},i_1,\ldots,i_{k_1})$. Several examples of these generalized swap gates and their decompositions are shown in Figure 4.3. In general, a (k_1,k_2)-swap gate can be decomposed using $k_1\cdot k_2$ standard swap gates in depth k_1+k_2-1. We call a swap network a *k-swap network* whenever it contains only (k_1,k_2)-swap gates for $k_1,k_2\leq k$.

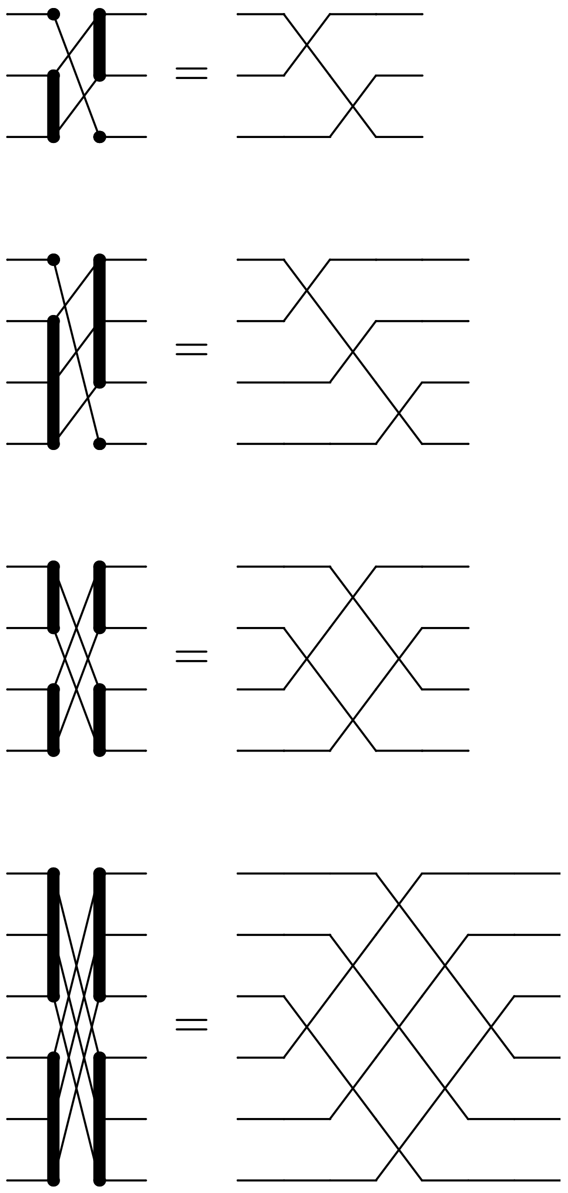

Figure 4.3: **Generalized swap gates.** From top to bottom, notation and decompositions for $(1,2)$-, $(1,3)$-, $(2,2)$-, and $(3,3)$-swap gates. A (k_1,k_2)-swap gate can be implemented in depth k_1+k_2-1 using k_1k_2 standard swap gates.

The *canonical $\mathcal{P}$-swap network* has the same structure as the 2-CCL swap network, except that instead of pairs of single qubits being swapped at a time, pairs of sets of qubits (i.e., the parts of the partition) are swapped. In the canonical $\mathcal{P}$-swap network, each (k_1, k_2)-swap gate is preceded by a $(k_1 + k_2)$-local acquaintance opportunity. To make the overall effect of a complete $\mathcal{P}$-swap network be a complete reversal of the qubit mapping, we append to the end a 1-swap network within each part. This is unnecessary when considering a single swap network, but may be helpful when using the swap network as a primitive in a larger construction. Note that this is is primarily for explanatory purposes, and in an actual implementation would likely be optimized away. In the recursive strategy for k-local hypergraphs (discussed in Sec. 4.5), each generalized swap gate is preceded by some number of acquaintance opportunities and swap gates that ensure that each set of k_1 or k_2 qubits is acquainted with each one of the other set.

The 2-CCL swap network has the exact same structure as the optimal sorting network on a line [13]. A sorting network is a fixed circuit consisting of "comparators". Given an initial assignment of objects to the wires, each comparator compares the objects and swaps them if they are out of order. This means that a subset of the 2-CCL swap network can be used to effect an arbitrary permutation of logical qubits in at most linear depth.

The swap networks above acquaint all pairs of sets of qubits. Another useful primitive is what we call a "bipartite swap network"; again, this should be more precisely called a "bipartite linear swap network" to emphasize that it acts on a line, but we leave this implicit for concision. Given a bipartition of sets of qubits, it acquaints all the unions of pairs of sets of qubits which can be formed by taking one set from the first part and the other set from the second part. While the depth of a bipartite swap network is similar to that of a complete swap network, the gate count is approximately halved. Figure 4.4 shows an example bipartite swap network for the sets of qubits $((1, 2), (3, 4), \ldots, (11, 12))$ with the first three in one part and the latter three in the second part.

Swap networks can be useful for measurement as well. In many cases, the gates to be executed correspond one-to-one with the terms of a Hamiltonian to be measured. Any swap network used to implement those gates thus yields a partition of the terms of the Hamiltonian into parts containing only gates acting on disjoint sets of qubits. This partition can then be used to parallelize the measurements. After an application of the swap network, the swap layers following the logical layer to be measured can be executed in reverse to return the mapping to one in which the terms of the Hamiltonian are mapped to adjacent sets of qubits, with appropriate optimizations made to account for the fact that many swap gates will likely cancel out once the logical gates are removed. Alternatively, a simple sorting network can be used to achieve the same end. For fermionic Hamiltonians, this approach can significantly reduce the number of measurements needed by reducing the locality of all measurement terms, in addition to the savings yielded by parallelization.

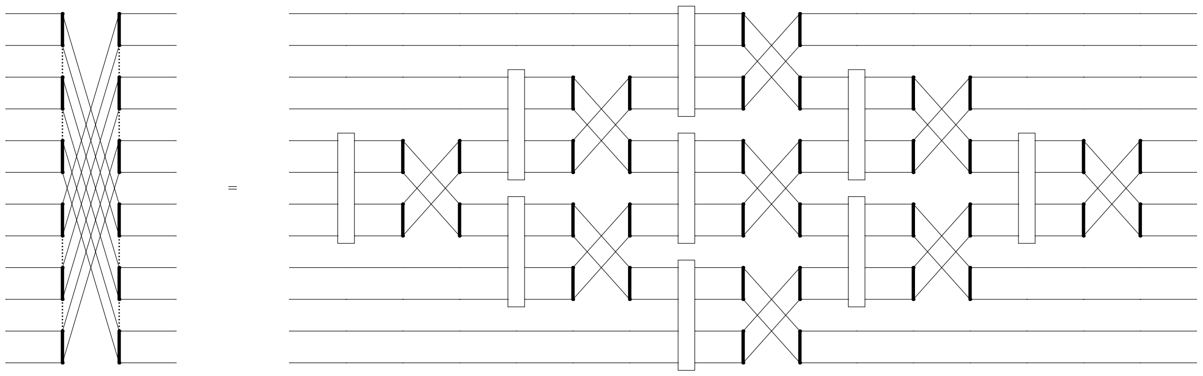

Figure 4.4: Notation (left) and decomposition (right) for a bipartite swap network corresponding to the bipartition $(((1,2),(3,4),(5,6)),((7,8),(9,10),(11,12)))$. For each pair of qubits in the first half and each pair of qubits in the second half, their union is acquainted; the overall effect is the same as that of a generalized swap gate corresponding to the two halves. Note the similarity of the notation to that for a complete swap network, except that the dotted lines connect only each part of the bipartition.

Application	QAOA	Quantum chemistry
Iteration	$\prod_I \exp\left(i\gamma c_I \prod_{i\in I} Z_i\right)$	$\prod_{p,q,r,s} \exp\left(-itH_{p,q,r,s}\right)$
Assignment	logical qubits	fermionic modes
Changed by	SWAP	FSWAP

Table 4.2: The two problem families we consider. The table lists the iterated operator we compile, and the logical unit and gates used in the compilation.

4.4 Problem families

In this section, we introduce two families of quantum circuits that come from quite different application domains but whose compilation can be addressed using essentially the same tools; the analogy is summarized in Table 4.2. Both cases involve repeated application of a circuit of a particular form such that for each iteration the compilation instance is the same. We provide constructions for a single iteration; these can be repeated sequentially for the full circuit. Solving the compilation instance for the full circuit all at once may provide a better solution, but likely at the cost of it being much harder to find.

4.4.1 Fermionic Hamiltonians

The general form of the electronic structure Hamiltonian in second quantization is

$$H = \sum_{p,q} c_{p,q} a_p^\dagger a_q + \sum_{p,q,r,s} c_{p,q,rs} a_p^\dagger a_q^\dagger a_r a_s, \tag{4.1}$$

where p, q, r, s label single-electron orbitals, $c_{p,q}$ and $c_{p,q,r,s}$ are real coefficients, and $a_p^\dagger$ is the creation operator for the pth orbital. A common subroutine of quantum simulation algorithms is the Trotterization of time evolution under such a fermionic Hamiltonian [10]:

$$e^{-iHt} \approx \prod_{l=1}^{t/\delta t} \left(\prod_{p,q,r,s} e^{-iH_{p,q,r,s}\delta t} \right), \tag{4.2}$$

where $H_{p,q,r,s}$ is the part of the Hamiltonian that acts exclusively on modes p,q,r,s. (For simplicity we absorb the terms acting on two fermionic modes into the terms acting on four.) One approach to mapping the fermionic operators $e^{-iH_{p,q,r,s}}$ into operators acting on the qubit Hilbert space is to employ the Jordan-Wigner transformation [126],

$$a_p = -\prod_{i=1}^{p-1} \sigma_i^{(z)} \cdot \sigma_p^{(-)}. \tag{4.3}$$

After performing the Jordan-Wigner transformation on Equation 4.2, many of the resulting operators will act non-trivially on $\Theta(n)$ qubits, resulting in a naive gate depth of $\Theta(n^5)$ for the implementation of Equation 4.2, assuming there are $\Theta(n^4)$ terms in the Hamiltonian. As we shall see, by reordering the fermionic modes (thereby changing the Jordan-Wigner ordering), this overhead from the non-locality of the Jordan-Wigner transformation is addressed automatically in our scheme for parallelization. For this reason, our constructions provide significant advantage even when connectivity is not a constraint, including in the error-corrected regime. As a result, at least with respect to scaling, we avoid the need for more sophisticated alternatives to the Jordan-Wigner transformation, such as those developed by Bravyi and Kitaev [33] and others [32].

A related approach, employed by a variety of works proposing the study of quantum chemistry using a near-term device, is the use of a quantum circuit to prepare and measure the unitary coupled cluster ansatz [103, 115, 130, 139]. Under the typical choice to include only single and double excitations in the cluster operator, this wave function is given by

$$|\psi\rangle = e^{T-T^\dagger}|\phi_0\rangle, \tag{4.4}$$

where the cluster operator T has a form similar to H in Equation 4.1. Usually, it contains only excitations from the η "occupied" orbitals which contain an electron in the reference state $|\phi_0\rangle$ to the $n-\eta$ "virtual" orbitals, and the coefficients are determined variationally. We refer to this case as UCCSD, and the case where all 2-electron excitations are included as UCCGSD.

The exact exponential of Equation 4.4 is typically approximated by a Trotter expansion and (assuming $n \gg \eta$), the $\Theta(n)$ overhead from the non-locality of the Jordan-Wigner strings discussed above would lead to a circuit depth of $\Theta(n^3\eta^2)$ for a single Trotter step.

We show how depths of $O(n^3)$ and $O(n^2\eta)$ can be achieved for a Trotter step of the time evolution under a fermionic Hamiltonian (or the similarly structured UCCGSD) and the UCCSD ansatz, respectively. These scalings match the asymptotic results of Ref. [81] while also respecting the spatial locality of the available gates and requiring no additional ancilla qubits.

4.4.2 QAOA

As originally proposed [62, 64], QAOA is a method for minimizing the expectation value of a diagonal Hamiltonian

$$H_f = \sum_{I \subset [n]} c_I \prod_{i \in I} Z_i \tag{4.5}$$

corresponding to a classical function $f : \{\pm 1\}^n \to \mathbb{R}$ whose multilinear form is

$$f(\mathbf{s}) = \sum_{I \subset [n]} c_I \prod_{i \in I} s_i. \tag{4.6}$$

The minimization is done variationally over states of the form

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = e^{i\beta_p M} e^{i\gamma_p H_f} \cdots e^{i\beta_1 M} e^{i\gamma_1 H_f} |+\rangle^{\otimes n}, \tag{4.7}$$

which consists of p alternating applications of the "phase separator" $e^{i\gamma H_f}$ and the "mixer" $M = \sum_{i=1}^n X_i$. The phase separator can be written as the product of gates corresponding to terms in the Hamiltonian,

$$e^{i\gamma H_f} = \prod_I e^{i\gamma c_I \prod_{i \in I} Z_i}. \tag{4.8}$$

Note that the gates are diagonal and so their order does not matter. The locality of the gates corresponds directly to the locality of the terms in the Hamiltonian, $\max |I|$. QAOA applied to k-CSP, in which each term acts on at most k variables, thus requires k-qubit gates.

Hadfield et al. [77] generalized QAOA to the Quantum Alternating Operator Ansatz, employing a wider variety of mixers, many of which involve k-qubit gates. While these gates, in general, do not commute, it is an open question how the order of the gates affects the efficacy of the mixing. In NISQ devices with limited depth, the depth in which different mixers can be implemented plays a key role in their usefulness. The techniques here can be applied to these alternative mixers, with different orderings giving different mixers within the same family, and the resulting compilation a key step in determining the most effective mixing strategy.

4.5 Complete hypergraphs

4.5.1 Cubic interactions

Now, suppose we want to implement a 3-qubit gate between every triple of logical qubits. We call a swap network that achieves this goal a 3-*complete linear* (3-CCL) swap network. We can do so in the following way. First, we start with the 2-CCL swap network, as shown in the top half of Figure 4.2. At each layer i where acquaintance opportunities appear, consider the partition $\mathcal{P}_i$ whose parts are the pairs of qubits appearing in the acquaintance opportunities together with singleton parts for any unpaired qubits at the boundary. To obtain a 3-complete linear swap network, we add 2-swap networks corresponding to the partition $\mathcal{P}_i$, as shown in the bottom half of Figure 4.2. The 3-way acquaintance opportunities, where 3-local gates (or compilations of them to 1-and 2-local gates) can be added, are interspersed between the generalized swaps making up the $\mathcal{P}_i$-swap network, as shown in the top half of Figure 4.5. We make use of the property that for any two pairs of logical qubits involved in a 2-swap, each triple consisting of one of the pairs and one qubit from the other pair is mapped to three contiguous physical qubits either before or after the swap. This ensures that overall every triple of logical qubits is acquainted because any triple T is the union of a pair and a third qubit. The 2-CCL network ensures that the pair is adjacent at some point, and thus a part S of some partition $\mathcal{P}_i$. The third qubit is necessarily in some other part S' of the same partition, so that at some point in the $\mathcal{P}_i$-swap network there is a 2-swap network involving S and S', ensuring that the triple T is acquainted. (Actually, it is acquainted thrice, because there are three pairs S for which the preceding logic applies.) There are exactly n 2-swap networks inserted, and each 2-swap gate can be implemented in depth 3 using standard swap gates, for a total depth of approximately $(3/2)n^2(\tau_2 + \tau_3) = \Theta(n^2)$.

4.5.2 General k-qubit gates

The above ideas generalize to arbitrary k. The construction is recursive. First, construct the network to implement all $(k-1)$-qubit gates. Then replace every layer i of acquaintance opportunities with the corresponding $\mathcal{P}_i$-complete swap network, inserting 1-swaps and acquaintance opportunities between the layers of $(k-1)$-swaps in order to acquaint each set of $k-1$ qubits with each qubit in the other set of $k-1$ qubits with which it will be swapped. Specifically, when inserting a $(k-1)$-swap involving two sets of $(k-1)$ qubits each, we want to ensure that each set of k qubits consisting of one of the sets and one qubit from the other set is mapped to k contiguous physical qubits either before or after the swap. For $k = 2$, this is the case without additional swaps. For larger k, this can be achieved by adding swaps that bring half each of set to the "interface" between them before the swap (the half closest to the interface), and the other half to the interface afterwards (when it will then be the closer half). This ensures that overall every set of k logical qubits is acquainted because any such set T is the union of a set S of $k-1$ qubits and a kth qubit t. Suppose we start with a swap network that acquaints every set of $k-1$ qubits, and in particular S, so that S is a part of

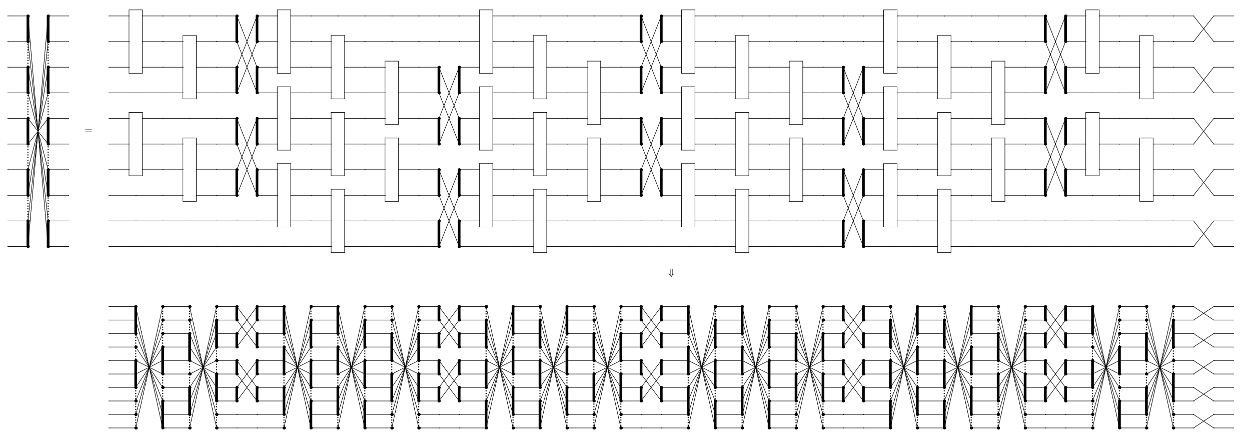

Figure 4.5: Top: Notation and decomposition for complete 2-swap network for acquainting each pair of qubits in the initial partition with every other third qubit. Note how the 3-qubit acquaintance opportunities are almost perfectly parallelized; this helps significantly when recursing further. Bottom: Swap network for acquainting every set of 4 qubits such that 2 of the 4 qubits were paired in the partition of the originating 2-swap network, formed by replacing each layer of acquaintance opportunities in the complete 2-swap network with a complete 3-swap network, in the same manner as in Figure 4.2. In the swap network to acquaint *every* set of 4 qubits, this replacement (i.e., from top to bottom of this figure) is done for every complete 2-swap network in the circuit for acquainting every set of 3 qubits, i.e., for every other layer in the bottom of Figure 4.2.

some partition $\mathcal{P}_i$ (corresponding to acquaintance layer i in the starting swap network) in the recursive step. The kth qubit t is necessarily in some other part S' of the same partition $\mathcal{P}_i$, so that at some point in the $\mathcal{P}_i$-swap network there is a $(k-1)$-swap involving S and S', ensuring that the set T is acquainted. (Actually, it is acquainted at least k times, because there are k sets $S \subset T$ for which the preceding logic applies.)

Each $(k-1)$-swap network has depth at most n in terms of $(k-1)$-swap gates. A k-swap gate has depth at most $2k-1$ in terms of standard swap gates, and the additional swaps for bringing inner qubits to the interface add depth $k-2$ at each swap. Therefore, if we have a depth $O(n^{k-2})$ construction for all $(k-1)$-qubit gates, we can use that to get an $O(n^{k-1})$ depth construction for all k-qubit gates. The base case is the linear-depth 2-CCL swap network for 2-qubit gates. Figures 4.2 and 4.5 show the steps for $k = 4$. Lower-locality gates can be included in one of two ways, or a combination thereof. First, they can be incorporated directly into the highest-locality gates. Alternatively, the lower-locality acquaintance opportunities can be kept when recursing.

Using this recursive method yields a significant amount of redundancy with respect to the number of times that each set of k qubits can be acquainted. For applications in which the gates do not commute, this can be exploited in two ways. First, distributing the gates over all possible acquaintance opportunities may lead to smaller Trotter errors. Second, for each gate a possible acquaintance opportunity may be chosen randomly. In other words, the swap network can be considered as a family of swap networks, each corresponding to a particular Trotter order; prior work shows that such random Trotter orderings may be helpful [41].

4.5.3 Alternative for 3-local

Here we present an alternative construction for sets of 3-local gates. Its depth is similar to that of the other given, but it doesn't obviously generalize. We include it for two reasons: it demonstrates a potentially useful property of complete linear swap networks, and it may be better when applied to specific hardware devices.

Note that in the 2-complete swap network, every pair of logical qubits that is initially at distance 2 from each other remains so, except near the ends of the line. Furthermore, every other logical qubit passes through them at some point. For our purposes, this means that in the course of the 2-complete swap network we can execute any 3-local gate such that some pair of the three logical qubits on which it acts is at distance 2 at the start of the network.

Consider a sequence of mappings labeled by $\Delta = 1, \ldots, n/2$. In the mapping labeled by Δ, the logical qubits $(1, 1+\Delta, 1+2\Delta, \ldots, 2, 2+\Delta, 2+2\Delta, \ldots, \Delta, 2\Delta, 3\Delta, \ldots,)$ are mapped to physical qubits $(1, n, 3, n-1, 5, \ldots, \lfloor n/2 \rfloor + 1)$, respectively. Any triple of logical qubits contains at least one pair that are mapped to physical qubits at distance 2 in at least one of the $n/2$ mappings. The construction is thus: alternate between 1) 2-complete swap networks with initial assignments given by the mappings, and 2) sorting networks to get to the next mapping. The 2-complete swap networks have depth n and the sorting networks depth at most n, so overall the total depth is at most $2n \cdot (n/2) = n^2$.

4.6 Unitary Coupled Cluster

In this section, we describe how swap networks can be used to implement Trotterized versions of three different types of unitary coupled cluster ansatz with a depth scaling that is optimal up to constant prefactors. We present the details for the standard unitary coupled cluster method with single and double excitations from occupied to virtual orbitals (UCCSD) [115, 130, 139], a unitary coupled cluster that includes additional, generalized, excitations (UCCGSD) [122, 157], and a recently introduced ansatz that is a sparsified version of UCCGSD (k-UpCCGSD) [103].

The standard unitary coupled cluster singles and doubles ansatz is given by

$$|\psi\rangle = e^{T-T^\dagger}|\phi_0\rangle, \tag{4.9}$$

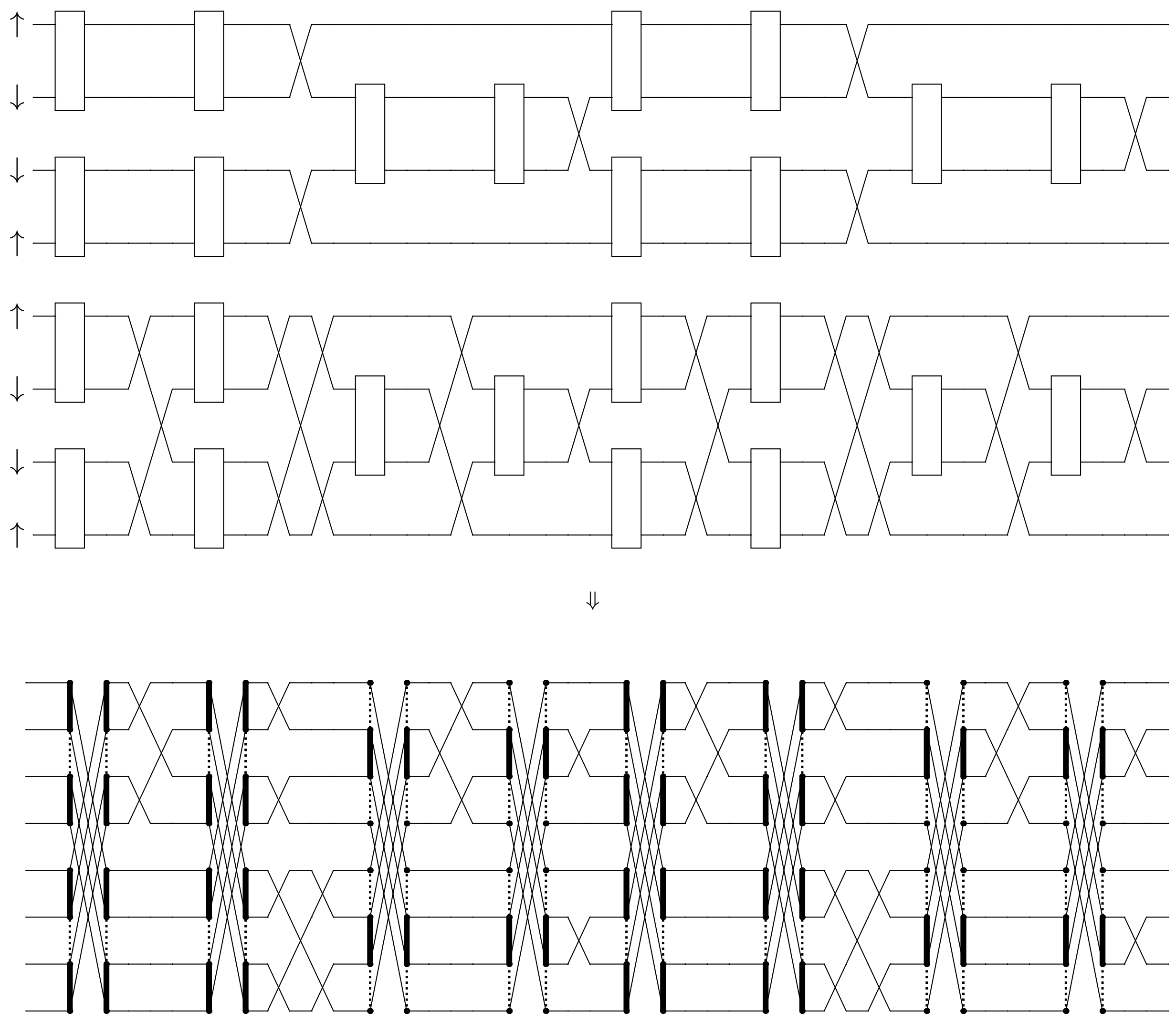

Figure 4.6: Construction of the swap network for a UCCSD operator (Equation 4.10) with two occupied and two virtual spatial orbitals. On top is an intermediate form of the construction useful for reasoning through the logic of its structure; on bottom is the actual network. On the first four qubits of the top half is a complete swap network with the acquaintance opportunity layers repeated twice. On the second four qubits are four concatenated complete swap networks, one for each acquaintance layer (before repetition) of the complete swap network on the first four qubits. The spins of the orbitals in the initial mapping to qubits is indicated; with this initial mapping, the parity of the spins of the orbitals to be acquainted in each layer of the complete swap networks is the same (either all ↑↑ or ↓↓, or all ↑↓). For every pair of occupied spin orbitals with some parity and every pair of virtual spin orbitals with the same parity, there is some layer of the combined swap network in which both pairs are simultaneously (but separately) acquainted. The construction is completed by replacing each acquaintance layer with a bipartite swap network over the occupied and virtual orbitals, which then acquaints the union of every such pair of pairs of spin orbitals. (An example bipartite swap network is shown in Figure 4.4.)

where $|\phi_0\rangle$ is the Hartree-Fock state, $T = T_1 + T_2$,

$$\begin{aligned} T_1 &= \sum_{\substack{i \in \text{occ} \\ a \in \text{vir}}} t_i^a a_a^\dagger a_i, \\ T_2 &= \sum_{\substack{i,j \in \text{occ} \\ a,b \in \text{vir}}} t_{i,j}^{a,b} a_a^\dagger a_b^\dagger a_j a_i. \end{aligned} \tag{4.10}$$

The i and j indices range over the η "occupied" orbitals (those which are occupied in the Hartree-Fock state $|\phi_0\rangle$) and the a and b indices over the $n - \eta$ "virtual" orbitals (those which are unoccupied in $|\phi_0\rangle$). A Trotter step of the corresponding unitary has $\binom{\eta}{2}\binom{n-\eta}{2}$ 4-local gates.

These can be implemented in $O(\eta n^2)$ depth, as shown in Figure 4.6. First, we assign the occupied orbitals to the first η physical qubits $(1, \dots, \eta)$ and the virtual orbitals to the last $n - \eta$ physical qubits $(\eta + 1, \dots, n)$. We have a 2-complete swap network on the occupied orbitals. In between every swap layer thereof, we do a 2-complete swap network on the virtual orbitals. For every pair of occupied orbitals and every pair of virtual orbitals, there is a layer in this composite network such that the pairs are simultaneously adjacent. Thus, if we then insert a final 2-swap network with appropriate partitions at every layer, then every set of 2 occupied orbitals and 2 virtual orbitals will be adjacent at some point and a 4-local gate can be implemented on them. There swap depth of just the 2-complete swap networks is $\eta(n - \eta + 1)\tau_2 = \Theta(\eta n)$. Before each one, a 2-swap network is inserted with an average depth of $(n + 2)(3\tau_2 + \tau_4) = \Theta(n)$. Overall, this yields the claimed $\Theta(\eta n^2)$ depth. The coefficient of the leading term in the depth can be halved by accounting for the fact that we are typically interested in implementing only those excitations that are spin-preserving. If we initially order the spin orbitals within the sets of occupied and virtual orbitals by $\uparrow, \downarrow, \downarrow, \uparrow, \uparrow, \dots$, then the parity of the spins of the pairs of orbitals acquainted in each layer of the 1-swap networks alternates, and we only need to do a bipartite 2-swap network when the spin parities of the layers of the two sets coincide.

A more general version of the unitary coupled cluster ansatz is obtained by allowing excitations between any pair of orbitals. Rather than the cluster operators given in Equation 4.10, we use

$$\begin{aligned} T_1 &= \sum_{p,q} t_p^q a_q^\dagger a_p, \\ T_2 &= \sum_{p,q,r,s} t_{p,q}^{r,s} a_r^\dagger a_s^\dagger a_q a_p, \end{aligned} \tag{4.11}$$

where the indices p, q, r, and s are allowed to range over the entire set of orbitals (except that we often disallow excitations that do not preserve spin). It has been shown that the inclusion of these "generalized' singles and doubles greatly increases the ability of unitary coupled cluster to target the kind of strongly correlated states that pose the greatest challenge for quantum chemical calculations on a classical computer [103, 157]. A Trotter step for unitary coupled cluster with generalized singles and doubles may be implemented by a straightforward application of the techniques for implementing 4-local gates described in Figure 4.5. That construction also yields the optimal scaling here, enabling the execution of all $\Theta(n^4)$ gates operations corresponding to the terms in Equation 4.11 using a circuit of depth $\Theta(n^3)$. One possibility for exploiting spin symmetry is as follows. Start with an initial mapping in which the orbitals of one spin are mapped to the first half of the physical qubits and those of the other spin to the second half. Then apply the quartic swap network to each half of the qubits in parallel, thus acquainting all sets of four orbitals with the same spin. Then apply a double bipartite swap network, of the sort used for UCCSD, to acquaint every set of four orbitals such that there are two orbitals of each spin.

As a final example of the utility of a swap network approach to circuit compilation, we describe the implementation of a sparse version of the unitary coupled cluster operator with generalized singles and doubles recently developed by Lee at al. [103]. Rather than the full set of double excitations as in Equation 4.11, this variant of unitary coupled cluster uses only those double excitations that transfer two electrons with opposite spins from one spatial orbital to another. The resulting cluster operators,

$$\begin{aligned} T_1 &= \sum_{p,q,\alpha} t_p^q a_{q\alpha}^\dagger a_{p\alpha}, \\ T_2 &= \sum_{p,q} t_{p,p'}^{q,q} a_{q\uparrow}^\dagger a_{q\downarrow}^\dagger a_{p\downarrow} a_{p\uparrow} \end{aligned} \tag{4.12}$$

contain only $\Theta(n^2)$ terms and can be implemented in $\Theta(n)$ depth using the approach detailed below.

Recall our prior observation that, throughout the execution of a complete 1-swap network, every pair of logical qubits that is initially at distance 2 from each other will remain so. Furthermore, every such pair of logical qubits will become adjacent to every other pair. Therefore we begin by ordering the fermionic modes $(1\uparrow, 2\uparrow, 1\downarrow, 2\downarrow, 3\uparrow, 4\uparrow, 3\downarrow, 4\downarrow, \ldots)$. Then, by executing a 2-complete swap network, we bring the fermionic modes involved in each of the 2-local and 4-local terms in Equation 4.12 adjacent to each other at some point. We show an example for $n = 8$ in Figure 4.7 below.

4.7 Instance-independent embedding for quantum annealing

Quantum annealing is an alternative model of quantum computation for minimizing a classical pseudo-Boolean function $f : \{\pm 1\}^n \to \mathbb{R}$, in which the Hamiltonian is slowly changed from an initial Hamiltonian H_{init} into the problem Hamiltonian H_f, whose ground state(s) we would like to find. Often, the desired Hamiltonian H_f cannot be implemented directly on a physical quantum annealer due to limited connectivity. To overcome this limitation, each logical qubit in H_f can be mapped to a connected set of physical qubits which are coupled

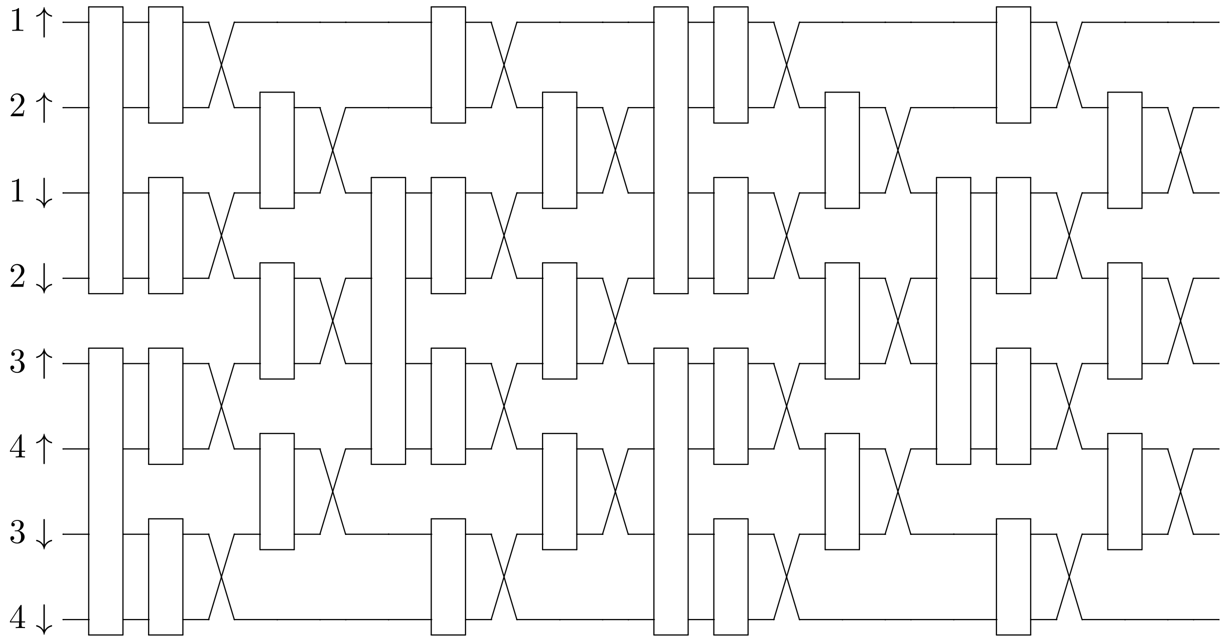

Figure 4.7: The swap network for a UpCCGSD operator (Equation 4.12) with four spatial orbitals. The initial assignment of spin orbitals to qubits is indicated; the important feature is that the two spin orbitals for each spatial orbital are assigned to qubits at distance 2 from each other. They then *stay* at distance 2 from each other throughout the evolution of the swap network (except temporarily at the edges). The swaps are exactly the same as in the standard 1-swap network, except that a layer of 4-local acquaintance opportunities is inserted before every other swap layer, allowing the four spin orbitals corresponding to a pair of spatial orbitals to be acquainted.

together with a ferromagnetic field that induces them to take on the same value. In the standard case in which H_f is 2-local (i.e., f is quadratic), it can be considered as a graph, and this mapping from logical to physical qubits as a minor embedding into the hardware graph. For example, Choi [43] gave a family of minor embeddings of the complete graph into a so-called Triad hardware graph (similar to the Chimera hardware graph used by D-Wave) in which the number of physical qubits scales quadratically with the number of logical qubits, which is optimal for bounded-degree hardware graphs. Zaribafiyan et al. [164] provide a deterministic embedding for Cartesian product graphs.

In practice, problem graphs of interest are usually much sparser than the complete graph, or the Cartesian product graphs, and so using an embedding for the complete graph is likely to use more physical qubits than necessary. Specifically, most problems run on the D-Wave quantum annealer make use of D-Wave's heuristic embedding software [36]. Many practitioners thus use instance-specific embeddings to maximize the use of scarce resources. The problem, however, is the difficulty of finding such instance-specific embeddings. An approach similar to the one we used for quantum circuits can be taken. Instead of using an embedding of either the complete graph (which is trivial to find but resource-inefficient) or a single problem graph (which is harder to find but more resource-efficient), one can use an embedding of a "supergraph" of a class of problem graphs. Such an embedding can be found either manually or algorithmically, but in any case can be reused for any instance in the class with negligible marginal cost. This approach thus strikes a potentially valuable balance between the two existing ones.

4.8 Lower bounds

The optimality of the complete swap network is easy to show. $\binom{n}{2}$ logical gates are executed in n almost perfectly parallelized layers. In a reasonable accounting in which any 2-qubit gate on adjacent qubits can be done in unit time, the logical qubits and swaps can be combined into one. However, for more complicated cases the reasoning becomes more involved. This section gives some methods for lower bounding the depth of solutions to the (unordered) circuit embedding problem. In particular, the lower bounds are on the depth of the 2-qubit swaps only, i.e., the "swap depth". For a bounded-degree physical graph and bounded-locality logical graph, the logical gates that can be executed with a single, fixed mapping of logical to physical qubits, i.e., that after an swap layer, can be executed in $O(1)$ depth. In such cases, which comprise almost all of practical interest, exact lower bounds on the swap depth thus yield scaling lower bounds on the total depth.

4.8.1 Acquaintance time

Benjamini et al. defined [14] the *acquaintance time* of a graph G, denoted $\mathcal{AC}(G)$ as follows. Consider placing an agent at each vertex of the graph and a series of matchings [1] of the graph. Each matching corresponds to simultaneously swapping the agents on the vertices of each edge. Such a a sequence of matchings of G is a *strategy for acquaintance* in G if every pair of agents are adjacent in the graph G at least once. The acquaintance time is the number of rounds (matchings) in the shortest strategy for acquaintance (and is finite if and only if the graph is connected).

This notion of strategies for acquaintance is a useful if limited abstraction for compiling quantum circuits around geometric constraints. As is, a strategy for acquaintance corresponds to a compilation of all 2-local gates in a hardware graph G, with agents corresponding to logical qubits, vertices corresponding to physical qubits, and edges of matchings to swap gates. A gate between two logical qubits can be implemented at any point that that they can become "acquainted". This level of abstraction has the advantage and disadvantage that it disregards the exact nature of the gates. This makes it extremely general but also

[1] A matching is a set of mutually disjoint edges of a graph.

constructions within it somewhat approximate. For example, in a strategy for acquaintance, it is permissible for an agent to become acquainted with more than one other agent in a single round, while the corresponding 2-local gates would need to be implemented sequentially.

Nevertheless, known results about acquaintance times [4, 14] can be interpreted in the context of quantum circuit embedding. For example, that the acquaintance time of the path graph P_n is $n-2$ provides an alternative proof of the optimality of the complete linear 1-swap network. Interestingly, the acquaintance time of the barbell graph B_n (two fully connected halves connected by a single edge) is also $n-2$. Generally, it is known that for a graph G of maximum degree Δ, $\mathcal{AC}(G) = \min\{O(n^2/\Delta), 20\Delta n\}$, which in particular implies that for any graph $\mathcal{AC}(G) = O(n^{3/2})$. There are also hardness results: $\mathcal{AC}(G)$ is NP-hard to approximate within a multiplicative factor of 2 or within any additive constant factor.

A strategy for acquaintance as defined above requires that *every* pair of agents become acquainted. However, it will often be the case that we care only about certain pairs of agents, or larger-sized sets of agents. We now define a generalization of acquaintance time that may be of value in finding lower bounds in such cases. Let H be the hypergraph whose vertices correspond to the agents and whose hyperedges correspond to the sets of agents that we would like to acquaint. We can then define a *strategy for H-acquaintance in G* as an initial (injective) mapping σ of the vertices of H to the vertices of G and a sequence of matchings as above such that, for every edge $\{i_1, \dots, i_k\}$ of H, if agent i_1 is placed on vertex $\sigma(i_1)$ in G, i_2 on vertex $\sigma(i_2)$, and so on, then the set of agents $\{i_1, \dots, i_k\}$ can be acquainted at some point. Whether a set of agents can be acquainted given their locations on the vertices of G can be specified in one of two ways. In the first case, G itself is a hypergraph and the agents can be acquainted if their positions $\{\sigma_t(i_1), \dots, \sigma_t(i_2)\}$ are a hyperedge of G, where $\sigma_t(i)$ is the location of agent i after t rounds. In the alternative, G is a simple graph, and the agents can be acquainted if their positions form a connected subgraph of G. The latter is closer to our application of strategies for acquaintance: the physical graph G specifies on which pairs of qubits a 2-qubit gate can be applied, and higher-locality gates are decomposed using such 2-qubit gates. The *H-acquaintance time* of G, denoted $\mathcal{AC}_H(G)$ then is the minimal size of a strategy for H-acquaintance in G. Note that this definition does not assume that $|V(H)| = |V(G)|$.

4.8.2 Circuit embeddings as minor embeddings

This section assumes that the reader is familiar with the basic ideas of graph minor embeddings and treewidth; see Klymko et al. [101] for a brief introduction to these ideas in a related context. All graphs in this section will be assumed to have edges of size 2.

Consider a strategy for G-acquaintance in Γ with d rounds. Let $H = \Gamma \boxtimes P_{d+1}$ be the strong product of Γ and the path graph on $d+1$ vertices. That is,

$$V(H) = \{(v,t) | v \in V(\Gamma), t \in \{0, \dots, d\}\}, \tag{4.13}$$

$$E(H) = \{\{(v,t),(v',t')\} | v = v' \vee \{v,v'\} \in E(\Gamma), |t-t'| \leq 1\}. \tag{4.14}$$

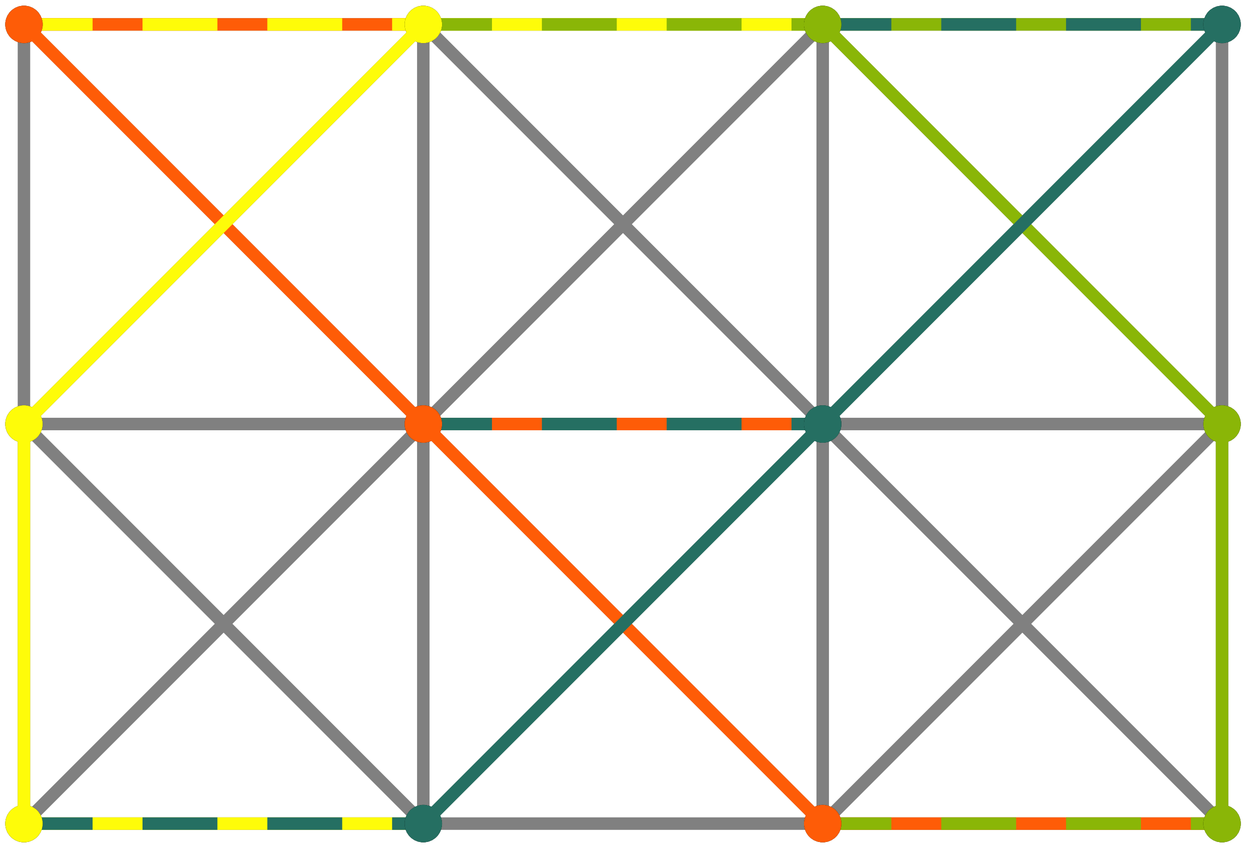

Figure 4.8: A 2-round strategy for K_4-acquaintance in P_4 as a minor embedding of K_4 into $P_4 \boxtimes P_3$. Each set of solid points and lines of a given color indicates a vertex model. Each bi-colored dashed line indicates an edge model. Solid gray lines indicate unused edges of $P_4 \boxtimes P_3$.

The strategy for G-acquaintance in Γ can be interpreted as a graph minor embedding of G into H as follows. Figure 4.8 shows an example for $G = K_4$ and $\Gamma = P_4$. The "agents" are the vertices of G. The vertex model of $v \in G$ is the set of vertices $\{(\sigma_t(v), t) | t \in \{0, \ldots, d\}\} \subset V(H)$ corresponding to the series of assignments of v to vertices of Γ. Note that this vertex model is connected (indeed, a simple path) and that the vertex models of distinct vertices are disjoint, by the properties of an acquaintance strategy. The edge model of an edge $\{v, w\} \in E(G)$ is $\{(\sigma_t(v), t), (\sigma_t(w), t)\} \in E(H)$ for some round t in which the vertices v and w are assigned to adjacent vertices of Γ. For any graphs A and B, if A is a minor of B, then $\text{pw}(A) \le \text{pw}(B)$ and $\text{tw}(A) \le \text{tw}(B)$, because any path or tree decomposition for B can be converted into one for A by edge-contracting the vertex models, without increasing the relevant width. In our case, we have shown that G is a minor of $H = \Gamma \boxtimes P_{d+1}$ whenever there exists a d-round strategy for G-acquaintance in Γ. Therefore,

$$\mathrm{pw}(G) \leq \mathrm{pw}\left(\Gamma \boxtimes P_{\mathcal{AC}_G(\Gamma)+1}\right), \tag{4.15}$$

and similarly for treewidth.

We show now that, for an arbitrary graph G on n vertices, the pathwidth $\mathrm{pw}(G)$ is at most about one more than the G-acquaintance time in the path graph P_n,

$$\mathrm{pw}(G) \leq 2\left\lceil \frac{\mathcal{AC}_G(P_n)}{2} \right\rceil + 1. \tag{4.16}$$

We do so by explicitly constructing a path decomposition of a graph from a strategy for G-acquaintance in P_n. Consider such a strategy and let $\sigma_t(v) \in P_n$ be the assignment of vertex $v \in G$ after round t. We can construct a path decomposition with $n-1$ bags as follows. Each bag corresponds to an edge of P_n and contains all the vertices of G that are assigned to an vertex of P_n adjacent to e. The bags form the path graph P_{n-1} corresponding to the line graph of P_n. Each bag can contain at most $2\lceil d/2 \rceil + 2$ vertices, where d is the number of rounds in the strategy for G-acquaintance. Lastly, the number of rounds in the strategy is at least the minimum number of rounds $\mathcal{AC}_G(P_n)$ and the pathwidth of the graph is at most the width of this decomposition, yielding the desired inequality.

One application of this inequality is yet another lower bound on the swap depth of a complete swap network. Equation 4.16 and the fact that $\mathrm{pw}(K_n) = n-1$ imply that

$$\mathrm{pw}(K_n) = n-1 \leq 2\left\lceil \frac{\mathcal{AC}(P_n)}{2} \right\rceil + 1 \tag{4.17}$$

$$\Rightarrow \quad \frac{n}{2} - 1 \leq \left\lceil \frac{\mathcal{AC}(P_n)}{2} \right\rceil \tag{4.18}$$

$$\Rightarrow \quad \mathcal{AC}(P_n) \geq \begin{cases} n-2, n \text{ odd}, \\ n-3, n \text{ even}. \end{cases} \tag{4.19}$$

Note that Equation 4.16 is not necessarily tight for arbitrary graphs. For example, consider the star graph S_k for large k. It has pathwidth 1 [2], but the minimum swap circuit depth is $\Omega(k)$. More generally, caterpillar graphs exemplify the looseness of the above bound for the same reason; the minimum depth of a swap circuit for any graph scales linearly with the degree of the graph.

4.9 Use case: Non-Orthogonal Variational Quantum Eigensolver

In Ref. 91, we used the swap network for the UpCCGSD ansatz as part of an extension to VQE that we called the Non-othogonal Variational Quantum Eigensolver (NOVQE). NOVQE

[2]Consider the decomposition in which there is a bag for each leaf containing that leaf and the internal vertex.

N_p	Primitive basis functions in diagonal basis
N_a	Functions in compact active space basis
N_b	Blocks in the DG basis
n_κ	Functions in DG block κ
N_d	Total DG functions, $(\sum_\kappa n_\kappa)$

Figure 4.9: Compact description of the notation used throughout this chapter in counting basis functions in different representations for the electronic structure problem. Here discontinuous Galerkin (DG) is the block basis we construct from primitive functions to represent the active space orbitals with a block diagonal Hamiltonian representation.

is essentially a way of increasing the expressiveness of an ansatz in a way that outweighs the slight additional overhead in circuit size and depth. Combining the efficiency of the swap network for UpCCGSD with some additional compilation tricks in the implementation of NOVQE leads to a variational method that can get within chemical accuracy for (small) strongly correlated systems with relatively short and small quantum circuits. The main idea is take a single ansatz (such as UpCCGSD), denoted by $|\phi(\boldsymbol{\theta})\rangle$, and extend it to an ansatz of the form

$$|\psi(\mathbf{c}, \Theta)\rangle = \sum_{i=1}^{M} c_i \, |\phi(\boldsymbol{\theta}_i)\rangle \, , \tag{4.20}$$

where $\mathbf{c} = (c_1, \ldots, c_M)$ and $\Theta = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M)$, i.e., a linear combination of "primitive" ansatz states. In general, these constituent primitive ansatz states are non-orthogonal, hence the name. Crucially, we can evaluate the energy of the "composite" ansatz with relatively little overhead compared to the resources needed for the primitive ansatzes. Specifically, by using the preparation circuits for two primitive ansatz states $|\phi(\boldsymbol{\theta}_i)\rangle$ and $|\phi(\boldsymbol{\theta}_j)\rangle$ in parallel and just two single-qubit-controlled swaps of the registers, we can estimate both the overlap $\langle\phi(\boldsymbol{\theta}_i)|\phi(\boldsymbol{\theta}_j)\rangle$ and the matrix element $\langle\phi(\boldsymbol{\theta}_i)|H|\phi(\boldsymbol{\theta}_j)\rangle$. By solving a generalized eigenvalue problem within the space spanned by $\{|\phi(\boldsymbol{\theta}_i)\rangle : 1 \leq i \leq M\}$, we can then find the coefficients $\mathbf{c}$. (See Ref. 91 for details.)

4.10 Block-diagonal Hamiltonians

This section is excerpted from Ref. 114 with slight modifications. See Fig. 4.9 for an overview of notation used throughout this section.

4.10.1 Discontinuous Galerkin discretization

At a high level, we construct the block-diagonal basis by fitting spatially connected blocks of the primitive basis set to the active basis set, while preserving the properties of the primitive basis set. We therewith interpolate between the primitive basis set and the active basis set.

We will refer to the general class of basis sets that achieve both completeness in some limit and have the diagonal property as primitive basis sets.

Our goal is to systematically compress the active basis set $\{\varphi_p(\mathbf{r})\}_{p=1}^{N_a}$ into a set of orthonormal basis functions partitioned into elements (groups), so that basis functions associated with different elements have mutually disjoint support. Assume that the index set $\Omega = \{1, \dots, N_p\}$ can be partitioned into N_b non-overlapping index sets

$$\mathcal{K} = \{\kappa_1, \kappa_2, \cdots, \kappa_{N_b}\}, \tag{4.21}$$

so that $\cup_{\kappa \in \mathcal{K}} \kappa = \Omega$. Then the matrix Φ can be partitioned into N_b blocks $\Phi_\kappa := [\Phi_{\mu p}]_{\mu \in \kappa}$ for $\kappa \in \mathcal{K}$. Performing the singular value decomposition for Φ_κ,

$$\Phi_\kappa \approx U_\kappa S_\kappa V_\kappa^\dagger, \tag{4.22}$$

where U_κ is a matrix with orthonormal columns corresponding to the leading n_κ singular values up to some truncation tolerance τ, we obtain our compressed basis

$$\phi_{\kappa,j}(\mathbf{r}) = \sum_{\mu \in \kappa} \chi_\mu(\mathbf{r}) (U_\kappa)_{\mu,j}. \tag{4.23}$$

The basis set is adaptively compressed with respect to the given set of basis functions, and are locally supported (in a discrete sense) only on a single index set κ. In the absence of SVD truncation, we clearly have $\text{span}\{\varphi_p\} \subseteq \text{span}\{\phi_{\kappa,j}\}$. We refer to this basis set $\{\phi_{\kappa,j}\}$ as the DG basis set. Note that each DG basis function $\phi_{\kappa,j}$ is a linear combination of primitive basis functions which are themselves continuous, so $\phi_{\kappa,j}$ is also technically continuous in real space. In fact, $\phi_{\kappa,j}$ might not be locally supported in real space if each primitive basis function χ_p is delocalized. When the primitive basis functions are localized, $\phi_{\kappa,j}$ can be very close to a discontinuous function. When computing the projected Hamiltonian, we do not need to evaluate the surface terms in the DG formalism. If we form a block diagonal matrix

$$U = \text{diag}[U_1, \dots, U_{N_b}], \tag{4.24}$$

the total number of basis functions is thus $N_d := \sum_{\kappa \in \mathcal{K}} n_\kappa$. We remark that the number of basis functions n_κ can be different across different elements.

To facilitate the complexity count below we may, without loss of generality, assume that n_κ is a constant and that $N_d = N_b n_\kappa$. Then we have defined a new set of creation and annihilation operators

$$\hat{c}_{\kappa,j}^\dagger = \sum_\mu \hat{b}_\mu^\dagger (U_\kappa)_{\mu j}, \quad \hat{c}_{\kappa,j} = \sum_\mu \hat{b}_\mu (\overline{U}_\kappa)_{\mu j}, \tag{4.25}$$

with $\kappa = 1, \dots, N_b$ and $j = 1, \dots, n_\kappa$ that correspond to the DG basis set.

Unlike the general case, the basis set rotation in Eq. (4.25) is restricted to each element κ. We readily obtain the projected Hamiltonian in the DG basis as

$$\begin{aligned}\hat{H}^{(d)} = &\sum_{\kappa,\kappa';j,j'} h^{(d)}_{\kappa,\kappa';j,j'} \hat{c}^{\dagger}_{\kappa,j} \hat{c}_{\kappa',j'} \\ &+ \frac{1}{2} \sum_{\kappa,\kappa';i,i',j,j'} v^{(d)}_{\kappa,\kappa';i,i',j,j'} \hat{c}^{\dagger}_{\kappa,i} \hat{c}^{\dagger}_{\kappa',i'} \hat{c}_{\kappa',j'} \hat{c}_{\kappa,j}. \end{aligned} \tag{4.26}$$

The matrix elements are

$$h^{(d)}_{\kappa,\kappa';j,j'} = \sum_{\mu\nu} (\overline{U}_{\kappa})_{\mu j} h^{(p)}_{\mu\nu} (U_{\kappa'})_{\nu j'}, \tag{4.27}$$

and

$$v^{(d)}_{\kappa,\kappa';i,i',j,j'} = \sum_{\mu\nu} (\overline{U}_{\kappa})_{\mu i} (\overline{U}_{\kappa'})_{\nu i'} v^{(p)}_{\mu\nu} (U_{\kappa})_{\mu j} (U_{\kappa'})_{\nu j'}. \tag{4.28}$$

In general, the one-body matrix $h^{(d)}$ can be a full dense matrix, but the two-body tensor $v^{(d)}$ always takes a "block diagonal" form in the following sense (it has a specific sparsity pattern). In principle, the two-body interaction in the DG basis set should take the form

$$\frac{1}{2} \sum_{\kappa,\kappa',\lambda,\lambda';i,i',j,j'} v_{\kappa,i;\kappa',i';\lambda,j;\lambda',j'} \hat{c}^{\dagger}_{\kappa,i} \hat{c}^{\dagger}_{\kappa',i'} \hat{c}_{\lambda',j'} \hat{c}_{\lambda,j}. \tag{4.29}$$

Compared to Eq. (4.26), we find that

$$v_{\kappa,i;\kappa',i';\lambda,j;\lambda',j'} = v^{(d)}_{\kappa,\kappa';i,i',j,j'} \delta_{\kappa\lambda} \delta_{\kappa'\lambda'}. \tag{4.30}$$

In other words, v can be viewed as a block diagonal matrix with respect to the grouped indices $(\kappa\kappa', \lambda\lambda')$.

We remark that the convergence of the DG basis set is independent of the choice of the primitive basis set so long as the primitive basis has sufficient degrees of freedom to form a good approximation to the active space functions of interest. At the end of this adaptive procedure, we expect the number of elements in the Hamiltonian to scale as $O(N_b^2 n_\kappa^4)$. However, we expect the number of basis functions required to reach a fixed accuracy within a block (i.e., n_κ) to be bounded by a constant as system size grows, and the scaling with system size becomes $O(N_d^2)$. We substantiate the rapid asymptotic convergence of n_κ for real systems later in this work; however, simple arguments from spatial locality and basis set completeness lead to the same conclusion.

4.10.2 Swap networks for block diagonal Hamiltonians

In the first work using a strictly diagonal basis in quantum computing for chemistry [11], the ability for quantum computers to perform fast Fourier transforms on quantum wavefunctions was exploited to capitalize on the representational advantages of being in either the plane wave

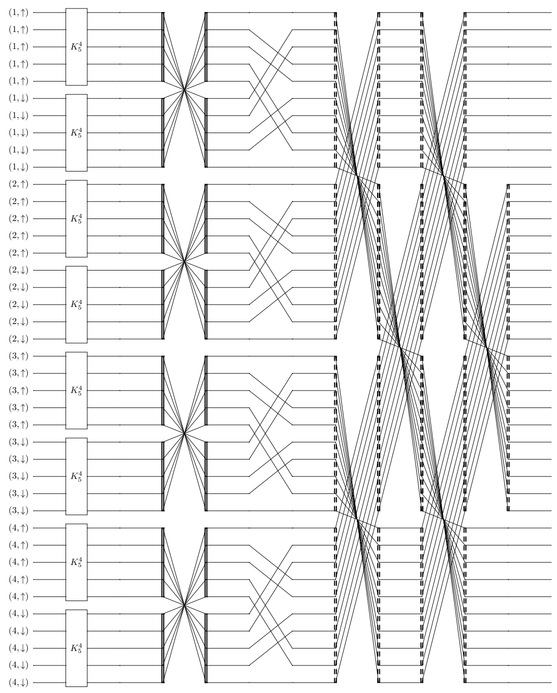

Figure 4.10: Acquaintance strategy for block-diagonal Hamiltonian with $N_b = 4$ and $n_\kappa = 10$. "K_n^4" indicates a 4-complete swap network on n qubits, i.e., one that acquaints the $\binom{n}{4}$ subsets of 4 qubits with each other. The other gates are double bipartite swap networks, explained in Figures 4.12 and 4.13.

basis or its Fourier-transformed dual. That method was originally restricted to Hamiltonians with that particular structure in the coefficients, similar to split-operator Fourier transform methods used in classical simulation of quantum systems. However, it was soon realized that the structure of any diagonal Hamiltonian could be similarly exploited. This generalization used a linear, fermionic swap network to achieve perfect parallelization of a Trotter step with depth that scales linearly in the number of orbitals [100], even when gates are restricted to act on nearest neighbors of a line of qubits.

Fermionic swap networks are analogous to sorting networks from traditional computer science except built upon the primitive of the fermionic swap operation,

$$\hat{f}^{pq}_{\text{swap}} = 1 + \hat{a}^\dagger_p \hat{a}_q + \hat{a}^\dagger_q \hat{a}_p - \hat{a}^\dagger_p \hat{a}_p - \hat{a}^\dagger_q a_q, \tag{4.31}$$

$$\hat{f}^{pq}_{\text{swap}} \hat{a}^\dagger_p (\hat{f}^{pq}_{\text{swap}})^\dagger = \hat{a}^\dagger_q, \tag{4.32}$$

where $\hat{f}^{pq}_{\text{swap}}$ is the fermionic swap that swaps the labeling of modes. The fermionic swap operation was introduced in [33] and also studied in the context of tensor networks. The difference between such a swap and a traditional swap is that by swapping fermionic modes instead of assignments to qubits, non-local parity strings used to enforce the fermionic anti-commutation relations can be avoided.

The basic idea of the linear swap network is to fermionic swap all neighboring qubits, interact them with their current neighbors, and repeat until all qubits have interacted with each other. Since the introduction and use of these linear fermionic swap networks in quantum algorithms, they have been generalized for use in non-diagonal Hamiltonians with some overhead. For example, the quantum chemistry Hamiltonian can be decomposed into a sum of diagonal Hamiltonians (each in a rotated basis) using techniques similar to Cholesky or density fitting methods, where each diagonal Hamiltonian can then be implemented in sequence [121].

For the general Hamiltonian in quantum chemistry, which is non-diagonal, a generalized swap network that works directly with such Hamiltonians was developed [125]. This network implements time steps for generic $O(N_a^4)$ Hamiltonians in a time that scales as $O(N_a^3)$, and we take advantage of it here with specializations for the block diagonal structure. To implement a Trotter step of the Hamiltonian, the swap network dynamically updates the mapping from qubits to orbitals so that for each term in the Hamiltonian, the involved orbitals are mapped to adjacent qubits. We say that a swap network "acquaints" a set of orbitals when it brings them together at some point in this way, and represent that point by an empty box in the circuit diagrams, which acts as a placeholder for the logical gate to be executed there. Prior work utilizing swap networks has applied them to two extremal regimes with respect to the structure of the two-electron terms in the Hamiltonian: the strictly diagonal case, which can be implemented with $O(N_p)$ depth [100]; and the fully general case, which can be implemented in $O(N_a^3)$ [125]. Here we show how to interpolate between these to achieve $O(N_b n_\kappa^3) = O(N_d n_\kappa^2)$ depth for block-diagonal Hamiltonians. (For simplicity, in this section we will assume that all blocks have the same size n_κ, but the techniques generalize in a straightforward way to non-uniform block sizes.)

We focus on how to implement the quartic terms in the Hamiltonian (i.e., two-electron terms involving four distinct spin orbitals). The lower-order terms can be addressed with negligible additional resources by incorporating them into the quartic terms. The quartic terms in the block-diagonal Hamiltonian satisfy the following properties:

1. Two orbitals are from one block κ and two orbitals are from another block κ' (or all four from the same block when $\kappa = \kappa'$).

2. The orbital spins have even parity (i.e., all up, all down, or two and two).

We will exploit both of these properties in constructing our swap network, which uses primitives originally designed for implementing unitary coupled cluster [125].

Figure 4.10 shows the overall swap network. Initially, the orbitals are arranged on the line in lexicographical ordering; only the block index κ and spin are indicated for concision. The logic of the strategy is as follows:

1. The first layer acquaints all sets of four spin orbitals within each block in which all four orbitals have the same spin. This is achieved by a "4-complete" swap network on

each half-block of orbitals, denoted by $K^4_{n_\kappa/2}$ because the sets of orbitals it acquaints correspond to the edges of a complete 4-uniform hypergraph; it has depth $O(n_\kappa^3)$. Note that the edges of the complete k-uniform hypergraph K_n^k on n are the $\binom{n}{k}$ sets of k vertices. The "uniform" qualifier indicates that all of the hyperedges have the same number of vertices. See Ref. 125 for details.

2. The second layer acquaints all sets of four spin orbitals within each block in which two orbitals have spin up and the other two have spin down. This is achieved by a "double bipartite" swap network on each block in depth $O(n_\kappa^3)$; see Figure 4.12.

3. The third layer permutes, in $O(n_\kappa)$ depth, the orbitals within each block in preparation for the inter-block acquaintances to follow.

4. The rest of the strategy consists of N_b alternating layers that acquaint pairs of parts. In each layer, each block of qubits is paired up with an adjacent one and a "balanced double bipartite" swap network is executed on the pair of blocks; see Figure 4.13. Each balanced double bipartite swap network acquaints the sets of four orbitals containing two from each block and with even ("balanced") spin parity. This also has the effect of swapping the blocks, so overall a balanced double bipartite swap network is applied to every pair of blocks. Each double bipartite swap network has depth $O(n_\kappa^3)$.

Overall, the depth is $O(N_b n_\kappa^3)$, dominated by the latter swap networks that effect the inter-block interactions. The components of this approach are explained in more detail below.

4.10.3 Swap network sub-circuits

In this section, we give some more detail about the components of the swap network described in Section 4.10.2. Recall the structure of the overall swap network:

1. A 4-complete swap network within each half-block. This acquaints all sets of 4 orbitals with the same spin and within each block.

2. A double bipartite swap network on each block. This acquaints all sets of 4 orbitals with no net spin and within each block. Details of the construction are given in Figure 4.12.

3. A permutation within each block. This changes the orbital to qubit mapping in preparation for the next stage.

4. Alternating layers of balanced double bipartite swap networks. Each balanced double bipartite swap network acquaints, for some pair of blocks, all sets of 4 orbitals with an even number of each spin and with two orbitals from each block. The N_b alternating layers ensure that every pair of blocks is involved together in some balanced double bipartite swap network. Details of the construction of a double bipartite swap network are given in Figure 4.13.

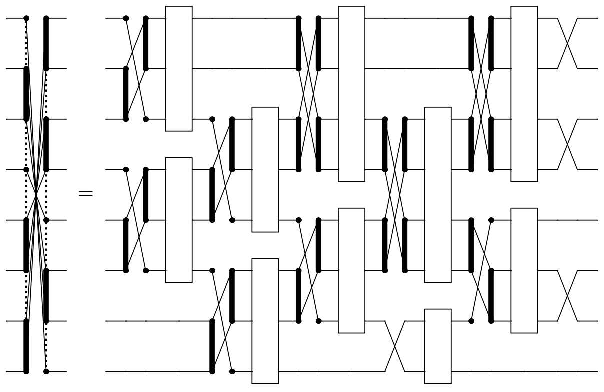

Figure 4.11: Notation and decomposition for a $\mathcal{P}$-swap network with partition sizes $(1, 2, 1, 2, 2)$. A $\mathcal{P}$-swap network for a partition $(P_1, P_2, \ldots, P_{|\mathcal{P}|})$ of the qubits $\bigcup_i P_i$ acquaints every union of a pair of parts, i.e., $\{P \cup P' | P, P' \in \mathcal{P}\}$. At a high level, the structure is similar to that of the simple linear swap network, except that instead of single qubits being swapped, groups are (i.e., the parts of the partition $\mathcal{P}$). There are $|\mathcal{P}|$ layers of generalized swap gates, each of which swaps sets of qubits. For more details, see [125].

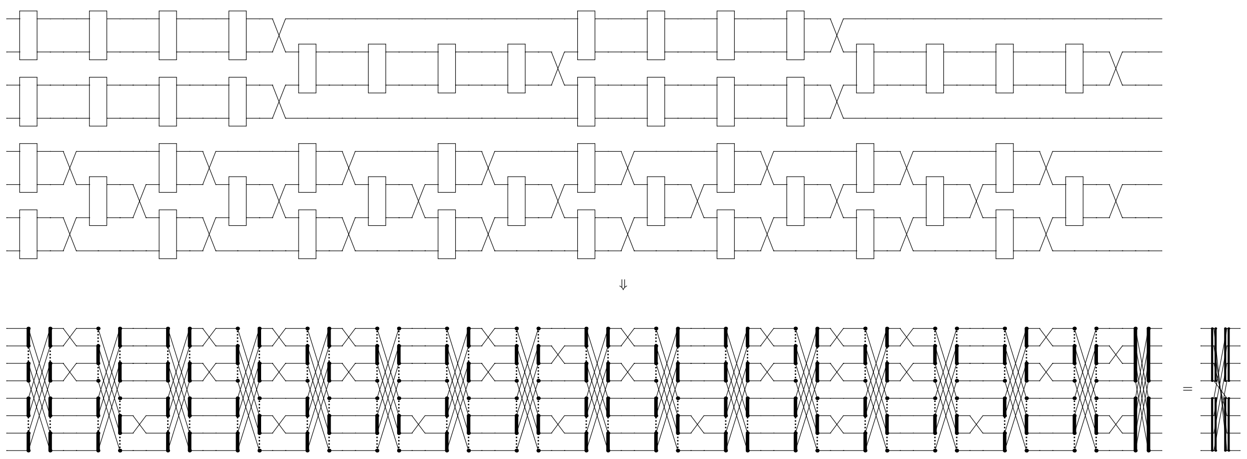

Figure 4.12: Construction of the double bipartite swap network, with parts of size 4. The top half of the top circuit contains the same swap gates as a linear swap network but with additional acquaintance opportunities. In the bottom half of the top circuit are 4 linear swap networks in a row, one for each acquaintance layer of the linear swap network in the top half, which is copied for each acquaintance layer of the bottom half. Overall, for every set of of four orbitals consisting of two from the top part and two from the bottom part, there is a layer in the circuit in which both pairs are simultaneously acquainted. The bottom circuit, depicting the double bipartite swap network, is formed by replacing each such acquaintance layer in the top circuit with a $\mathcal{P}$-swap network, where a pair of qubits acquainted in the top circuit corresponds to a part of the partition $\mathcal{P}$. The $\mathcal{P}$-swap network acquaints the union of each pair of pairs; see Fig. 4.11. The final gate ensures that overall effect is to shift the parts.

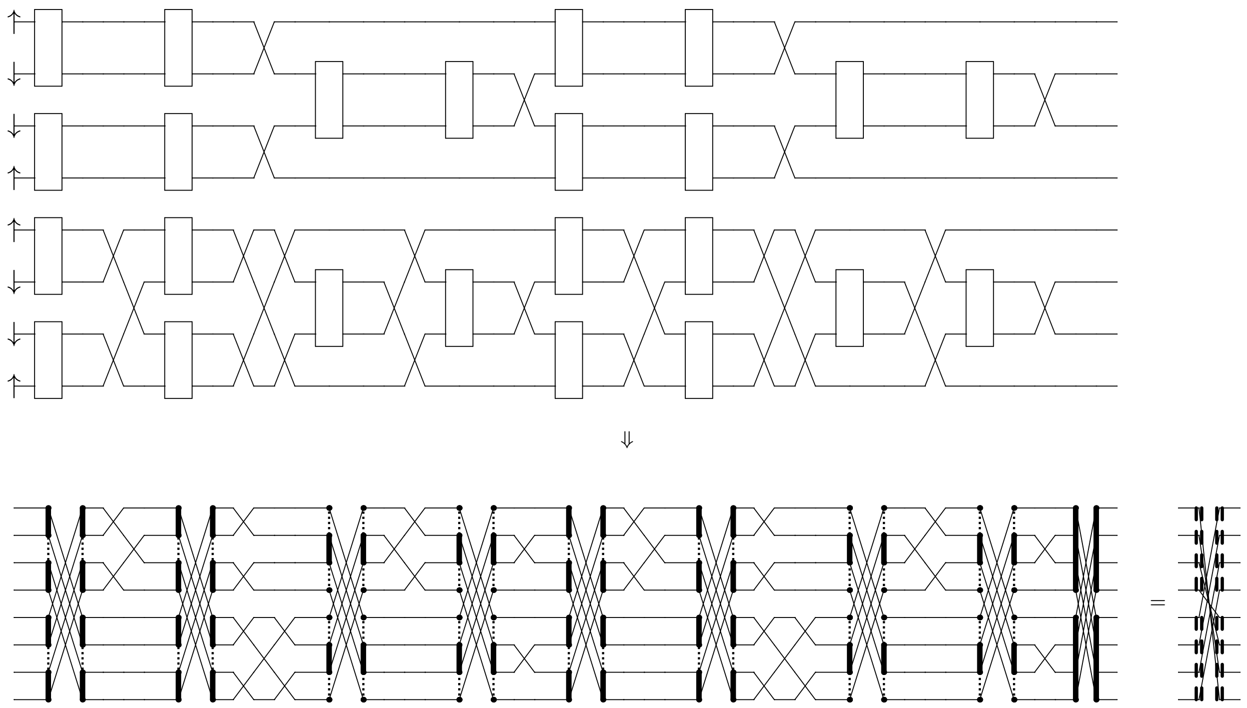

Figure 4.13: Construction of the balanced double bipartite swap network. Similar to the double bipartite swap network, except that pairs of orbitals from each part are only acquainted when their spins have the same parity. The spins of the orbitals in the initial mapping of qubits to orbitals are indicated in the top left.

Chapter 5

Classical optimization and quantum computing

This chapter contains a brief summary of work by my collaborators and me at the intersection of classical optimization and quantum computation. It primarily consists of quantum algorithms (both rigorous and heuristic) for solving classical optimization problems.

5.1 Quantum Alternating Operator Ansatz

This section describes a framework my collaborators and I introduced in Refs. 78 and 77 that generalizes the Quantum Approximate Optimization Algorithm, one of the most popular heuristic quantum algorithms for classical optimization. The text is mostly mostly exerpted from Ref. 77, with some modification. See Ref. 77 for detailed examples of mappings and a compendium with brief descriptions of mappings for a diverse array of problems.

5.1.1 Introduction

Today, challenging computational problems arising in the practical world are frequently tackled by heuristic algorithms. These algorithms are empirically shown to be effective, but they have not been analytically proven to be the best approach, or even to outperform the best approach of the previous year. Until recently, empirical investigation of quantum algorithms has been limited to tiny problems, given the typically exponential overhead of simulating quantum algorithms on classical processors. As prototype quantum hardware emerges that enables experimentation beyond what is reachable by even the world's largest supercomputers, we come into a new era for quantum heuristic algorithms.

One of the most popular heuristics is Farhi et al.'s quantum approximate optimization algorithm, a quantum gate-model meta-heuristic which alternates between applying unitaries drawn from two families, a cost function based unitary family $U_{\mathrm{P}}(\gamma) = e^{-i\gamma H_f}$ and a family of mixing unitaries $U_{\mathrm{M}}(\beta) = e^{-i\beta H_B}$, for some fixed cost function based Hamiltonian H_f

and some fixed mixing Hamiltonian H_B. Here, we formally describe a *quantum alternating operator ansatz* (QAOA), extending the approach of Farhi et al. [62] to allow alternation between more general families of operators. This ansatz supports the representation of a much more varied, and potentially more useful, set of states than the original formulation. Our extension is particularly useful for situations in which the feasible subspace is smaller than the full space, such as when the optimization is over solutions that must satisfy hard constraints. Intuitively, mixing operators that restrict the search to the feasible subspace should result in better-performing algorithms. Our expansion includes families of mixing operators $U_{\mathrm{M}}(\beta)$ that cannot be expressed, as a family, as $e^{-i\beta H_B}$ for a fixed mixing Hamiltonian H_B. As we shall see, expanding the design space of families of one-parameter mixing operators allowed enables the ansatz to support more efficiently implementable mixers than was possible in the original framework. More efficient implementation enables earlier experimental exploration of an alternating operator approach, in the spirit of the quantum approximate optimization algorithm, to a wide variety of approximate optimization, exact optimization, and sampling problems.

We reworked the original acronym so that "QAOA" continues to apply to both prior work and future work to be done in this more general framework. More generally, the reworked acronym refers to a set of states representable in a certain form, and so can be used without confusion in contexts other than approximate optimization, e.g., exact optimization and sampling. (Incidentally, this reworking also removes the redundancy from the now commonly-used phrase "QAOA algorithm".) The class QAOA$_p$ consists of level-p QAOA circuits, in which there are p iterations of applying a classical Hamiltonian (derived from the cost function) and a mixing Hamiltonian. The $2p$ parameters of the algorithm specify the durations for which each of these two Hamiltonians are applied.

We comment here on the relation between these mappings and those for non-gate-model quantum computing, such as quantum annealing (QA). Because current quantum annealers have a fixed driver (the mixing Hamiltonian in the QA setting), all problem dependence must be captured in the cost Hamiltonian on such devices. The general strategy is to incorporate the hard constraints as penalty terms in the cost function and then convert the cost function to a cost Hamiltonian [19, 76, 108, 136]. However, this approach means that the algorithm must search a much larger space than if the evolution were confined to feasible configurations, making the search less efficient than if it were possible to constrain the evolution. This issue, and other drawbacks, led Hen and Spedalieri [85] and Hen and Sarandy [84] to suggest a different approach for adiabatic quantum optimization (AQO), in which the standard driver is replaced by an alternative driver that confines the evolution to the feasible subspace. Their approach resembles a restricted class, H-QAOA (defined below), of QAOA algorithms. While some of our mapppings, e.g., H-QAOA mappings of graph coloring, graph partitioning, and not-all-equal 3-SAT, are close to those in Refs. 84, 85, other mappings we describe, including for these problems, are quite different and take advantage of the more general families of mixers supported by this ansatz. Indeed, while QAOA mappings are different from quantum annealing mappings, with most of the design effort going into the mixing operator rather than the cost function based phase separator, QAOA algorithms, like QA and AQO, but unlike most other quantum algorithms, are relatively easy for people familiar with classical computer science but not quantum computing to design, as we illustrate here.

Prior work suggests the power and flexibility of QAOA circuits. Farhi et al. [63] exhibited a QAOA_1 algorithm that beat the existing best approximation bound for efficient classical algorithms for the problem E3Lin2, only to inspire a better classical algorithm [12]. Jiang et al. [93] demonstrated that the class of QAOA circuits is powerful enough to obtain the $\Theta(\sqrt{2^n})$ query complexity on Grover's problem and also provided the first algorithm within the QAOA framework to show a quantum advantage for a finite number of iterations greater than two. Farhi and Harrow [64] proved that, under reasonable complexity assumptions, the output distribution of even QAOA_1 circuits cannot be efficiently sampled classically. Yang et al. [163] proved that for evolution under a Hamiltonian that is the weighted sum of Hamiltonian terms, with the weights allowed to vary in time, the optimal control is (essentially always) bang-bang, i.e., constant magnitude, of either the maximum or minimum allowed weight, for each of the terms in the Hamiltonian at any given time. Their work implies that QAOA circuits with the right parameters are optimal among Hamiltonians of the form $H(s) = \big(1 - f(s)\big)H_B + f(s)H_C$, where $f(s)$ is a real function in the range $[0, 1]$. It remains an open question whether QAOA provides a quantum advantage over classical algorithms for approximate optimization, either in terms of the quality of approximate solution returned, or the speed of achieving such an approximation.

Since the the publication of this work, the approach we proposed to deal with constrained optimization problems has been applied to a benchmarking study on graph-coloring problems [154] and a protein-folding optimization problem [66]. QAOA also provides a viable platform to study quantum circuit compilation to realistic architectures [61, 102, 151]. (See also Section 5.2.) Applications and extensions of QAOA beyond optimization include state preparation [88] and machine learning [128, 152]. A different approach in the setting of quantum walks to QAOA for constrained problems has recently been proposed [112], and very recently, Lloyd showed that the QAOA framework with a carefully constructed cost Hamiltonian can be made universal for quantum computation [107].

5.1.2 The Original Quantum Approximate Optimization Algorithm

We now give an overview of the original quantum approximation optimization algorithm proposed in Ref. 62.

Consider an unconstrained optimization problem on n-bit strings we seek to approximate. Given a problem instance, the algorithm is specified by two Hamiltonians H_P and H_M, and $2p$ real parameters $\gamma_1, \ldots, \gamma_p, \beta_1, \ldots, \beta_p$. The main details are the following:

- The *phase Hamiltonian* H_P encodes the cost function f to be optimized, i.e., acts diagonally on n-qubit computational basis states as:

$$H_\mathrm{P} \left|\mathbf{y}\right\rangle = f(\mathbf{y}) \left|\mathbf{y}\right\rangle .$$

- The *mixing Hamiltonian* H_{M} is the transverse field Hamiltonian:

$$H_{\mathrm{M}} = \sum_{j=1}^{n} X_j,$$

  where X_j is the Pauli X operator acting on the jth qubit. (The Pauli X operator acts as a bit flip, i.e., $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$.)

- The initial state is selected to be the equal superposition state of all possible solutions:

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle \ ,$$

  which is also the ground-state of $-H_M$ and is used similarly in AQO [62].

- A *parameterized quantum state* is created by alternately applying Hamiltonians H_{P} and H_{M} for p rounds, where the duration in round j is specified by the parameters γ_j and β_j, respectively:

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = e^{-i\beta_p H_{\mathrm{M}}} e^{-i\gamma_p H_{\mathrm{P}}} \dots e^{-i\beta_2 H_{\mathrm{M}}} e^{-i\gamma_2 H_{\mathrm{P}}} e^{-i\beta_1 H_{\mathrm{M}}} e^{-i\gamma_1 H_{\mathrm{P}}} |s\rangle \, .$$

- A computational basis measurement is performed on the state, which returns a candidate solution $\mathbf{y}$ with probability $|\langle \mathbf{y}|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle|^2$. With many repetitions of the above state preparation and measurement, the expected value of the cost function over the returned solution samples is given by

$$\langle f \rangle = \langle \boldsymbol{\beta}, \boldsymbol{\gamma}| H_{\mathrm{P}} |\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle \, ,$$

  which can be statistically estimated from the samples produced. (For a constraint satisfaction problem with m constraints, fom Chebyshev's inequality it follows that an outcome achieving at least $\langle f \rangle - 1$ will be obtained with probability at least $1 - 1/m$ after $O(m^2)$ repetitions.)

- The above steps may then be repeated altogether, with updated sets of time parameters, as part of a classical optimization loop (such as gradient descent or other approaches) used to optimize the algorithm parameters with respect to an objective such as $\langle f \rangle$.

- The best problem solution found overall is returned.

A key to success for the algorithm is the selection or discovery of good values for the parameters $\gamma_1, \dots, \gamma_p, \beta_1, \dots, \beta_p$, which result in good approximate solutions. In some cases, where the analysis is tractable, such angles may be found analytically [93, 156]. Parameter setting strategies for QAOA and for the general class of variational quantum algorithms remains an active area of research [75, 116].

We now turn to our generalized QAOA framework, which is the main subject of this section.

5.1.3 The Quantum Alternating Operator Ansatz (QAOA)

Here, we formally describe the quantum alternating operator ansatz, extending the approach of Farhi et al. [62]. QAOA, in our sense, encompasses a more general class of quantum states that may be algorithmically accessible and useful. We focus here on the application of QAOA to approximate optimization, though it may also be used in exact optimization [93, 158] and sampling [64].

An instance of an *optimization problem* is a pair (F, f), where F is the *domain* (set of feasible points) and $f : F \to \mathbb{R}$ is the *objective function* to be optimized (minimized or maximized). Let $\mathcal{F}$ be the Hilbert space of dimension $|F|$, whose standard basis we take to be $\{|\mathbf{x}\rangle : \mathbf{x} \in F\}$. Generalizing Reference [62], a QAOA circuit is characterized by two parameterized families of operators on $\mathcal{F}$:

- A family of *phase-separation operators* $U_{\mathrm{P}}(\gamma)$ that depends on the *objective function* f, and;
- A family of *mixing operators* $U_{\mathrm{M}}(\beta)$ that depends on the domain and its structure,

where β and γ are real parameters. Specifically, a QAOA$_p$ circuit consists of p alternating applications of operators from these two families:

$$Q_p(\boldsymbol{\beta}, \boldsymbol{\gamma}) = U_{\mathrm{M}}(\beta_p)U_{\mathrm{P}}(\gamma_p)\cdots U_{\mathrm{M}}(\beta_1)U_{\mathrm{P}}(\gamma_1). \tag{5.1}$$

This quantum alternating operator ansatz (QAOA) consists of the states representable as the application of such a circuit to a suitably simple initial state $|s\rangle$:

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = Q_p(\boldsymbol{\beta}, \boldsymbol{\gamma})\,|s\rangle\,. \tag{5.2}$$

For a given optimization problem, a *QAOA mapping* of a problem consists of a family of phase-separation operators, a family of mixing operators, and a starting state. The circuits for the original quantum approximate optimization algorithm fit within this paradigm, with unitaries of the form $e^{-i\gamma H_{\mathrm{P}}}$ and $e^{-i\beta H_{\mathrm{M}}}$, with parameters γ and β indicating the time for which a fixed Hamiltonian is applied.

The domain will usually be expressed as the feasible subset of a larger *configuration space*, specified by a set of problem *constraints*. For implementation on expected near-term quantum hardware, each configuration space will need to be *encoded* into a subspace of a Hilbert space of a multiqubit system, with the domain corresponding to a *feasible subspace* of the configuration space. For each domain, there are many possible mixing operators. As we will see, using more general one-parameter families of unitaries enables more efficiently implementable mixers that preserve the feasible subspace. Given a domain, an encoding of its configuration space, a phase separator, and a mixer, there are a variety of *compilations* of the phase separator and mixer to circuits that act on qubits.

For any function f, not just an objective (cost) function, we define H_f to be the quantum Hamiltonian that acts as f on basis states as:

$$H_f\,|\mathbf{x}\rangle = f(\mathbf{x})\,|\mathbf{x}\rangle\,. \tag{5.3}$$

In prior work, the domain F was the set of all n-bit strings, $U_{\mathrm{P}}(\gamma) = e^{-i\gamma H_f}$, and $U_{\mathrm{M}}(\beta) = e^{-i\beta H_B}$. Furthermore, with just one exception, the mixing Hamiltonian was $H_B = \sum_{j=1}^{n} X_j$. We used the notation X_j, Y_j, Z_j to indicate the Pauli matrices X, Y, and Z acting on the jth qubit. The corresponding parameterized unitaries are denoted by $X_j(\theta) = e^{-i\theta X_j}$ and similarly for Y_j and Z_j. The one exception is Section VIII of Ref. 62, which discusses a variant for the maximum independent set problem, in which F is the set of bitstrings corresponding to the independent sets of a graph, the phase separator depends on the cost function as above, and the mixing operator is $U_{\mathrm{M}}(\beta) = e^{-i\beta H_B}$, where H_B is such that:

$$\langle \mathbf{x}|H_B|\mathbf{y}\rangle = \begin{cases} 1, & \mathbf{x}, \mathbf{y} \in F \text{ and } \mathrm{Ham}(\mathbf{x}, \mathbf{y}) = 1, \\ 0, & \text{otherwise,} \end{cases} \tag{5.4}$$

which connects feasible qubit computational basis states with unit Hamming distance (Ham). Section VIII of Ref. 62 does not discuss the implementability of $U_{\mathrm{M}}(\beta)$. A closely related generalization of QAOA for problems with hard constraints based on quantum walks has recently been proposed [112]. However, row-computable feasibility oracles are required to enable mixing between feasible states, which are likely to be more expensive to implement in practice than the approach presented here.

We extended and formalized the approach of Section VIII of Ref. 62 with an eye to implementability, both in the short and long term. We also built on a theory developed for adiabatic quantum optimization (AQO) by Hen and Spedalieri [85] and Hen and Sarandy [84], though the gate-model setting of QAOA leads to different implementation considerations than those for AQO. For example, Hen et al. identified driver Hamiltonians of the form $H_{\mathrm{M}} = \sum_{j,k} H_{j,k}$, where $H_{j,k} = X_j X_k + Y_j Y_k$, as useful in the AQO setting for a variety of optimization problems with hard and soft constraints; such mixers restrict the mixing to the feasible subspace defined by the hard constraints. Analogously, the unitary $U_{\mathrm{M}} = e^{-i\beta H_{\mathrm{M}}}$ meets our criteria, discussed in Section 5.1.3.1, for good mixing for a variety of optimization problems, including those considered in Refs. 84, 85. Since $H_{j,k}$ and $H_{i,l}$ do not commute when $|\{j,k\} \cap \{i,l\}| = 1$, compiling U_{M} to two-qubit gates is nontrivial. One could Trotterize, but it may be more efficient and just as effective to use an alternative mixing operator, such as $U_{\mathrm{M}} = e^{-i\beta H_{S_r}} \cdots e^{-i\beta H_{S_2}} e^{-i\beta H_{S_1}}$, where the pairs of qubits have been partitioned into r subsets $\{S_i\}_i$ containing only disjoint pairs, motivating in part our more general ansatz.

We define as "Hamiltonian-based QAOA" (H-QAOA) the class of QAOA circuits in which both the phase separator family $U_{\mathrm{P}}(\gamma) = e^{-i\gamma H_{\mathrm{P}}}$ and the mixing operator family $U_{\mathrm{M}}(\beta) = e^{-i\beta H_{\mathrm{M}}}$ correspond to time evolution under some Hamiltonians H_{P} and H_{M}, respectively. (In the example mappings to follow, we consider only phase separators $U_{\mathrm{P}}(\gamma) = \sum_{\mathbf{x}} e^{-i\gamma g(\mathbf{x})} |\mathbf{x}\rangle\langle\mathbf{x}|$ that correspond to classical functions and thus also correspond to time evolution under some (potentially nonlocal) Hamiltonians, though more general types of phase separators may be considered). We further define "local Hamiltonian-based QAOA" (LH-QAOA) as the subclass of H-QAOA in which the Hamiltonian H_{M} is a sum of (polynomially many) local terms.

Before discussing design criteria, we briefly mention that there are obvious generalizations in which U_{P} and U_{M} are taken from families parameterized by more than a single parameter.

For example, in Ref. 61, a different parameter for every term in the Hamiltonian is considered. We only consider the case of one-dimensional families, given that it is a sufficiently rich area of study, with the task of finding good parameters $\gamma_1, \ldots, \gamma_p$, and $\beta_1, \ldots, \beta_p$ already challenging enough due to the curse of dimensionality [156]. A larger parameter space may support more effective circuits but increases the difficulty of finding such circuits by opening up the design space and making the parameter setting more difficult.

We remark that the quantum gate-model setting offers several advantages over Hamiltonian-based algorithms such as AQO and quantum annealing. Higher order (k-local) interactions may be compiled down to two-local gates, and compilations using SWAP gates [28, 151] enable the implementation of quantum operations between qubits that are non-neighboring in the physical hardware; indeed, locality and connectivity are both well-known bottlenecks for physical quantum annealing devices. In the longer term, once mature quantum hardware has been built, quantum error correction can be applied to robustly implement QAOA.

5.1.3.1 Design Criteria

Here, we briefly specify design criteria for the three components of a QAOA mapping of a problem. We expect that as exploration of QAOA proceeds, these design criteria will be strengthened and will depend on the context in which the ansatz is used. For example, when the aim is a polynomial-time quantum circuit, the components should have more stringent bounds on their complexity; without such bounds, the ansatz is not useful as a model for a strict subset of states producible via polynomially-sized quantum circuits. On the other hand, when the computation is expected to grow exponentially, a simple polynomial bound on the depth of these operators might not be reasonable. One example might be for exact optimization of the problems considered here; for these problems, the worst case algorithmic complexity is exponential, but it is worth exploring whether QAOA might outperform classical heuristics in expanding the tractable range for some problems.

Initial state. We require that the initial state $|s\rangle$ be *trivial* to implement, by which we mean that it can be created by a constant-depth (in the size of the problem) quantum circuit from the $|0\ldots0\rangle$ state. Here, we often take as our initial state a single feasible solution, usually implementable by a depth-1 circuit consisting of single-qubit bit-flip operations X. Because in such a case the initial phase operator only applies a global phase, we may want to consider the algorithm as starting with a single-mixing operator $U_{\mathrm{M}}(\beta_0)$ to the initial state as a first step. In the quantum approximate optimization algorithm, the standard starting state $|+\cdots+\rangle$ is obtained by a depth-1 circuit that applies a Hadamard H gate to each of the qubits in the $|0\ldots0\rangle$ state.

This criterion could be relaxed to logarithmic depth if needed. It should not be relaxed too much: relaxing the criterion to polynomial depth would obviate the usefulness of the ansatz as a model for a strict subset of states producible via polynomially-sized quantum circuits. Algorithms with more complex initial states should be considered hybrid algorithms, with an initialization part and a QAOA part. Such algorithms are of interest in cases when one expects the computation to grow exponentially, such as is the case for exact optimization for many of the problems here, but might still outperform classical heuristics in expanding the tractable range.

Phase-separation unitaries. We require the family of phase-separation operators to be diagonal in the computational basis. In almost all cases, we take $U_\mathrm{P}(\gamma) = e^{-i\gamma H_f}$, where f is the objective function.

Mixing unitaries (or "mixers"). We require the family of mixing operators $U_\mathrm{M}(\beta)$ to:

- Preserve the feasible subspace: For all values of the parameter β, the resulting unitary takes feasible states to feasible states, and;
- Provide transitions between all pairs of states corresponding to feasible points. More concretely, for any pair of feasible computational-basis states $\mathbf{x}, \mathbf{y} \in F$, there is some parameter value β^* and some positive integer r such that the corresponding mixer connects those two states: $|\langle \mathbf{x} \,|\, U_\mathrm{M}^r(\beta^*) \,|\, \mathbf{y} \rangle| > 0$.

In some cases, we may want to relax some of these criteria. For example, if a QAOA circuit is being used as a subroutine within a hybrid quantum-classical algorithm, or in a broader quantum algorithm, we may use starting states informed by previous runs and thus allow mixing operators that mix less.

This framework can be used in many different contexts. Depending on the context, different measures of success are appropriate. As indicated by the name, the original motivation for Farhi et al.'s work was to develop a quantum approximation algorithm, one for which rigorous bounds on the approximation ratio can be proven [62, 63]. The same style of algorithm was then applied to exact optimization [158] and sampling [64], which have different measures of success. In certain cases, rigorous performance guarantees can be provided in these contexts, e.g., for the Grover problem in Ref. 93. Alternatively, it can be applied as a heuristic approach for any of exact optimization, approximate optimization, or sampling. In these cases, the measure of success is not in terms of rigorous analytical bounds, but rather empirical typical time-to-solution or approximation ratio or sample quality within a given time. Our approach facilitates low-resource constructions that support empirical evaluation of QAOA as a heuristic for a variety of combinatorial optimization problems, and is agnostic as to which success criterion is being used for evaluation.

5.2 Constraint programming

Constraint programming (CP) is both a model for specifying computational problems and a particular approaching for solving them. A CP instance is specified by a set of variables together with a domain for each of those variables and a set of constraints acting on those variables. Importantly, there is no restriction on the arity of the constraints; in general, CP problems include so-called *global* constraints such as the ALLDIFFERENT constraint, which requires that the values assigned to the variables are all different. While this can be equivalently expressed as $\binom{n}{2}$ inequality constraints, doing so obscures the global structure that a solver could exploit. CP as a problem-solving paradigm typically involves *branch-and-infer*. Branch-and-infer is an extension of backtracking search where at each step inference is performed, based on the global constrants, to filter out values from the domains of variables that cannot be part of an eventual solution. An instantiation of branch-and-infer consists of three ingredients: a *variable-selection* heuristic that chooses which variable to branch on, a *value-selection* heuristic that orders the values in the selected variable's domains, and a propogation function that prunes the domains of all the variables based on the value assigned to the variable selected by the heuristics. Together, these completely specify a search tree and depth-first-search ordering of the nodes therein. The computational cost of the branch-and-infer is determined by the number of nodes explored before reaching a satisfying solution (if one exists) and the aggregate cost of the heuristics and filtering over all of those nodes. In Section 5.2.1, we describe how quantum algorithms can be used to speed up this branch-and-infer procedure. In Section 5.2.2, we describe how CP can be used as a tool for the compilation of quantum circuits to real hardware.

5.2.1 Quantum algorithms for constraint programming

This section summarizes Ref. 29, which shows how existing quantum algorithms can be combined to yield speedups for both the backtracking and filtering components of CP. The initial idea to apply quantum algorithms to the filtering subproblem was due to Kyle Booth, and the details were worked out collectively by all of the authors; I showed how quantum algorithms can be use for the backtracking component, including the introduction of the "chunky" search that interpolates between classical and quantum backtracking.

In general, CP is NP-complete and thus an exponential number of nodes must be searched before finding the first satisfying solution. Nevertheless, a wealth of heuristics and propagation functions have been developed that often allow the solution of practical instances in various applications.

A set of domains is said to be *domain consistent* (with respect to given constraints) if for every value in the domain of each variable, the other variables can be assigned values from their domains such that all constraints are satisfied. There are also weaker forms of consistency, such as range consistency, that in general prune less of the search tree but are computationally easier to reach. Our work focuses on the ALLDIFFERENT constraint, for which domain consistency can be captured by the existence of a maximum matching in a particular bipartite graph. The two vertex sets of the graph are the CP variables and the union of those variables' domains. For each variable and value in that variable's domain, there is an edge between the corresponding variable vertex and value vertex. An assignment of values to the variables is represented by a set of edges (each connecting a variable vertex to its corresponding value vertex) in this graph, and it satisfies the ALLDIFFERENT constraint if and only if it is a matching (set of disjoint edges) that covers all of the variable vertices. Domain consistency for the ALLDIFFERENT constraint can thus be achieved in the following way. First, the size of the maximum matching is determined; if it less than the number of variables, then the constraint is unsatisfiable. Then, if there is a matching of size equal to the number of variables, the *maximally-matchable edges* (those that are included in at least one maximum-cardinality matching) are identified and retained. In Ref. 29, we show how existing quantum algorithms [3, 92, 119] can be used to achieve domain consistency for ALLDIFFERENT (and other similar global constraints), by "quantizing" a classical algorithm due to Régin[135].

In addition to speeding up the filtering, which happens at every node, quantum algorithms can also be used to achieve a quadratic speedup with respect to the number of search nodes. Specifically, given a classical algorithm that explores T nodes, there is a quantum algorithm that performs the same backtracking in time $\tilde{O}(\sqrt{T})$. Furthermore, for any χ, quantum search can be performed over χ-sized chunks of the tree in total time $\tilde{O}(T/\sqrt{\chi})$; for constant-sized chunks ($\chi = O(1)$), this yields the classical $\tilde{O}(T)$ time, and for a single chunk ($\chi = T$), it recovers the quantum $\tilde{O}(\sqrt{T})$. There are several subtleties involved, which are addressed in more detail in Ref. 29.

5.2.2 Circuit routing as constraint programming

This section summarizes Ref. 151. In the previous subsection, we described the application of quantum computing to CP. Here, we describe inverse: the application of automated reasoning (including both CP and other approaches, including temporal planning) to the compilation of quantum circuits. Earlier, in Chapter 4, we described an instance-independent approach to the compilation of quantum circuits. By exploiting known structure in a family of instances, we were able to improve the efficiency of compilation relative to more general constructions and with essentially no per-instance overhead. However, many families of compilation instances have *instance-specific* structure that can be exploited, with accompanying per-instance optimization costs. If the per-instance cost is small enough, this can be a scalable approach. Even when the per-instance cost is high, such optimization may be worth it in the short term in order to maximize the efficacy of near-term devices. Ref. 151 introduces an automated reasoning framework for performing such optimization. The quantum circuit is first translated into a directed acyclic graph (DAG), whose nodes correspond to gates and whose (directed) edges indicate a precedence constraint. The goal is to find a dynamic mapping of logical to physical qubits together with a sequence of logical actions (corresponding to the logical gates) and SWAP actions (which update the mapping one pair of qubits at a time) such that the precedence relationships are satisfied. Such a model can be implemented in a standard language such as the Planning Domain Definition Language (PDDL), and then input into general-purpose automated reasoning software (e.g., CP solvers or temporal planners). This allows quantum circuit compilation to easily benefit from the significant development that has gone into such general-purpose solvers. Preliminary work has shown some successes of this approach, but only for relatively small instances. Whether or not this approach can be made to scale remains an open question.

Table 5.1: Parameterized ensembles of SMS instances explored numerically. "CC" indicates complexity class and "NPC" indicates NP-complete.

P	W	CC	Fixed parameters	Order parameter(s)	Empirical hardness
$\{p\}$	$\{w\}$	P	$T = np$, $p \in \{2, 7\}$	w	easy
$\{p_\mathrm{s}, p_\mathrm{l}\}$	$\{w\}$	P	$T = np_\mathrm{l}$, $P = \{2, 3\}$	w	easy
$\{p_\mathrm{s}, p_\mathrm{l}\}$	$\{w, T\}$	P	$T = np_\mathrm{l}$, $P = \{2, 3\}$, $w = 4$	$\lvert\{i \vert w_i = w\}\rvert / n$	easy
$\{p_\mathrm{s}, p_\mathrm{l}\}$	$[1, w_\mathrm{max}]$	NPC	$P \in \{\{3, 19\}, \{7, 11\}, \{3, 11\}\}$	w_max, T	typically easy

5.3 Phase transitions in single-machine scheduling

This section is a slight modification of Ref. 155.

5.3.1 Introduction

Single-machine scheduling (SMS), in which a set of jobs with release times, deadlines, and processing times are to be scheduled on a single machine, forms the backbone of many practical applications such as telescope scheduling, satellite scheduling, and manufacturing. A phase transition is a sudden change in a global property of a family with respect to an order parameter. For a number of NP-complete problems, the transition from solvable to unsolvable problems as a function of the number and tightness of constraints is of interest. On average, instances near this phase transition are typically observed to be exponentially harder than those that are not, and this clustering of hard instances near the phase transition becomes more concentrated as problem size increases.

In contrast to the many problems for which a phase transition has been found, to date the existence of phase transitions in SMS remains unexplored. We explore solvable-unsolvable phase transitions in the decision version of the SMS problem. We construct a general model for parameterized families of SMS instances and identify order parameters for a variety of specific families.

Our main findings are: 1) empirical evidence of a rapid transition in solvability (phase transition) for SMS, where none was previously characterized; 2) evidence of this transition in both provably tractable and provably intractable SMS ensembles; and 3) in the NP-complete ensemble (2 processing times), a transition characterized by two parameters, which is novel for phase transitions.

Where phase transitions have been identified in hard problems, most published results have shown "easy-hard-easy" behavior accompanying the phase transition, with the notable exception of Hamiltonicity. Our empirical results for the intractable SMS family show that there are few hard instances at the phase transition. While our results don't rule out an easy-hard-easy pattern in the NP-complete SMS family, they invite further questions into the relationship between hardness and phase transitions, and demonstrate the challenge of generating hard families of instances.

5.3.2 Related Work

In their pioneering work [60], Erdős and Rényi identified phase transitions of graph properties in their eponymous model. Ref. 37 later exhibited a connection between phase transitions and the location of hard instances in a handful of NP-complete problems (Hamiltonian cycle, graph coloring, k-SAT, and TSP). Since then, phase transitions have been found in a number of other combinatorial optimization problems such as independent set [71], number partitioning [30, 70, 117], and constraint satisfaction [134, 148]. The characterization of phase transitions in Cheeseman et al.'s original examples has later been refined (e.g. Hamiltonian cycle [150], graph coloring [2], k-SAT [97, 118], and travelling salesman problem [72]). Specifically, the typical solution time of instances at the phase transition grows exponentially with the problem size, and instances away from the transition are typically easy.

There is also evidence that in some problems the hardest instances suddenly emerge at some critical threshold below the solvability one [89]. The concentration of hard instances near the phase transition is of practical interest because it enables the generation of hard instances for benchmarking algorithms and solvers [90, 137].

The occurence of phase transitions is not limited to hard problems; they occur in provably easy problems as well, including graph properties such as connectivity [60], 2-SAT [44, 73], XOR-SAT [49] and HornSAT [120]. For hard problems, the existence of a solvability threshold does not necessarily imply an easy-hard-easy pattern, though counterexamples are rare. One such counterexample is the phase transition of Hamiltonicity in the Erdős-Rényi model [150]. Our work introduces the first analysis of phase transitions in ensembles of SMS instances in the literature.

5.3.3 Preliminaries

5.3.3.1 Single-Machine Scheduling

In this section, the most general problem we consider is the decision version of (non-preemptive) single-machine scheduling with (integer-valued) release times, deadlines, and processing times (i.e., $1|r_j d_j|U_{\max}$), which we henceforth refer to simply as SMS. An instance of SMS consists of a set of n jobs; each job j has a release time, deadline, and processing time ($r_j, d_j, p_j \in \mathbb{Z}_0^+$, respectively). The problem is to decide whether or not there exists a *schedule* $\boldsymbol{\sigma} \in \left(\mathbb{Z}_0^+\right)$ on a single machine, where σ_j indicates the starting time of job j, such that

- every job j starts no sooner than its release time, $r_j \leq \sigma_j$;
- every job j finishes by its deadline, $\sigma_j + p_j \leq d_j$; and
- no two jobs i and j overlap, $\sigma_i + p_i \leq \sigma_j$ or $\sigma_j + p_j \leq \sigma_i$.

We refer to the difference between the release time and the deadline for each job as that job's *window* $w_j = d_j - r_j$. More concisely, we write an instance as a tuple $(\mathbf{r}, \mathbf{w}, \mathbf{p})$. The *horizon* T is a time no earlier than the latest deadline, often exactly so.

In general, SMS is NP-complete [132]; this can be shown, e.g., by reduction from bin packing. However, there are numerous complexity results on refinements of the SMS problem. For example, with unit processing time, the greedy earliest-deadline algorithm suffices [58, 143]. More generally, a valid schedule can be found in quasi-linear time when the processing times are identical [69, 146] and in polynomial time when the processing times are restricted to be either one or some arbitrary but fixed constant [143]. When the processing times are restricted to two fixed constants greater than one, SMS remains NP-complete [59]. Alternatively, we show (in Section 5.3.6) that the greedy earliest-deadline algorithm suffices when the window lengths are either identical or restricted to two constants that differ by one.

5.3.3.2 Parameterized Ensembles

In order to find phase transitions in SMS, we construct suitable ensembles of instances parameterized by the horizon T and the sets of possible processing times P and window lengths W, i.e., we define a probability distribution $\Pr[\mathbf{r}, \mathbf{w}, \mathbf{p} | P, W, T]$ over instances for each set of parameter values (P, W, T). In our ensembles,

- the probability distribution of every job's properties (r_j, w_j, d_j) is identical and independent,
- for each job j the probability of p_j is independent of r_j and w_j,
- the probability of r_j is independent of all but w_j and T, and
- the probabilities for each job property are as uniform as possible given the parameterization and independence assumptions above.

Table 5.1 summarizes the special cases of this model that we focus on in this work. The left side shows the general parameterized ensemble and its complexity, and the right side shows the specific numeric values that we used and the resulting empirical hardness.

5.3.4 Experimental Methods

5.3.4.1 Solution of SMS instances

We deployed a variety of solvers by mapping SMS to different canonical problems: Mixed Integer Linear Programming (MILP), Satisfiability (SAT), and Constraint Programming (CP). We found that IBM ILOG's CP solver, CP Optimizer, significantly outperforms the alternatives we tried, both mapping to MILP or SAT and using other CP solvers. This is to be expected, given that the CP model more naturally captures the structure of SMS, whereas this structure is lost in the mappings to both MILP (because it requires ancillary variables to account for the disjunction in the overlap constraint) and to SAT (because integer variables must be encoded using Boolean variables). Besides their differing running times, all the solvers we tried gave consistent answers with respect to solvability when run on the

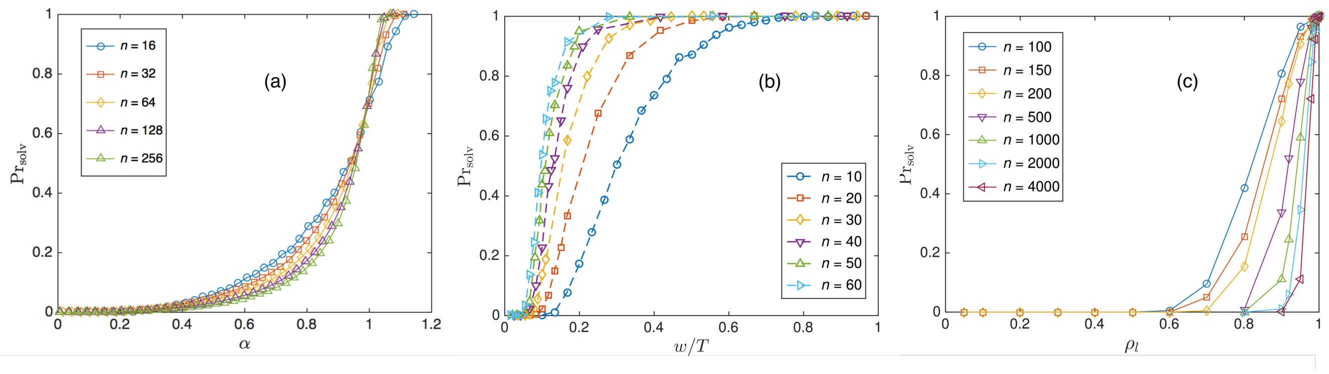

Figure 5.1: Phase transitions in problem Families 1 to 3. (a) Family 1. Exactly calculated solvability probability $\mathrm{Pr}_{\mathrm{solv}}$ versus normalized window length $\alpha = w/n(p - 1/2)$ for $P = \{\log_2 n\}$, $W = \{\alpha n(p - 1/2)\}$, $T = np$. See Section 5.3.4.2 for details for the calculation. (b) Family 2. $\mathrm{Pr}_{\mathrm{solv}}$ versus normalized window length w/T for $P = \{2, 3\}$, $T = 3n$, and various n. (c) Family 3. $\mathrm{Pr}_{\mathrm{solv}}$ versus fraction of jobs with longer window length for $P = \{2, 3\}$, $W = \{4, T\}$, $T = 3n$, with the restriction that exactly half of the jobs have each processing time. Each probability is calculated using 200 random instances.

same instances. CP optimizer is an exact solver, but we allowed a maximum running time of twenty-four hours. Only the ensemble with two processing times and a range of window lengths (Section 5.3.5) contained instances not solved within this twenty-four-hour window, and very few at that (fewer than one percent for every set of parameter values); these unsolved instances are thus statistically negligible for our purposes.

For the presentation of the computational time, we present only the results of CP optimizer. The computation was performed on the Ivy Bridge nodes of NASA's Pleiades supercomputer. To examine the computational effort across the transition, we use either the elapsed wall clock time or the number of nodes explored in the constraint programming search as a measure of the computational cost; because we ran all experiments on identical hardware and the time spent at each node is approximately constant, these are roughly proportional to each other.

5.3.4.2 Calculation of solvability probability for Family 1

In the simplest case of a single processing time and single window length, $P = \{p\}$ and $W = \{w\}$, we calculate exactly the solvability probability. The calculation is recursive, adding jobs one by one. That exact calculations by this method match estimates from sampling confirm the correctness of both our algorithm and implementation thereof.

The solvability probabilities for various p and w when $n = 100$ are shown in Fig. 5.2. The plot looks essentially the same for other values of n, with discretization effects at smaller n; the data is well converged by $n = 100$. That the slope is greatest at $\alpha = 1$ for a wide range of p indicates that the choice of α appropriately scales the parameters. Because the sharpness

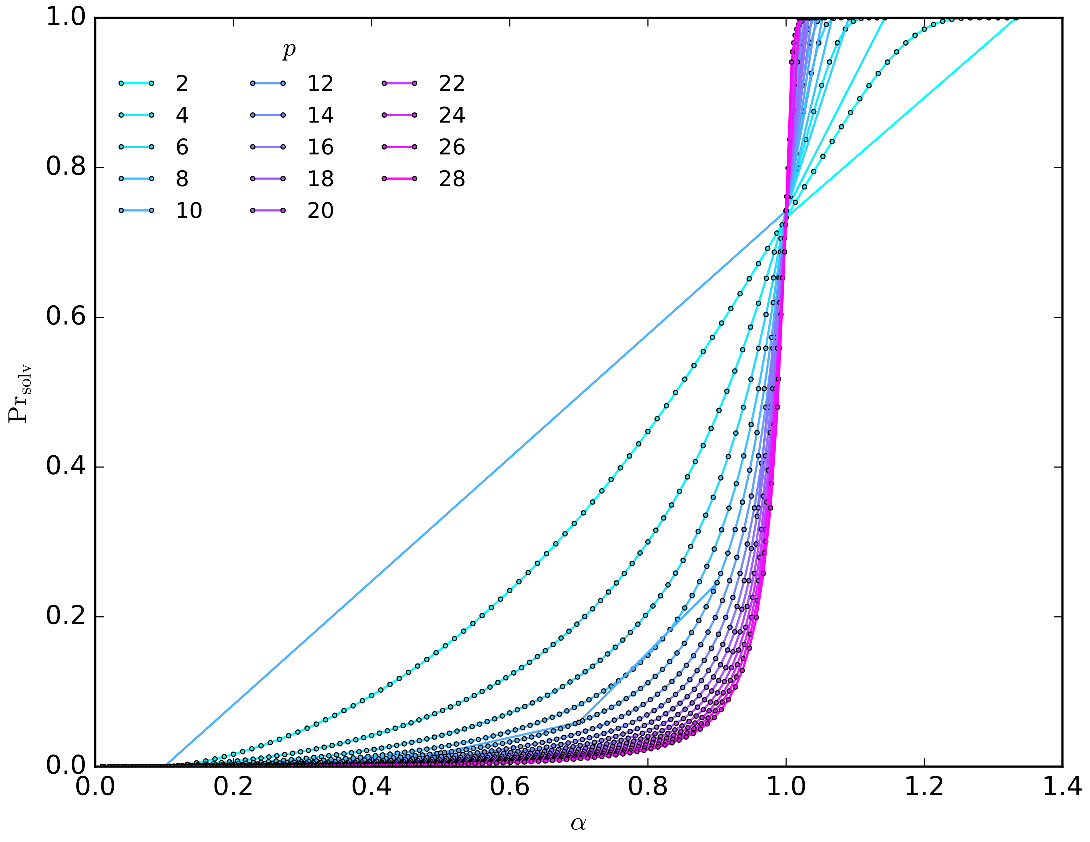

Figure 5.2: Solvability probability vs. order parameter $\alpha = w/n(p - 1/2)$ for Family 1, with $n = 100$.

of the transition is fixed for a given p but increases therewith, we chose $p = \log_n$ so that the sharpness increases with n. This logarithmic scaling was chosen to minimize computation time, but setting p to any strictly increasing function of n suffices to sharpen the transition.

5.3.5 Phase Transitions

In this section, we present experimental results on solvable-unsolvable phase transitions for several SMS families. The order parameter of such transitions varies the constrainedness of the instances using the time horizon, window size, or both. When the instance is barely constrained (e.g. large horizon, large windows), it is likely to be solvable, and when it is heavily constrained, it is likely to be unsolvable; at both extremes, the solvability of an instance is easy to determine.

Family 1: Single processing time, single window length In the simplest case, we have a single processing time and a single window size, $P = \{p\}$ and $W = \{w\}$, and set the horizon to the minimal non-trivial value $T = np$. Problem instances are generated so that for each job i, r_i is uniformly sampled from all possible times $[0,\ T - w]$. The solvability probabilities for $p = \log_2 n$ and $w = \alpha n(p - 1/2)$ are shown in Fig. 5.1 (a) for various α. There is a transition from a high probability of unsolvability to a high probability of solvability at $\alpha = 1$ that sharpens with the number of tasks n, which is strong empirical evidence of a phase transition.

The probability of solvability decreases for the same value of w/T when p increases. One way to understand this drop is to think of fixing the number of jobs, and scale w and T with p. A rise in the processing time greatly increases the number of ways in which unsolvable instances are created.

Family 2: Two processing times, single window length Next, we allow for two processing times, $P = \{p_\mathrm{s}, p_\mathrm{l}\}$, while keeping the single window length, $W = \{w\}$ and setting the horizon to be $T = np_\mathrm{l}$. The solvability probabilities are shown in Fig. 5.1 (b). Again the probability varies monotonically from zero to one with the window length. Using the normalized window length w/T as the order parameter, the transition sharpens with increasing n.

Family 3: Two processing times, two window lengths Next, we allow for two processing times $P = \{p_\mathrm{s}, p_\mathrm{l}\}$ and two window lengths $W = \{w_\mathrm{s}, w_\mathrm{l}\}$. The processing times are independent parameters, and the horizon is set to be $T = np_\mathrm{l}$. We choose the small window length $w_\mathrm{s} = p_\mathrm{l} + 1$ just big enough to have some wiggle room regardless of the processing time and the large one $w_\mathrm{l} = T$ equal to the horizon. Note that unlike the previous cases when the window size change leads to tighter or looser constraints, we now need a parameter reflecting the number of loose constraints in the problem instead. The fraction of jobs with the larger window length, $\rho_l \equiv |\{j|w_j = T\}|/n$, is thus used as the order parameter.

For $P = \{2, 3\}$, the solvability probabilities as a function of the order parameter are shown in Fig. 5.1 (c). When most jobs have the shorter window length, there is little flexibility and the probability of an instance being solvable is low; when most jobs have the window length equal to the horizon, jobs can be assigned almost freely and an instance is very likely to be solvable.

Family 4: Two processing times, range of window lengths This is the most general problem we consider, with two processing times, $P = \{p_\mathrm{s}, p_\mathrm{l}\}$ and a range of window times $W = [p_\mathrm{l} + 1, w_\mathrm{max}]$. We fix $p_\mathrm{s}, p_\mathrm{l}$, and study the sets of instances parameterized by the tuple (T, w_max). For each value of the tuple, 100 to 1000 instances are drawn. For each job, p_i and w_i are uniformly sampled from P and W, respectively. The release time r_i is then uniformly sampled from $[0,\ T - w_i]$.

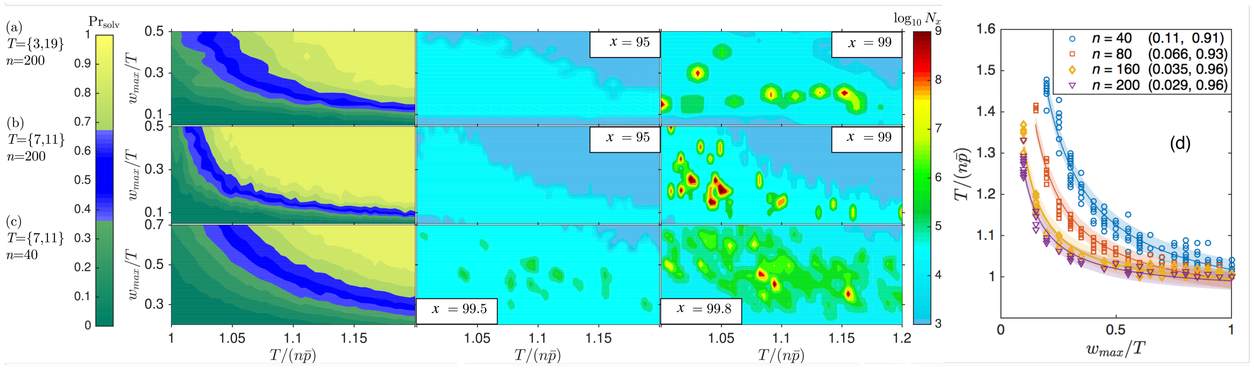

Figure 5.3: Numerical data for Family 4. Left panel: The first column of panels shows the phase transition. Contour plot of solvability probability versus normalized horizon and normalized window length. The horizon is normalized by the average total processing time $n\bar{p}$, and the window length is normalized by the (not normalized) horizon. The dark blue region is the phase transition frontier. The second and third columns show the contour plot of hardness where color encodes $\log_{10}(N_x)$, where N_x is the x-percentile of the number of nodes for each parameter tuple $(T, w_{\max})$. (a) $n = 200$, $P = \{3,\ 19\}$ from left to right: Pr_{solv}, $\log_{10}(N_{95})$, $\log_{10}(N_{99})$. For each parameter tuple, 100 instances are drawn. (b) $n = 200$, $P = \{7,\ 11\}$, from left to right: Pr_{solv}, $\log_{10}(N_{95})$, $\log_{10}(N_{99})$. For each parameter tuple, 100 instances are drawn. (c) $n = 40$, $P = \{7,\ 11\}$ from left to right: Pr_{solv}, $\log_{10}(N_{99.5})$, $\log_{10}(N_{99.8})$. For each parameter tuple, 1000 instances are drawn. Right panel: (d) $\text{Pr}_{\text{solv}} = 0.5 \pm 0.05$ for $P = \{3, 19\}$. The transition frontier is fit as $\frac{T}{np} = \frac{a_0}{w/T} + c_0$. The shaded area shows 95 percent confidence interval. The fitted coefficients (a_0, c_0) are in the legend for each problem size.

We choose the problem family of parameters $P = \{3,\ 19\}$, $\{7,\ 11\}$, and $\{3,\ 11\}$, and problem size n ranging from 16 to 200.

In Fig. 5.3 the first column of subplots show the probability of solvability in the two-dimensional parameter space $(T, w_{\max})$, for different parameter choices and problem sizes in (a) to (c). On each subplot, with a normalization of the parameters as $T/(n\bar{p})$ and $w_{\max}/T$, a clear transition is seen. On the bottom left of the scanning area (where the time horizon is close to the sum of the processing times, and the jobs have very little flexibility), few solution exists. On the top right of the area, i.e., the horizon is ample to fit all jobs loosely, and the windows apply almost no constraint to the schedule and there are many solutions to the problem. This transition gets steeper as the problem size increases, as reflected by the reduction in width of the “dark (blue) band" in the middle of the plot (c) versus plot (b). The transition between these two regimes is a simple curve that is a function of T and $w_{\max}$. The shape of the curves also suggests that the transition region is converging as n increases.

We take a closer look in Fig. 5.3 (d) where for $P = \{3,\ 19\}$, numerical data with

$\mathrm{Pr}_{\mathrm{solv}} = 0.5 \pm 0.05$ is singled out as the transition frontier, and fit as

$$\frac{T}{np} = \frac{a_0}{w/T} + c_0 \tag{5.5}$$

As n increases, the fit parameters a_0 decreases, and c_0 approaches 1.

A similar pattern in the phase transition is observed for the other two sets of processing times $P = \{7,\ 11\}$ and $\{3,\ 11\}$.

5.3.6 Hardness: empirical and provable

When there is a single window length, $W = \{w\}$, or two window lengths that differ by one, $W = \{w, w+1\}$, the greedy earliest-deadline algorithm suffices for any set of processing times P.

When the sets of processing times and window lengths are fixed (or the set of window lengths is some fixed set and the horizon), SMS is easy. We explored the specific case with $P = \{p_{\mathrm{s}}, p_{\mathrm{l}}\}$, $W = \{p_{\mathrm{l}}+1, T\}$, and $T = np_{\mathrm{l}}$ (Family 3); as expected we observe solution times that are polynomial in n. With two non-unit processing times and arbitrary window lengths (that may grow with n), SMS is NP-complete. The ensembles we investigate in Family 4 where $P = \{p_{\mathrm{s}}, p_{\mathrm{l}}\}$, $W = [1, w_{\max}]$, and $T = np_{\mathrm{l}}$ should therefore contain some hard problems. Nevertheless, we find that such instances are typically easy.

To study the hardness of this family, 50-th to 99.8-th percentiles of the number of nodes explored in the attempt to solve the problem instances are taken as a measure.

Typical instances are easy. We first examine the median effort to solve problems. We do not observe a clear correlation between the locations of relative hard instances and the regions of the phase transition. Also, no evidence of an exponential increase in the median hardness with problem size is observed. This indicates that the median case for such generated problems are probably easy. The same phenomenon appears in higher percentiles, we observed no hard instances up to 95-percentiles; see the results for 95-th percentiles in Figure. 5.3 (a), and (b).

High percentiles: rare hard instances, weak correlation with the phase transition. We then look at even higher percentiles of the number of nodes explored. As shown in Figure. 5.3 (a), (b) and (c), relatively hard instances show up at these very high percentiles (e.g., 99-percentile for $n = 200$, 99.5- and 99.8-percentiles for $n = 40$). High percentiles of the nodes explored is plotted in logarithmic scale for parameters $P = \{3,\ 19\}$ and $P = \{7,\ 11\}$, and the phase transitions are shown in parallel for comparison. Hard instances scatter around the phase transition region, suggesting weak correlation. However, due to the rareness of the hard instances and large statistical fluctuations in the high percentiles, we cannot confirm a correlation between the phase transition and the hardness in high percentiles.

To compare hardness for different problem sizes, due to the dispersion of hard instances in the parameter space, a reliable fitting for extracting the number of nodes of certain percentiles is missing and the results are thus tempered by large statistical fluctuations. Furthermore, for larger problem sizes, results of higher percentile is also limited by the 24-hour timeout of the solver. Therefore, we do not draw conclusions on the empirical time complexity for such high percentiles.

SMS with identical window lengths or two window lengths that differ by one is easy This problem can be solved in polynomial time by the following algorithm. First, order the jobs by their release date in non-decreasing order, breaking ties with the deadlines. Then, schedule each job in order by setting its start time to the maximum of its release time and the completion of the previous job. We prove below that the resulting schedule is feasible if and only if there is a feasible schedule for the instance by showing that if there is a feasible schedule for the instance with a pair of immediately subsequent jobs in the schedule that are out of order with respect to their release times, then there is feasible schedule in which they are in order. Iterating this logic, it follows that if there is a feasible schedule, then there is a feasible schedule in which all the jobs are in order.

Formally, suppose there is a schedule $\boldsymbol{\sigma}$ with a pair of jobs i and j such that $\sigma_j < \sigma_i$ and there is no job k such that $\sigma_j < \sigma_k < \sigma_i$, but $r_i \leq r_j$ and $d_i \leq d_j$. Consider the schedule $\boldsymbol{\sigma}'$ that is the same as $\boldsymbol{\sigma}$ except that $\sigma'_i = \sigma_j$ and $\sigma'_j = \sigma_i + p_i - p_j$. In both schedules, the first job starts at $\sigma_j = \sigma'$ and the second job ends at $\sigma_i + p_i = \sigma'_j + p_j$, so no conflict with other jobs can be introduced. If $\boldsymbol{\sigma}$ satisfies the release, deadline, and overlap constraints for jobs i and j,

$$\begin{aligned} r_i \leq \sigma_i &\leq d_i - p_i, \\ r_j \leq \sigma_j &\leq d_j - p_j, \\ \sigma_j + p_j &\leq \sigma_i, \end{aligned} \tag{5.6}$$

then so does $\boldsymbol{\sigma}'$,

$$\begin{aligned} \sigma'_i &= \sigma_j \geq r_j \geq r_i, \\ \sigma'_i &= \sigma_j < \sigma_i \leq d_i - p_i, \\ \sigma'_j &= \sigma_i + p_i - p_j \geq \sigma_i - p_j \geq \sigma_j \geq r_j, \\ \sigma'_j + p_j &= \sigma_i + p_i \leq d_i \leq d_j, \\ \sigma'_i + p_i &= \sigma_j + p_i \leq \sigma_i - p_j + p_i = \sigma'_j. \end{aligned} \tag{5.7}$$

SMS with a fixed set of processing times and a set of window lengths that is the union of a fixed set and the horizon is easy Here we show that SMS is easy when the processing times are restricted to some fixed set P and the window lengths to $W = W_0 \cup \{T\}$, where W_0 is some fixed set, by sketching an algorithm with running time $\mathcal{O}\left(|P||W|n^{|P||W|}\right)$.

First, we note if there is a feasible schedule $\boldsymbol{\sigma}$, then there is another feasible schedule $\boldsymbol{\sigma}'$ with the same completion time such that for every pair of jobs $\{i, j\}$ if $p_i = p_j$ and $w_i = w_j$ then $\sigma'_i < \sigma'_j$ implies that $r_i \leq r_j$. Suppose that in valid schedule $\boldsymbol{\sigma}$ there are two jobs $\{i, j\}$ with $p_i = p_j$, $w_i = w_j$, $\sigma_i < \sigma_j$, and $r_i > r_j$. Because $\boldsymbol{\sigma}$ is a valid schedule,

$$r_j < r_i \leq \sigma_i < \sigma_j \leq d_j - p_j < d_i - p_j, \tag{5.8}$$

so too is the schedule $\boldsymbol{\sigma}'$ formed by setting $\sigma'_i = \sigma_j$, $\sigma'_j = \sigma_i$, and $\sigma'_k = \sigma_k$ for $k \notin \{i, j\}$. Because they have identical processing times and window sizes, no new conflicts can be introduced by this switch, either with other jobs or between the two. That is, if the instance is solvable, then there is always a valid schedule in which all jobs with the same processing time and the same window length start in order of the release times.

Now, consider the standard dynamic programming approach. For each subset of jobs $J \subset [n]$, let $m(J)$ be the earliest completion time of a valid partial schedule for the jobs J, or infinity if there is none. Thus the instance is solvable if and only if $m([n])$ is finite. In general, we can calculate $m(J)$ recursively as

$$m(\emptyset) = 0, \tag{5.9}$$

$$m(J) = \min_{j \in J} \left\{ \tilde{m}_j(J \setminus \{j\}) + p_j \right\}, \tag{5.10}$$

$$\tilde{m}_j(J) = \begin{cases} r_j, & m(J) \leq r_j, \\ m(J), & r_j < m(J) \leq d_j - p_j, \\ \infty, & m(J) > d_j - p_j, \end{cases} \tag{5.11}$$

because any valid schedule for jobs J is the concatenation of schedules for $J \setminus \{j\}$ and $\{j\}$ for some $j \in J$. In the present case, we know that without loss of generality we can assume that the last job in the schedule has a release time at least that of all the other jobs with the same processing time and window length:

$$m(J) = \min_{\substack{p \in P \\ w \in W}} \left\{ \tilde{m}_j(J \setminus \{j\}) \Big|_{j = \left(\underset{j' \in J \big|_{\substack{p_{j'} = p \\ w_{j'} = w}}}{\arg\max} \{r_{j'}\} \right)} + p \right\}. \tag{5.12}$$

Let $i_{p,w,k}$ be the kth job with processing time $p_j = p$ and $w_j = w$ when the jobs with the same processing times and window lengths are ordered by release times. Then we can parameterize the sets of jobs J that we encounter during recursion by a $|P| \times |W|$ matrix C whose entries indicate how many of each type of job have been scheduled, i.e. $C_{p,w}$ indicates how many jobs with processing time p and window length w have been scheduled; within the jobs of a given type, the ones with the earliest release times are scheduled first:

$$J(C) = \bigcup_{\substack{p \in P \\ w \in W}} \left\{ i_{p,w,k} | k \in [C_{p,w}] \right\}, \tag{5.13}$$

so that we can write

$$m\left(J(C)\right) = \min_{\substack{p \in P \\ w \in W}} \left\{ \tilde{m}_{i_{p,w,C_{p,w}}} \left(J(C) - \{i_{p,w,C_{p,w}}\} \right) + p \right\}, \tag{5.14}$$

Ultimately, we want to find $m([n])$. To calculate $m(C)$, we need to consider $m(C')$ for $|P||W|$ other schedules C'. But each $C_{p,w}$ is bounded by n, so the maximum number of values of m that we need to calculate is $n^{|P||W|}$. Therefore, the algorithm runs in $\mathcal{O}\left(|P||W|n^{|P||W|}\right)$ time.

SMS with two non-unit processing times is NP-complete See Ref. 59.

Bibliography

[1] Scott Aaronson and Lijie Chen. "Complexity-Theoretic Foundations of Quantum Supremacy Experiments". In: *arXiv:1612.05903 [quant-ph]* (Dec. 2016). arXiv: 1612.05903 [quant-ph].

[2] Dimitris Achlioptas and Ehud Friedgut. "A Sharp Threshold for K-Colorability". en. In: *Random Structures & Algorithms* 14.1 (1999), pp. 63–70. ISSN: 1098-2418. DOI: 10.1002/(SICI)1098-2418(1999010)14:1<63::AID-RSA3>3.0.CO;2-7.

[3] Andris Ambainis and Martins Kokainis. "Quantum Algorithm for Tree Size Estimation, with Applications to Backtracking and 2-Player Games". en. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. Montreal Canada: ACM, June 2017, pp. 989–1002. ISBN: 978-1-4503-4528-6. DOI: 10.1145/3055399.3055444.

[4] Omer Angel and Igor Shinkar. "A Tight Upper Bound on Acquaintance Time of Graphs". en. In: *Graphs and Combinatorics* 32.5 (Sept. 2016), pp. 1667–1673. ISSN: 1435-5914. DOI: 10.1007/s00373-016-1700-4.

[5] Tom M. Apostol. *Calculus*. Second. Vol. 1. Wiley, 1991. ISBN: 978-0-471-00005-1.

[6] Itai Arad and Zeph Landau. "Quantum Computation and the Evaluation of Tensor Networks". In: *SIAM Journal on Computing* 39.7 (June 2010), pp. 3089–3121. ISSN: 0097-5397. DOI: 10.1137/080739379.

[7] Stefan Arnborg and Andrzej Proskurowski. "Linear Time Algorithms for NP-Hard Problems Restricted to Partial k-Trees". en. In: *Discrete Applied Mathematics* 23.1 (Apr. 1989), pp. 11–24. ISSN: 0166-218X. DOI: 10.1016/0166-218X(89)90031-0.

[8] Frank Arute et al. "Quantum Supremacy Using a Programmable Superconducting Processor". en. In: *Nature* 574.7779 (Oct. 2019), pp. 505–510. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1666-5.

[9] Neil W. Ashcroft and N. David Mermin. *Solid State Physics*. 1976. ISBN: 978-0-03-083993-1.

[10] Alán Aspuru-Guzik et al. "Simulated Quantum Computation of Molecular Energies". en. In: *Science* 309.5741 (Sept. 2005), pp. 1704–1707. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1113479.

[11] Ryan Babbush et al. "Low-Depth Quantum Simulation of Materials". In: *Physical Review X* 8.1 (Mar. 2018), p. 011044. DOI: 10.1103/PhysRevX.8.011044.

[12] Boaz Barak et al. "Beating the Random Assignment on Constraint Satisfaction Problems of Bounded Degree". In: *arXiv:1505.03424 [cs]* (Aug. 2015). arXiv: `1505.03424 [cs]`.

[13] Robert Beals et al. "Efficient Distributed Quantum Computing". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 469.2153 (May 2013), p. 20120686. DOI: `10.1098/rspa.2012.0686`.

[14] Itai Benjamini, Igor Shinkar, and Gilad Tsur. "Acquaintance Time of a Graph". In: *SIAM Journal on Discrete Mathematics* 28.2 (Jan. 2014), pp. 767–785. ISSN: 0895-4801. DOI: `10.1137/130930078`.

[15] Debjyoti Bhattacharjee and Anupam Chattopadhyay. "Depth-Optimal Quantum Circuit Placement for Arbitrary Topologies". In: *arXiv:1703.08540 [quant-ph]* (Mar. 2017). arXiv: `1703.08540 [quant-ph]`.

[16] Jacob Biamonte and Ville Bergholm. "Tensor Networks in a Nutshell". In: *arXiv:1708.00006 [cond-mat, physics:gr-qc, physics:hep-th, physics:math-ph, physics:quant-ph]* (July 2017). arXiv: `1708.00006 [cond-mat, physics:gr-qc, physics:hep-th, physics:math-ph, physics:quant-ph]`.

[17] Jacob D. Biamonte, Jason Morton, and Jacob Turner. "Tensor Network Contractions for #SAT". en. In: *Journal of Statistical Physics* 160.5 (Sept. 2015), pp. 1389–1404. ISSN: 1572-9613. DOI: `10.1007/s10955-015-1276-z`.

[18] Dan Bienstock. "On Embedding Graphs in Trees". en. In: *Journal of Combinatorial Theory, Series B* 49.1 (June 1990), pp. 103–136. ISSN: 0095-8956. DOI: `10.1016/0095-8956(90)90066-9`.

[19] Rupak Biswas et al. "A NASA Perspective on Quantum Computing: Opportunities and Challenges". en. In: *Parallel Computing* 64 (May 2017), pp. 81–98. ISSN: 01678191. DOI: `10.1016/j.parco.2016.11.002`.

[20] Lennart Bittel and Martin Kliesch. "Training Variational Quantum Algorithms Is NP-Hard – Even for Logarithmically Many Qubits and Free Fermionic Systems". en. In: (Jan. 2021).

[21] Hans Bodlaender, Michael R. Fellows, and Dimitrios M. Thilikos. "Derivation of Algorithms for Cutwidth and Related Graph Layout Parameters". en. In: *Journal of Computer and System Sciences* 75.4 (June 2009), pp. 231–244. ISSN: 0022-0000. DOI: `10.1016/j.jcss.2008.10.003`.

[22] Hans L. Bodlaender. "A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth". In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '93. San Diego, California, USA: Association for Computing Machinery, June 1993, pp. 226–234. ISBN: 978-0-89791-591-5. DOI: `10.1145/167088.167161`.

[23] Hans L. Bodlaender and Dimitrios M. Thilikos. “Constructive Linear Time Algorithms for Branchwidth”. en. In: *Automata, Languages and Programming*. Ed. by Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1997, pp. 627–637. ISBN: 978-3-540-69194-5. DOI: `10.1007/3-540-63165-8_217`.

[24] Hans L. Bodlaender et al. “Approximating Treewidth, Pathwidth, and Minimum Elimination Tree Height”. en. In: *Graph-Theoretic Concepts in Computer Science*. Ed. by Gunther Schmidt and Rudolf Berghammer. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1992, pp. 1–12. ISBN: 978-3-540-46735-9. DOI: `10.1007/3-540-55121-2_1`.

[25] Hans L. Bodlaender et al. “Approximating Treewidth, Pathwidth, Frontsize, and Shortest Elimination Tree”. en. In: *Journal of Algorithms* 18.2 (Mar. 1995), pp. 238–255. ISSN: 0196-6774. DOI: `10.1006/jagm.1995.1009`.

[26] Sergio Boixo et al. “Characterizing Quantum Supremacy in Near-Term Devices”. en. In: *Nature Physics* 14.6 (June 2018), pp. 595–600. ISSN: 1745-2481. DOI: `10.1038/s41567-018-0124-x`.

[27] Sergio Boixo et al. “Simulation of Low-Depth Quantum Circuits as Complex Undirected Graphical Models”. In: *arXiv:1712.05384 [quant-ph]* (Jan. 2018). arXiv: `1712.05384 [quant-ph]`.

[28] Kyle E. C. Booth et al. “Comparing and Integrating Constraint Programming and Temporal Planning for Quantum Circuit Compilation”. en. In: *Twenty-Eighth International Conference on Automated Planning and Scheduling*. June 2018.

[29] Kyle E. C. Booth et al. “Quantum-Accelerated Constraint Programming”. In: *arXiv:2103.04502 [quant-ph]* (Mar. 2021). arXiv: `2103.04502 [quant-ph]`.

[30] Christian Borgs, Jennifer Chayes, and Boris Pittel. “Phase Transition and Finite-Size Scaling for the Integer Partitioning Problem”. en. In: *Random Structures and Algorithms* 19.3-4 (Oct. 2001), pp. 247–288. ISSN: 1042-9832, 1098-2418. DOI: `10.1002/rsa.10004`.

[31] Sergey Bravyi, David P. DiVincenzo, and Daniel Loss. “Schrieffer–Wolff Transformation for Quantum Many-Body Systems”. en. In: *Annals of Physics* 326.10 (Oct. 2011), pp. 2793–2826. ISSN: 0003-4916. DOI: `10.1016/j.aop.2011.06.004`.

[32] Sergey Bravyi et al. “Tapering off Qubits to Simulate Fermionic Hamiltonians”. In: *arXiv:1701.08213 [quant-ph]* (Jan. 2017). arXiv: `1701.08213 [quant-ph]`.

[33] Sergey B. Bravyi and Alexei Yu. Kitaev. “Fermionic Quantum Computation”. en. In: *Annals of Physics* 298.1 (May 2002), pp. 210–226. ISSN: 0003-4916. DOI: `10.1006/aphy.2002.6254`.

[34] Stephen Brierley. “Efficient Implementation of Quantum Circuits with Limited Qubit Interactions”. In: *Quantum Information & Computation* 17.13-14 (Nov. 2017), pp. 1096–1104. ISSN: 1533-7146.

[35] Anne Broadbent and Alex B. Grilo. “QMA-Hardness of Consistency of Local Density Matrices with Applications to Quantum Zero-Knowledge”. In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. Durham, NC, USA: IEEE, Nov. 2020, pp. 196–205. ISBN: 978-1-72819-621-3. DOI: 10.1109/FOCS46700.2020.00027.

[36] Jun Cai, William G. Macready, and Aidan Roy. “A Practical Heuristic for Finding Graph Minors”. In: *arXiv:1406.2741 [quant-ph]* (June 2014). arXiv: 1406.2741 [quant-ph].

[37] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. “Where the Really Hard Problems Are”. In: *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI’91. Sydney, New South Wales, Australia: Morgan Kaufmann Publishers Inc., Aug. 1991, pp. 331–337. ISBN: 978-1-55860-160-4.

[38] Jianxin Chen et al. “Classical Simulation of Intermediate-Size Quantum Circuits”. In: *arXiv:1805.01450 [quant-ph]* (May 2018). arXiv: 1805.01450 [quant-ph].

[39] Andrew M. Childs, David Gosset, and Zak Webb. “Complexity of the XY Antiferromagnet at Fixed Magnetization”. In: *Quantum Information and Computation* (Jan. 2016), pp. 1–18. ISSN: 15337146, 15337146. DOI: 10.26421/QIC16.1-2-1.

[40] Andrew M. Childs, David Gosset, and Zak Webb. “The Bose-Hubbard Model Is QMA-Complete”. EN. In: *Theory of Computing* 11.1 (Dec. 2015), pp. 491–603. ISSN: 1557-2862. DOI: 10.4086/toc.2015.v011a020.

[41] Andrew M. Childs, Aaron Ostrander, and Yuan Su. “Faster Quantum Simulation by Randomization”. en-GB. In: *Quantum* 3 (Sept. 2019), p. 182. DOI: 10.22331/q-2019-09-02-182.

[42] Andrew M. Childs, Eddie Schoute, and Cem M. Unsal. “Circuit Transformations for Quantum Architectures”. In: *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*. Ed. by Wim van Dam and Laura Mancinska. Vol. 135. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 3:1–3:24. ISBN: 978-3-95977-112-2. DOI: 10.4230/LIPIcs.TQC.2019.3.

[43] Vicky Choi. “Minor-Embedding in Adiabatic Quantum Computation: II. Minor-Universal Graph Design”. en. In: *Quantum Information Processing* 10.3 (June 2011), pp. 343–353. ISSN: 1573-1332. DOI: 10.1007/s11128-010-0200-3.

[44] V. Chvatal and B. Reed. “Mick Gets Some (the Odds Are on His Side) (Satisfiability)”. In: *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*. Pittsburgh, PA, USA: IEEE, 1992, pp. 620–627. ISBN: 978-0-8186-2900-6. DOI: 10.1109/SFCS.1992.267789.

[45] Stephen A. Cook. “The Complexity of Theorem-Proving Procedures”. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC ’71. Shaker Heights, Ohio, USA: Association for Computing Machinery, May 1971, pp. 151–158. ISBN: 978-1-4503-7464-4. DOI: `10.1145/800157.805047`.

[46] William Cook and Paul Seymour. “Tour Merging via Branch-Decomposition”. In: *INFORMS Journal on Computing* 15.3 (Aug. 2003), pp. 233–248. ISSN: 1091-9856. DOI: `10.1287/ijoc.15.3.233.16078`.

[47] Philippe Corboz and Guifré Vidal. “Fermionic Multiscale Entanglement Renormalization Ansatz”. In: *Physical Review B* 80.16 (Oct. 2009), p. 165129. DOI: `10.1103/PhysRevB.80.165129`.

[48] Alexander Cowtan et al. “On the Qubit Routing Problem”. In: *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*. Ed. by Wim van Dam and Laura Mancinska. Vol. 135. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 5:1–5:32. ISBN: 978-3-95977-112-2. DOI: `10.4230/LIPIcs.TQC.2019.5`.

[49] Nadia Creignou and Hervé Daude. “Satisfiability Threshold for Random XOR-CNF Formulas”. en. In: *Discrete Applied Mathematics* 96-97 (Oct. 1999), pp. 41–53. ISSN: 0166218X. DOI: `10.1016/S0166-218X(99)00032-3`.

[50] Gavin E. Crooks. “Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem”. In: *arXiv:1811.08419 [quant-ph]* (Nov. 2018). arXiv: `1811.08419 [quant-ph]`.

[51] Toby Cubitt and Ashley Montanaro. “Complexity Classification of Local Hamiltonian Problems”. In: *SIAM Journal on Computing* 45.2 (Jan. 2016), pp. 268–316. ISSN: 0097-5397. DOI: `10.1137/140998287`.

[52] Marek Cygan et al. “Lower Bounds Based on the Exponential-Time Hypothesis”. en. In: *Parameterized Algorithms*. Ed. by Marek Cygan et al. Cham: Springer International Publishing, 2015, pp. 467–521. ISBN: 978-3-319-21275-3. DOI: `10.1007/978-3-319-21275-3_14`.

[53] David Deutsch. “Quantum Theory, the Church–Turing Principle and the Universal Quantum Computer”. en. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (July 1985), pp. 97–117. ISSN: 0080-4630. DOI: `10.1098/rspa.1985.0070`.

[54] Reinhard Diestel. *Graph Theory*. en. Fifth. Graduate Texts in Mathematics. Berlin Heidelberg: Springer-Verlag, 2017. ISBN: 978-3-662-53621-6. DOI: `10.1007/978-3-662-53622-3`.

[55] P. A. M. Dirac. "Quantum Mechanics of Many-Electron Systems". en. In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 123.792 (Apr. 1929), pp. 714–733. ISSN: 0950-1207, 2053-9150. DOI: 10.1098/rspa.1929.0094.

[56] Eugene Dumitrescu. "Tree Tensor Network Approach to Simulating Shor's Algorithm". In: *Physical Review A* 96.6 (Dec. 2017), p. 062322. DOI: 10.1103/PhysRevA.96.062322.

[57] Eugene F. Dumitrescu et al. "Benchmarking Treewidth as a Practical Component of Tensor Network Simulations". en. In: *PLOS ONE* 13.12 (Dec. 2018). Ed. by Emanuele G. Dalla Torre, e0207827. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0207827.

[58] Christoph Dürr and Mathilde Hurand. "Finding Total Unimodularity in Optimization Problems Solved by Linear Programs". en. In: *Algorithmica* 59.2 (Feb. 2011), pp. 256–268. ISSN: 0178-4617, 1432-0541. DOI: 10.1007/s00453-009-9310-7.

[59] Jan Elffers and Mathijs de Weerdt. "Scheduling with Two Non-Unit Task Lengths Is NP-Complete". In: *arXiv:1412.3095 [cs]* (Oct. 2017). arXiv: 1412.3095 [cs].

[60] Paul Erdős and A Rényi. "On the Evolution of Random Graphs". In: *Publ. Math. Inst. Hungar. Acad. Sci* 5 (1960), pp. 17–61.

[61] E. Farhi et al. "Quantum Algorithms for Fixed Qubit Architectures". en. In: (Mar. 2017).

[62] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A Quantum Approximate Optimization Algorithm". In: *arXiv:1411.4028 [quant-ph]* (Nov. 2014). arXiv: 1411.4028 [quant-ph].

[63] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem". In: *arXiv:1412.6062 [quant-ph]* (June 2015). arXiv: 1412.6062 [quant-ph].

[64] Edward Farhi and Aram W. Harrow. "Quantum Supremacy through the Quantum Approximate Optimization Algorithm". In: *arXiv:1602.07674 [quant-ph]* (Oct. 2019). arXiv: 1602.07674 [quant-ph].

[65] Richard P. Feynman. "Simulating Physics with Computers". en. In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pp. 467–488. ISSN: 0020-7748, 1572-9575. DOI: 10.1007/BF02650179.

[66] Mark Fingerhuth, Tomáš Babej, and Christopher Ing. "A Quantum Alternating Operator Ansatz with Hard and Soft Constraints for Lattice Protein Folding". In: *arXiv:1810.13411 [quant-ph]* (Oct. 2018). arXiv: 1810.13411 [quant-ph].

[67] Austin G. Fowler, Ashley M. Stephens, and Peter Groszkowski. "High-Threshold Universal Quantum Computation on the Surface Code". en. In: *Physical Review A* 80.5 (Nov. 2009), p. 052312. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.80.052312.

[68] E. Schuyler Fried et al. “qTorch: The Quantum Tensor Contraction Handler”. en. In: *PLOS ONE* 13.12 (Dec. 2018). Ed. by Itay Hen, e0208510. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0208510.

[69] M. R. Garey et al. “Scheduling Unit–Time Tasks with Arbitrary Release Times and Deadlines”. en. In: *SIAM Journal on Computing* 10.2 (May 1981), pp. 256–269. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/0210018.

[70] Ian Gent and Toby Walsh. “Phase Transitions and Annealed Theories: Number Partitioning as a Case Study”. In: *In Proceedings of ECAI-96*. John Wiley & Sons, 1996, pp. 170–174.

[71] Ian P. Gent and Toby Walsh. “The Hardest Random SAT Problems”. en. In: *KI-94: Advances in Artificial Intelligence*. Ed. by Bernhard Nebel and Leonie Dreschler-Fischer. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1994, pp. 355–366. ISBN: 978-3-540-48979-5. DOI: 10.1007/3-540-58467-6_31.

[72] Ian P. Gent and Toby Walsh. “The TSP Phase Transition”. en. In: *Artificial Intelligence* 88.1-2 (Dec. 1996), pp. 349–358. ISSN: 00043702. DOI: 10.1016/S0004-3702(96)00030-6.

[73] Andreas Goerdt. “A Threshold for Unsatisfiability”. en. In: *Journal of Computer and System Sciences* 53.3 (Dec. 1996), pp. 469–486. ISSN: 00220000. DOI: 10.1006/jcss.1996.0081.

[74] Lov K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. en. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing - STOC ’96*. Philadelphia, Pennsylvania, United States: ACM Press, 1996, pp. 212–219. ISBN: 978-0-89791-785-8. DOI: 10.1145/237814.237866.

[75] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. “Practical Optimization for Hybrid Quantum-Classical Algorithms”. In: *arXiv:1701.01450 [quant-ph]* (Jan. 2017). arXiv: 1701.01450 [quant-ph].

[76] Stuart Hadfield. “On the Representation of Boolean and Real Functions as Hamiltonians for Quantum Computing”. In: *arXiv:1804.09130 [quant-ph]* (Apr. 2018). arXiv: 1804.09130 [quant-ph].

[77] Stuart Hadfield et al. “From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz”. en. In: *Algorithms* 12.2 (Feb. 2019), p. 34. ISSN: 1999-4893. DOI: 10.3390/a12020034.

[78] Stuart Hadfield et al. “Quantum Approximate Optimization with Hard and Soft Constraints”. In: *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*. PMES’17. Denver, CO, USA: Association for Computing Machinery, Nov. 2017, pp. 15–21. ISBN: 978-1-4503-5126-3. DOI: 10.1145/3149526.3149530.

[79] Asger Halkier et al. “Basis-Set Convergence in Correlated Calculations on Ne, N2, and H2O”. en. In: *Chemical Physics Letters* 286.3-4 (Apr. 1998), pp. 243–252. ISSN: 00092614. DOI: 10.1016/S0009-2614(98)00111-0.

[80] Daniel J. Harvey and David R. Wood. “The Treewidth of Line Graphs”. en. In: *Journal of Combinatorial Theory, Series B* 132 (Sept. 2018), pp. 157–179. ISSN: 0095-8956. DOI: 10.1016/j.jctb.2018.03.007.

[81] Matthew B. Hastings et al. “Improving Quantum Algorithms for Quantum Chemistry”. In: *Quantum Information & Computation* 15.1-2 (Jan. 2015), pp. 1–21. ISSN: 1533-7146.

[82] W. J. Hehre, R. F. Stewart, and J. A. Pople. “Self-Consistent Molecular-Orbital Methods. I. Use of Gaussian Expansions of Slater-Type Atomic Orbitals”. en. In: *The Journal of Chemical Physics* 51.6 (Sept. 1969), pp. 2657–2664. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.1672392.

[83] Trygve Helgaker, Poul Jorgensen, and Jeppe Olsen. *Molecular Electronic-Structure Theory*. Wiley, 2014. ISBN: 978-1-119-01955-8.

[84] Itay Hen and Marcelo S. Sarandy. “Driver Hamiltonians for Constrained Optimization in Quantum Annealing”. In: *Physical Review A* 93.6 (June 2016), p. 062312. DOI: 10.1103/PhysRevA.93.062312.

[85] Itay Hen and Federico M. Spedalieri. “Quantum Annealing for Constrained Optimization”. In: *Physical Review Applied* 5.3 (Mar. 2016), p. 034007. DOI: 10.1103/PhysRevApplied.5.034007.

[86] Steven Herbert and Akash Sengupta. “Using Reinforcement Learning to Find Efficient Qubit Routing Policies for Deployment in Near-Term Quantum Computers”. In: *arXiv:1812.11619 [quant-ph]* (Jan. 2019). arXiv: 1812.11619 [quant-ph].

[87] Yuichi Hirata et al. “An Efficient Method to Convert Arbitrary Quantum Circuits to Ones on a Linear Nearest Neighbor Architecture”. In: *2009 Third International Conference on Quantum, Nano and Micro Technologies*. Feb. 2009, pp. 26–33. DOI: 10.1109/ICQNM.2009.25.

[88] Wen Wei Ho and Timothy H. Hsieh. “Efficient Variational Simulation of Non-Trivial Quantum States”. en. In: *SciPost Physics* 6.3 (Mar. 2019), p. 029. ISSN: 2542-4653. DOI: 10.21468/SciPostPhys.6.3.029.

[89] Tad Hogg and Colin P. Williams. “The Hardest Constraint Problems: A Double Phase Transition”. en. In: *Artificial Intelligence* 69.1-2 (Sept. 1994), pp. 359–377. ISSN: 00043702. DOI: 10.1016/0004-3702(94)90088-4.

[90] Holger H. Hoos and Thomas Stützle. “SATLIB: An Online Resource for Resesarch on SAT”. In: *SAT 2000*. Ed. by Ian Gent, Hans van Maaren, and Toby Walsh. IOS Press, 2000, pp. 283–292.

[91] William J Huggins et al. “A Non-Orthogonal Variational Quantum Eigensolver”. In: *New Journal of Physics* 22.7 (July 2020), p. 073009. ISSN: 1367-2630. DOI: 10.1088/1367-2630/ab867b.

[92] Michael Jarret and Kianna Wan. "Improved Quantum Backtracking Algorithms Using Effective Resistance Estimates". en. In: *Physical Review A* 97.2 (Feb. 2018), p. 022337. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.97.022337.

[93] Zhang Jiang, Eleanor G. Rieffel, and Zhihui Wang. "Near-Optimal Quantum Circuit for Grover's Unstructured Search Using a Transverse Field". In: *Physical Review A* 95.6 (June 2017), p. 062317. DOI: 10.1103/PhysRevA.95.062317.

[94] Zhang Jiang et al. "Quantum Algorithms to Simulate Many-Body Physics of Correlated Fermions". In: *Physical Review Applied* 9.4 (Apr. 2018), p. 044036. DOI: 10.1103/PhysRevApplied.9.044036.

[95] Abhinav Kandala et al. "Hardware-Efficient Variational Quantum Eigensolver for Small Molecules and Quantum Magnets". en. In: *Nature* 549.7671 (Sept. 2017), pp. 242–246. ISSN: 1476-4687. DOI: 10.1038/nature23879.

[96] Tosio Kato. "On the Eigenfunctions of Many-Particle Systems in Quantum Mechanics". en. In: *Communications on Pure and Applied Mathematics* 10.2 (1957), pp. 151–177. ISSN: 00103640, 10970312. DOI: 10.1002/cpa.3160100201.

[97] S. Kirkpatrick and B. Selman. "Critical Behavior in the Satisfiability of Random Boolean Expressions". en. In: *Science* 264.5163 (May 1994), pp. 1297–1301. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.264.5163.1297.

[98] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. USA: American Mathematical Society, 2002. ISBN: 978-0-8218-3229-5.

[99] Alexei Y. Kitaev. "Quantum Measurements and the Abelian Stabilizer Problem". In: *Electron. Colloquium Comput. Complex.* 3.3 (1996).

[100] Ian D. Kivlichan et al. "Quantum Simulation of Electronic Structure with Linear Depth and Connectivity". In: *Physical Review Letters* 120.11 (Mar. 2018), p. 110501. DOI: 10.1103/PhysRevLett.120.110501.

[101] Christine Klymko, Blair D. Sullivan, and Travis S. Humble. "Adiabatic Quantum Programming: Minor Embedding with Hard Faults". en. In: *Quantum Information Processing* 13.3 (Mar. 2014), pp. 709–729. ISSN: 1573-1332. DOI: 10.1007/s11128-013-0683-9.

[102] Wolfgang Lechner. "Quantum Approximate Optimization With Parallelizable Gates". In: *IEEE Transactions on Quantum Engineering* 1 (2020), pp. 1–6. ISSN: 2689-1808. DOI: 10.1109/TQE.2020.3034798.

[103] Joonho Lee et al. "Generalized Unitary Coupled Cluster Wave Functions for Quantum Computation". In: *Journal of Chemical Theory and Computation* 15.1 (Jan. 2019), pp. 311–324. ISSN: 1549-9618. DOI: 10.1021/acs.jctc.8b01004.

[104] Gushu Li, Yufei Ding, and Yuan Xie. "Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices". In: *arXiv:1809.02573 [quant-ph]* (May 2019). arXiv: 1809.02573 [quant-ph].

[105] Chia-Chun Lin, Susmita Sur-Kolay, and Niraj K. Jha. “PAQCS: Physical Design-Aware Fault-Tolerant Quantum Circuit Synthesis”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23.7 (July 2015), pp. 1221–1234. ISSN: 1557-9999. DOI: `10.1109/TVLSI.2014.2337302`.

[106] Yi-Kai Liu, Matthias Christandl, and F. Verstraete. “Quantum Computational Complexity of the N-Representability Problem: QMA Complete”. In: *Physical Review Letters* 98.11 (Mar. 2007), p. 110503. DOI: `10.1103/PhysRevLett.98.110503`.

[107] Seth Lloyd. “Quantum Approximate Optimization Is Computationally Universal”. In: *arXiv:1812.11075 [quant-ph]* (Dec. 2018). arXiv: `1812.11075 [quant-ph]`.

[108] Andrew Lucas. “Ising Formulations of Many NP Problems”. In: *Frontiers in Physics* 2 (2014). ISSN: 2296-424X. DOI: `10.3389/fphy.2014.00005`.

[109] Aaron Lye, Robert Wille, and Rolf Drechsler. “Determining the Minimal Number of Swap Gates for Multi-Dimensional Nearest Neighbor Quantum Circuits”. In: *The 20th Asia and South Pacific Design Automation Conference*. Jan. 2015, pp. 178–183. DOI: `10.1109/ASPDAC.2015.7059001`.

[110] Yuri Manin. *Computable and Uncomputable*. Russian. Moscow: Sovetskoye Radio, 1980.

[111] Igor L. Markov and Yaoyun Shi. “Simulating Quantum Computation by Contracting Tensor Networks”. en. In: *SIAM Journal on Computing* 38.3 (Jan. 2008), pp. 963–981. ISSN: 0097-5397, 1095-7111. DOI: `10.1137/050644756`.

[112] S. Marsh and J. B. Wang. “A Quantum Walk-Assisted Approximate Algorithm for Bounded NP Optimisation Problems”. In: *Quantum Information Processing* 18.3 (Mar. 2019), pp. 1–18. ISSN: 1570-0755. DOI: `10.1007/s11128-019-2171-3`.

[113] Dmitri Maslov, Sean M. Falconer, and Michele Mosca. “Quantum Circuit Placement: Optimizing Qubit-to-Qubit Interactions through Mapping Quantum Circuits into a Physical Experiment”. In: *2007 44th ACM/IEEE Design Automation Conference*. June 2007, pp. 962–965.

[114] Jarrod R McClean et al. “Discontinuous Galerkin Discretization for Quantum Simulation of Chemistry”. In: *New Journal of Physics* 22.9 (Sept. 2020), p. 093015. ISSN: 1367-2630. DOI: `10.1088/1367-2630/ab9d9f`.

[115] Jarrod R McClean et al. “The Theory of Variational Hybrid Quantum-Classical Algorithms”. In: *New Journal of Physics* 18.2 (Feb. 2016), p. 023023. ISSN: 1367-2630. DOI: `10.1088/1367-2630/18/2/023023`.

[116] Jarrod R. McClean et al. “Barren Plateaus in Quantum Neural Network Training Landscapes”. en. In: *Nature Communications* 9.1 (Nov. 2018), p. 4812. ISSN: 2041-1723. DOI: `10.1038/s41467-018-07090-4`.

[117] Stephan Mertens. "Phase Transition in the Number Partitioning Problem". In: *Physical Review Letters* 81.20 (Nov. 1998), pp. 4281–4284. DOI: 10.1103/PhysRevLett.81.4281.

[118] David Mitchell, Bart Selman, and Hector Levesque. "Hard and Easy Distributions of SAT Problems". In: *Proceedings of the Tenth National Conference on Artificial Intelligence*. AAAI'92. San Jose, California: AAAI Press, July 1992, pp. 459–465. ISBN: 978-0-262-51063-9.

[119] Ashley Montanaro. "Quantum-Walk Speedup of Backtracking Algorithms". In: *Theory of Computing* 14.15 (2018), pp. 1–24. DOI: 10.4086/toc.2018.v014a015.

[120] Cristopher Moore et al. "A Continuous-Discontinuous Second-Order Transition in the Satisfiability of Random Horn-SAT Formulas". In: *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*. Ed. by David Hutchison et al. Vol. 3624. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 414–425. ISBN: 978-3-540-28239-6. DOI: 10.1007/11538462_35.

[121] Mario Motta et al. "Low Rank Representations for Quantum Simulation of Electronic Structure". In: *arXiv:1808.02625 [physics, physics:quant-ph]* (Aug. 2018). arXiv: 1808.02625 [physics, physics:quant-ph].

[122] Marcel Nooijen. "Can the Eigenstates of a Many-Body Hamiltonian Be Represented Exactly Using a General Two-Body Cluster Expansion?" In: *Physical Review Letters* 84.10 (Mar. 2000), pp. 2108–2111. DOI: 10.1103/PhysRevLett.84.2108.

[123] Bryan O'Gorman. "Parameterization of Tensor Network Contraction". en. In: (2019), 19 pages. DOI: 10.4230/LIPICS.TQC.2019.10.

[124] Bryan O'Gorman et al. "Electronic Structure in a Fixed Basis Is QMA-Complete". In: *arXiv:2103.08215 [quant-ph]* (Mar. 2021). arXiv: 2103.08215 [quant-ph].

[125] Bryan O'Gorman et al. "Generalized Swap Networks for Near-Term Quantum Computing". In: *arXiv:1905.05118 [physics, physics:quant-ph]* (May 2019). arXiv: 1905.05118 [physics, physics:quant-ph].

[126] G. Ortiz et al. "Quantum Algorithms for Fermionic Simulations". In: *Physical Review A* 64.2 (July 2001), p. 022319. DOI: 10.1103/PhysRevA.64.022319.

[127] M. I. Ostrovskii. "Minimal Congestion Trees". en. In: *Discrete Mathematics* 285.1-3 (Aug. 2004), pp. 219–226. ISSN: 0012-365X. DOI: 10.1016/j.disc.2004.02.009.

[128] J. S. Otterbach et al. "Unsupervised Machine Learning on a Hybrid Quantum Computer". In: *arXiv:1712.05771 [quant-ph]* (Dec. 2017). arXiv: 1712.05771 [quant-ph].

[129] Edwin Pednault et al. "Pareto-Efficient Quantum Circuit Simulation Using Tensor Contraction Deferral". In: *arXiv:1710.05867 [quant-ph]* (Aug. 2020). arXiv: 1710.05867 [quant-ph].

[130] Alberto Peruzzo et al. "A Variational Eigenvalue Solver on a Photonic Quantum Processor". en. In: *Nature Communications* 5.1 (Sept. 2014), p. 4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[131] Stephen Piddock and Ashley Montanaro. "The Complexity of Antiferromagnetic Interactions and 2D Lattices". In: *Quantum Information & Computation* 17.7-8 (June 2017), pp. 636–672. ISSN: 1533-7146.

[132] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. en. Fourth. New York: Springer-Verlag, 2012. ISBN: 978-1-4899-9043-3. DOI: 10.1007/978-1-4614-2361-4.

[133] John Preskill. "Quantum Computing in the NISQ Era and Beyond". en-GB. In: *Quantum* 2 (Aug. 2018), p. 79. DOI: 10.22331/q-2018-08-06-79.

[134] Patrick Prosser. "An Empirical Study of Phase Transitions in Binary Constraint Satisfaction Problems". In: *Artificial Intelligence* 81.1-2 (1996), pp. 81–109.

[135] Jean-Charles Régin. "A Filtering Algorithm for Constraints of Difference in CSPs". In: *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*. AAAI '94. Seattle, Washington, USA: American Association for Artificial Intelligence, Oct. 1994, pp. 362–367. ISBN: 978-0-262-61102-2.

[136] Eleanor G. Rieffel et al. "A Case Study in Programming a Quantum Annealer for Hard Operational Planning Problems". en. In: *Quantum Information Processing* 14.1 (Dec. 2014), pp. 1–36. ISSN: 1570-0755, 1573-1332. DOI: 10.1007/s11128-014-0892-x.

[137] Eleanor G. Rieffel et al. "Parametrized Families of Hard Planning Problems from Phase Transitions". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI'14. Québec City, Québec, Canada: AAAI Press, July 2014, pp. 2337–2343.

[138] Neil Robertson and P. D. Seymour. "Graph Minors. X. Obstructions to Tree-Decomposition". en. In: *Journal of Combinatorial Theory, Series B* 52.2 (July 1991), pp. 153–190. ISSN: 0095-8956. DOI: 10.1016/0095-8956(91)90061-N.

[139] Jonathan Romero et al. "Strategies for Quantum Computing Molecular Energies Using the Unitary Coupled Cluster Ansatz". In: *Quantum Science and Technology* 4.1 (Oct. 2018), p. 014008. ISSN: 2058-9565. DOI: 10.1088/2058-9565/aad3e4.

[140] Mehdi Saeedi, Robert Wille, and Rolf Drechsler. "Synthesis of Quantum Circuits for Linear Nearest Neighbor Architectures". en. In: *Quantum Information Processing* 10.3 (June 2011), pp. 355–377. ISSN: 1573-1332. DOI: 10.1007/s11128-010-0201-2.

[141] Norbert Schuch and Frank Verstraete. "Computational Complexity of Interacting Electrons and Fundamental Limitations of Density Functional Theory". en. In: *Nature Physics* 5.10 (Oct. 2009), pp. 732–735. ISSN: 1745-2473, 1745-2481. DOI: 10.1038/nphys1370. arXiv: 0712.0483.

[142] P. D. Seymour and R. Thomas. "Call Routing and the Ratcatcher". en. In: *Combinatorica* 14.2 (June 1994), pp. 217–241. ISSN: 1439-6912. DOI: 10.1007/BF01215352.

[143] Jiří Sgall. "Open Problems in Throughput Scheduling". In: *Algorithms – ESA 2012*. Ed. by David Hutchison et al. Vol. 7501. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 2–11. ISBN: 978-3-642-33090-2. DOI: 10.1007/978-3-642-33090-2_2.

[144] P.W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. Santa Fe, NM, USA: IEEE Comput. Soc. Press, 1994, pp. 124–134. ISBN: 978-0-8186-6580-6. DOI: 10.1109/SFCS.1994.365700.

[145] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". en. In: *SIAM Review* 41.2 (Jan. 1999), pp. 303–332. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/S0036144598347011.

[146] Barbara Simons. "A Fast Algorithm for Single Processor Scheduling". In: *19th Annual Symposium on Foundations of Computer Science (Sfcs 1978)*. Ann Arbor, MI, USA: IEEE, Oct. 1978, pp. 246–252. DOI: 10.1109/SFCS.1978.4.

[147] Marcos Yukio Siraichi et al. "Qubit Allocation". en. In: *Proceedings of the 2018 International Symposium on Code Generation and Optimization*. Vienna Austria: ACM, Feb. 2018, pp. 113–125. ISBN: 978-1-4503-5617-6. DOI: 10.1145/3168822.

[148] Barbara M. Smith and Martin E. Dyer. "Locating the Phase Transition in Binary Constraint Satisfaction Problems". In: *Artificial Intelligence* 81.1 (1996), pp. 155–181. ISSN: 0004-3702. DOI: 10.1016/0004-3702(95)00052-6.

[149] The Parameterized Algorithms and Computational Experiments Challenge. *Track A: Treewidth.* https://pacechallenge.wordpress.com/pace-2017/track-a-treewidth/. 2017.

[150] Basil Vandegriend and Joseph Culberson. "The $G_{n,m}$phase Transition Is Not Hard for the Hamiltonian Cycle Problem". In: *Journal of Artificial Intelligence Research* 9.1 (Nov. 1998), pp. 219–245. ISSN: 1076-9757.

[151] Davide Venturelli et al. "Compiling Quantum Circuits to Realistic Hardware Architectures Using Temporal Planners". In: *Quantum Science and Technology* 3.2 (Apr. 2018), p. 025004. ISSN: 2058-9565. DOI: 10.1088/2058-9565/aaa331.

[152] Guillaume Verdon, Michael Broughton, and Jacob Biamonte. "A Quantum Algorithm to Train Neural Networks Using Low-Depth Circuits". In: *arXiv:1712.05304 [cond-mat, physics:quant-ph]* (Aug. 2019). arXiv: 1712.05304 [cond-mat, physics:quant-ph].

[153] Benjamin Villalonga et al. "A Flexible High-Performance Simulator for Verifying and Benchmarking Quantum Circuits Implemented on Real Hardware". en. In: *npj Quantum Information* 5.1 (Oct. 2019), pp. 1–16. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0196-1.

[154] Zhihui Wang et al. "XY Mixers: Analytical and Numerical Results for the Quantum Alternating Operator Ansatz". In: *Physical Review A* 101.1 (Jan. 2020), p. 012320. DOI: 10.1103/PhysRevA.101.012320.

[155] Zhihui Wang et al. "An Investigation of Phase Transitions in Single-Machine Scheduling Problems". en. In: *Twenty-Seventh International Conference on Automated Planning and Scheduling*. June 2017.

[156] Zhihui Wang et al. "Quantum Approximate Optimization Algorithm for MaxCut: A Fermionic View". In: *Physical Review A* 97.2 (Feb. 2018), p. 022304. DOI: 10.1103/PhysRevA.97.022304.

[157] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. "Progress towards Practical Quantum Variational Algorithms". In: *Physical Review A* 92.4 (Oct. 2015), p. 042303. DOI: 10.1103/PhysRevA.92.042303.

[158] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. "Training a Quantum Optimizer". In: *Physical Review A* 94.2 (Aug. 2016), p. 022309. DOI: 10.1103/PhysRevA.94.022309.

[159] Tzu-Chieh Wei, Michele Mosca, and Ashwin Nayak. "Interacting Boson Problems Can Be QMA Hard". In: *Physical Review Letters* 104.4 (Jan. 2010), p. 040501. DOI: 10.1103/PhysRevLett.104.040501.

[160] James Daniel Whitfield, Peter John Love, and Alán Aspuru-Guzik. "Computational Complexity in Electronic Structure". en. In: *Physical Chemistry Chemical Physics* 15.2 (Dec. 2012), pp. 397–411. ISSN: 1463-9084. DOI: 10.1039/C2CP42695A.

[161] James Daniel Whitfield, Norbert Schuch, and Frank Verstraete. "The Computational Complexity of Density Functional Theory". en. In: *Many-Electron Approaches in Physics, Chemistry and Mathematics: A Multidisciplinary View*. Ed. by Volker Bach and Luigi Delle Site. Mathematical Physics Studies. Cham: Springer International Publishing, 2014, pp. 245–260. ISBN: 978-3-319-06379-9. DOI: 10.1007/978-3-319-06379-9_14.

[162] Robert Wille et al. "Look-Ahead Schemes for Nearest Neighbor Optimization of 1D and 2D Quantum Circuits". In: *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jan. 2016, pp. 292–297. DOI: 10.1109/ASPDAC.2016.7428026.

[163] Zhi-Cheng Yang et al. "Optimizing Variational Quantum Algorithms Using Pontryagin's Minimum Principle". In: *Physical Review X* 7.2 (May 2017), p. 021027. DOI: 10.1103/PhysRevX.7.021027.

[164] Arman Zaribafiyan, Dominic J. Marchand, and Seyed Saeed Changiz Rezaei. "Systematic and Deterministic Graph Minor Embedding for Cartesian Products of Graphs". In: *Quantum Information Processing* 16.5 (May 2017), pp. 1–26. ISSN: 1570-0755. DOI: 10.1007/s11128-017-1569-z.

[165] Alwin Zulehner, Alexandru Paler, and Robert Wille. "Efficient Mapping of Quantum Circuits to the IBM QX Architectures". In: *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*. Mar. 2018, pp. 1135–1138. DOI: 10.23919/DATE.2018.8342181.