ABSTRACT

Title of Dissertation: DESIGN AND OPTIMIZATION IN NEAR-TERM QUANTUM COMPUTATION

Aniruddha A. Bapat
Doctor of Philosophy, 2021

Dissertation Directed by: Professor Alexey V. Gorshkov
Department of Physics

Professor Stephen P. Jordan
Department of Physics

Quantum computers have come a long way since conception, and there is still a long way to go before the dream of universal, fault-tolerant computation is realized. In the near term, quantum computers will occupy a middle ground that is popularly known as the "Noisy, Intermediate-Scale Quantum" (or NISQ) regime. The NISQ era represents a transition in the nature of quantum devices from experimental to computational. There is significant interest in engineering NISQ devices and NISQ algorithms in a manner that will guide the development of quantum computation in this regime and into the era of fault-tolerant quantum computing.

In this thesis, we study two aspects of near-term quantum computation. The first of these is the design of device architectures, covered in Chapters 2 to 4. We examine different qubit connectivities on the basis of their graph properties, and present numerical and analytical results on the speed at which large entangled states can be created on nearest-neighbor grids and graphs with modular structure. Next, we discuss the

problem of permuting qubits among the nodes of the connectivity graph using only local operations, also known as routing. Using a fast quantum primitive to reverse the qubits in a chain, we construct a hybrid, quantum/classical routing algorithm on the chain. We show via rigorous bounds that this approach is faster than any SWAP-based algorithm for the same problem.

The second part, which spans Chapters 5 to 7, discusses variational algorithms, which are a class of algorithms particularly suited to near-term quantum computation. Two prototypical variational algorithms, quantum adiabatic optimization (QAO) and quantum approximate optimization algorithm (QAOA), are studied for the difference in their control strategies. We show that on certain crafted problem instances, bang-bang control (QAOA) can be as much as exponentially faster than quasistatic control (QAO). Next, we demonstrate the performance of variational state preparation on an analog quantum simulator based on trapped ions. We show that using classical heuristics that exploit structure in the variational parameter landscape, one can find circuit parameters efficiently in system size as well as circuit depth. In the experiment, we approximate the ground state of a critical Ising model with long-ranged interactions on up to 40 spins. Finally, we study the performance of Local Tensor, a classical heuristic algorithm inspired by QAOA on benchmarking instances of the MaxCut problem, and suggest physically motivated choices for the algorithm hyperparameters that are found to perform well empirically. We also show that our implementation of Local Tensor mimics imaginary-time quantum evolution under the problem Hamiltonian.

DESIGN AND OPTIMIZATION IN NEAR-TERM QUANTUM COMPUTATION

by

Aniruddha A. Bapat

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Zohreh Davoudi, Chair
Professor Alexey V. Gorshkov, Co-chair/Co-advisor
Professor Stephen P. Jordan, Co-advisor
Professor Christopher Monroe
Professor Andrew M. Childs, Dean's Representative

Acknowledgments

It is impossible to acknowledge everyone in a reasonable amount of space, but I will try my best. To those who could not be named here, thank you for helping me reach this important academic milestone.

First and foremost, I thank my advisors, Alexey Gorshkov and Stephen Jordan. It has been a pleasure to work with both of them, and I feel privileged to have received the mentorship of two of the finest researchers in the field today. Obviously, this work could not have been possible without the time and interest they devoted to my academic development, so, from the bottom of my heart, thank you Stephen and Alexey!

I owe a large part of my research accomplishments to QuICS and the academics and administrators who make it the world-class hub of quantum computation that it is today - thank you all! I extend my sincere thanks to Andrew Childs, not only for his indirect support as QuICS co-director but also for his mentorship and collaboration on our common projects. Speaking of mentors, I am very grateful to Zohreh Davoudi, who has been like a third advisor to me the last two years. Thank you, Zohreh! I also thank Bill Dorland, who kindly agreed to chair my candidacy presentation, and Professor P. S. Krishnaprasad, who was generous with his time whenever I came to him with questions on control theory.

This thesis, and indeed, my education, would be incomplete without my colleagues

and collaborators at UMD. In particular, I thank my friend and collaborator on all things routing, Eddie Schoute. I also thank Lucas Brady, Zachary Eldredge, Abhinav Deshpande, Jim Garrison, Niklas Mueller, Indrakshi Raychowdhury, Przemyslaw Bienias, Yaroslav Kharkov, Chris Baldwin, Dhruv Devulapalli, Chris White, Fangli Liu, Guido Pagano, Wen Lin Tan, and all the members of the QSim lab in the Monroe group. I also want to thank Chris Monroe himself, who made my collaboration with QSim possible!

Among my friends, I thank Troy Sewell, a friend, collaborator, office mate, and partner in speculative banter. I also thank Yidan Wang for her friendship. Additionally, I want to name a few special groups of friends from during and before the years of graduate school. I'm not always social, but somehow these excellent people found and adopted me! So, I thank The Pitty Gang, The Blackfeet, The Lanczos Four, The Founding Fathers of Catawba, The Core Tex Crew, and The Triple. You know who you are.

My family is always with me, no matter the distance between us (which, on average, is *a lot*). My parents made so many sacrifices in their personal life, but never compromised even an iota when it came to their children. I cannot thank my mother Swatee Bapat enough. She nurtured me, and what little of her relentless drive stayed with me is what keeps me afloat today. I thank my sister Asilata Bapat for being brilliant and infecting me with a bit of that enthusiasm she always had for learning. I could not have asked for a better role model, singing partner, and sharer of inside jokes. And since I am most like my father Anand Bapat by nature, I thank him for, well, making me me. Lastly, I thank my soulmate Arushi Bodas for her kind and loving companionship through the ups and downs of graduate school. She is telling me to get back to my thesis now.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

NISQ	Noise, Intermediate-Scale, Quantum
QAO	Quantum Adiabatic Optimization
QAOA	Quantum Approximate Optimization Algorithm
VQE	Variational Quantum Eigensolver
QA	Quantum Annealing
NP	Non-deterministic Polynomial
BQP	Bounded-error, Quantum Probabilistic polynomial
IQP	Instantaneous Quantum Polynomial
GHZ	Greenberger-Horne-Zeilinger
CNOT	Controlled NOT
LOCC	Local Operators and Classical Communication
TFIM	Transverse-Field Ising Model
OES	Odd-Even Sort
TBS	Tripartite Binary Sort
ATBS	Adaptive Tripartite Binary Sort
GDC	Generic Divide-and-Conquer
SA	Simulated Annealing
BBSA	Bang-bang Simulated Annealing
PMP	Pontryagin's Minimum Principle
DMRG	Density Matrix Renormalization Group
COM	Center Of Mass
KL	Kullback-Liebler
LT	Local Tensor
LP	Linear Programming
QP	Quadratic Programming
SDP	Semi-Definite Programming
GW	Goemans-Williamson

Citations to Previously Published Work

Much of this dissertation appeared in papers already published. Here we outline those publications, with additional citations to related papers whose text does not appear here.

- Chapter 2: "Unitary entanglement construction in hierarchical networks," A. Bapat, Z. Eldredge, J.R. Garrison, A. Desphande, F.T. Chong, A.V. Gorshkov, Phys. Rev. A 98, 062328. 2018.
  - Subsequent work on entanglement construction on networks in the non-unitary setting can be found in "Entanglement bounds on the performance of quantum computing architectures" L. Zhou, Z. Eldredge, A. Bapat, J. Garrison, A. Deshpande, F.T. Chong, A.V. Gorshkov, Phys. Rev. Research 2, 033316, 2020.
- Chapter 3: "Nearly optimal time-independent reversal of a spin chain", A. Bapat, E. Schoute, A.V. Gorshkov, A.M. Childs, arXiv:2003.02843.
- Chapter 4: "Quantum routing with fast reversals", A. Bapat, A.M. Childs, A.V. Gorshkov, S. King, E. Schoute, H. Shastri, arXiv:2103.03264.
- Chapter 5: "Bang-bang control as a design principle for classical and quantum optimization algorithms", A. Bapat, S. P. Jordan, Quantum Inf. Comput. 19.5-6 (2019): 424-446.
  - Subsequent work on optimal control of variational algorithms can be found in "Optimal protocols in quantum annealing and quantum approximate

optimization algorithm problems" L.T. Brady, C.L. Baldwin, A. Bapat, Y. Kharkov, A.V. Gorshkov, Phys. Rev. Lett.126, 070505, 2021.

  - "Behavior of analog quantum algorithms" L.T. Brady, L. Kocia, P. Bienias, A. Bapat, Y. Kharkov, A.V. Gorshkov, arXiv:2107.01218, 2021.

- Chapter 6: "Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator", G. Pagano, A. Bapat, P. Becker, K.S. Collins, A. De, P.W. Hess, H.B. Kaplan, A. Kyprianidis, W.L. Tan, C.L. Baldwin, L.T. Brady, A. Deshpande, F. Liu, S.P. Jordan, A.V. Gorshkov, C. Monroe, Proc. Natl. Acad. Sci., 117 (41) 25396-25401, October 13, 2020.

- Chapter 7: "Approximate optimization of the MaxCut problem with a local spin algorithm", A. Bapat, S.P. Jordan, Phys. Rev. A 103 (5), 052413, 2021.

Chapter 1: Introduction

1.1 General remarks

Since its conception four decades ago [1–3], quantum computation has progressed from infancy in the form of few-qubit quantum computers to an adolescent stage that John Preskill dubbed the "Noisy, Intermediate-Scale Quantum", or NISQ, era [4]. Like adolescence, this is a time filled with unprecedented growth and a combination of uncertainty and excitement about the future. Since device capabilities are still limited by noise as well as the number of qubits, a fundamental goal in this era is to construct well-designed device architectures and algorithms that deliver the "biggest bang for the buck". In this thesis, we study several problems motivated by this maxim.

Before giving a technical overview of the thesis, let us understand the nature of quantum computation in the near-term. In the last decade, practical quantum computation has made rapid progress. Small quantum computers, i.e., devices on 1-10 physical qubits, have now been engineered with an impressive degree of control on a range of platforms. However, it would be more accurate to describe systems at this scale as physics experiments than as computers. The ultimate goal of quantum computation is the universal, fault-tolerant quantum computer, which can run arbitrary quantum circuits with an error that can be made arbitrarily small via efficient quantum error correction

procedures. The majority of proposals for fault-tolerance involve the use of the logical qubit, which is a redundant composite of several physical qubits. Depending on the error-correction scheme and the algorithm to be implemented, the cost of engineering logical qubits and fault-tolerant logical operations on them could be beyond thousands of physical qubits *per logical qubit*. For example, Ref. [5] estimates that using the surface code, it could take about 46 million superconducting qubits to factorize a 1024-bit number using Shor's algorithm. This is well beyond the capabilities of near-term quantum computing platforms.

Pushing the technological envelope a little closer towards that end goal, NISQ computers are pre-fault-tolerant machines operating on the scale of 10-1000 qubits. As such, they are noisy and support circuits of limited size, but at 100 qubits, the computation is already too expensive to be directly represented on a classical computer due to an exponential blowup in the Hilbert space dimension. This puts the NISQ regime in an interesting middle ground where it is plausible that there are useful, non-classical computational problems to be solved, but it is less clear how to solve them within the NISQ constraints. There are many challenges to be overcome at once: Which problems are promising? Which physical platforms are most scalable in, e.g., "quantum volume" [6]? What is the optimal qubit connectivity? How to mitigate the effects of noise? And so on.

In NISQ architectures, a major design constraint is the allowed connectivity between qubits. This is evident when one looks at current NISQ platforms such as trapped ions or superconducting qubits. A single ion trap consists of multiple ions (i.e., the physical qubits) physically confined and interacting via phonon modes in the trap.

Since any two-qubit gate can be performed via the appropriate combination of phonon excitations, every qubit is connected to every other qubit. However, the control required to achieve full connectivity does not scale well with the number of trapped ions, capping out at the scale of about 100 ions [7, 8]. A more scalable solution to this issue is to connect multiple ion traps (or modules) via a photonic network that only connects to one designated "communicator" qubit per trap. Superconducting qubit systems opt for grid-like geometries due to the engineering constraints present in such systems [9]. Therefore, one trades qubit connectivity for scalability. Even when all-to-all connectivity is possible, it could prove more feasible to divide the system into distinct units, much the way a traditional architecture separates CPU and RAM.

Another hallmark of the NISQ era is that the architecture and the algorithms depend on each other. In contrast, on a hypothetical universal quantum computer, we expect the underlying device architecture to become an irrelevant concern to algorithm design. There are three primary features of NISQ devices that constrain NISQ algorithms: lack of fault tolerance, intermediate size, and non-universality. Non-fault-tolerance implies that NISQ algorithms must be shallow-depth. While error mitigation (e.g.,Refs. [10, 11]) can improve the quality of output, qubit coherence times still set a limit on the runtime. Secondly, intermediate size implies that NISQ algorithms must target problems that lie outside the realm of classical computation at relatively modest input size. This has motivated the search for *quantum supremacy*, which is the demonstration of a quantum algorithm for a task that is beyond realistic classical capabilities. And while this milestone has recently been crossed on the problem of sampling the output distribution of a pseudo-random, constant-depth circuit on 53 qubits [12], it is hard to make the case for the

applicability of such an algorithm. Lastly, non-universality may constrain the set of operations that can be feasibly performed, which further restricts the scope of NISQ algorithms. Therefore, to design solutions that fit within these tight constraints demands a creativity that is not unlike writing poetry!

Now we introduce the themes central to this thesis.

1.2 Quantum architectures

In the context of computing, the word 'architecture' refers to the organization of individual informational units (bits or qubits) into progressively more abstract logical structures. For example, 8 bits together form a register that can hold data such as an integer with value less than $256 = 2^8$. Several registers may be organized together into memory buffers, logic units, etc., which then organize to form a memory chip, or a processor. Finally, several CPUs may be connected to form a computing cluster. Since quantum information is fundamentally different from classical information, the architecture of quantum computers will necessarily look different. A great example of this is the concept of quantum random access memory, or QRAM [13], which, while analogous to the (classical) RAM, is unique to the quantum paradigm in that it can be queried *in superposition* the way one would query a quantum oracle. In time, quantum architectures will grow in complexity as layers of abstraction are added one at a time. Currently, however, there are more fundamental challenges in our understanding of qubit connectivity in quantum devices. We address some of these questions in the thesis.

For the remainder of this discussion, we shall express a qubit architecture by its

essential details, namely, as a graph where the nodes are the qubits, and the edges connect pairs of qubits whenever it is possible to address them via two-qubit interactions.

1.2.1 Evaluating a quantum architecture

Let us start with the following question. Given N qubits, what is the optimal way to connect them? Given no additional information, the answer is clear: the complete graph, K_N. This way, a gate is implementable on any two qubits, implying that circuits can be mapped to this architecture with trivial overhead.

However, as we have seen with the examples of ion traps, complete connectivity may not be scalable in N due to engineering constraints. Therefore, it is natural to assign a cost to adding connections, or to set hard constraints on the number of edges allowed as a function of the number of qubits. Suppose the number of edges is required to be linear in N; what is the optimal connectivity then? In that case, it is less clear how to choose decisively from the undoubtedly vast set of valid graphs.

To begin to answer such questions, one can look to the field of graph theory, which has a rich history that spans many disciplines. Whether it is the layout of telephone lines, the structure of social networks or the connectivity of qubits, properties about the underlying graph structure can provide a language to compare seemingly incomparable graphs.

In Chapter 2, we use a graph-theoretic approach to argue about the efficacy of several different architectures for quantum applications. We analyze how the graphs perform under the benchmark of the time taken to prepare the globally entangled GHZ

state, $\frac{1}{\sqrt{2}}|00\ldots0\rangle + \frac{1}{\sqrt{2}}|11\ldots1\rangle$, via local unitary operations. We also design heuristics for circuit mapping on modular graphs. This work is based on Refs. [14].

1.2.2 Quantum routing

Quantum algorithms are not always designed with architectural constraints in mind. Therefore, there arises an inevitable problem of mapping an algorithm to a restricted architecture. To illustrate this point, consider an $N \times N$ grid of qubits. An arbitrary quantum algorithm will involve gates on distant qubits. To implement such a gate, it is necessary to first move the distant qubits so that they become neighbors. In one layer of a circuit, there could be as many as $O(N^2)$ two-qubit gates, each acting on qubits that are a distance $O(N)$ apart. If one performed each gate serially, it would take time $O(N^3)$ to implement one circuit layer. In actuality, it takes only a $O(N)$-depth SWAP circuit to implement any permutation on the grid [15]. Such polynomial improvements could be the difference between a prohibitive circuit and a NISQ-implementable circuit. Therefore, it is essential to solve the *quantum routing problem*, which is the problem of implementing permutations on a qubit graph using local operations.

In a model of routing involving data of unspecified type, the operations are restricted to neighboring packet swaps. But since we are interested specifically in the routing of qubits, it is natural to consider the possibility of speedups to routing given access to local quantum operations. In fact, there is already evidence for such speedups. It is known that using engineered Hamiltonians, one can swap the states of two qubits in an N-qubit system polynomially faster than the distance between the qubits [16]. Similarly, it is

possible to teleport quantum data across long distances essentially instantaneously via entanglement swapping [17]. With the addition of such tools, it is plausible that quantum algorithms for routing could be much faster than a simple, SWAP-based approach. However, while the tools above are directly applicable to the swapping of two qubits, the routing problem is more general, and therefore more challenging. This is because

1. we demand no dependence on the initial state,

2. the goal is to implement any permutation, not just a special long-distance swap, and

3. we wish to design routing algorithms on a variety of graph geometries, not just a chain or a grid.

In Chapter 3, we show that using a time-independent Hamiltonian, it is possible to carry out a reversal of a spin chain faster than any SWAP-based protocol. The reversal is a special permutation that swaps qubits about the center of the chain. This work is based on Refs. [18].

Then, in Chapter 4, we construct algorithms that use the fast reversal as a quantum primitive to carry out faster-than-SWAP routing of any permutation on the chain. We provide rigorous bounds on the runtime of our routing algorithms, and also analyze their average-case runtime. This work is based on Ref. [19].

Note that one can think of the above approach as a hybrid quantum/classical scheme. It could be quite challenging (or even impossible) to design a family of faster-than-classical circuits for every input permutation on a given graph. A more expedient approach is to generate general permutations by composing a sequence of known quantum protocols for special permutations. The task of finding the correct sequence for a given

permutation is non-trivial and may require sophisticated classical algorithms. Therefore, one can imagine a system where fast quantum protocols are discovered and fed into a classical apparatus for finding new, faster ways to implement routing.

In the next section, the hybrid approach will make a reappearance as a tool for designing optimization algorithms on NISQ devices.

1.3 Variational Algorithms

Traditional quantum algorithm design assumes access to a universal, fault-tolerant quantum computer with polynomial space and time in the problem size. However, many algorithms designed this way are rendered unimplementable on NISQ devices, which are limited in size, noisy, and non-universal. In the past few years, the search for NISQ applications has brought about a shift in perspectives regarding algorithm design. Instead of thinking of the circuit as a static construct, one can reframe the computation as a parameterized, time-constrained evolution of the quantum state. The choice of parameters is problem-dependent, and may be discovered via classical optimization that queries the output of the circuit with a particular choice of parameters. Thus, the quantum and classical parts can work in tandem as co-processors. This hybrid approach effectively lightens the computational burden on the quantum device by outsourcing the parameter search to the classical computer.

There are several ways to analyze the usefulness of a parameterized circuit design, of which we note two: expressibility and control. The expressibility of a parameterized circuit framework is its ability to approximate states in the Hilbert space [20]. Control,

on the other hand, is concerned with achieving a specific outcome out of all possible expressions of a parameterized circuit, and is closely related to the notions of trainability and reachability also found in the literature. In fact, it has been found that too much expressibility can lead to low trainability due to the appearance of the so-called 'barren plateaus' in the cost landscape [21]. Results like this are indicative of the need for good design in parameterized quantum circuits.

One of the most promising applications of the parameterized circuit approach is towards problems of approximate optimization. Broadly, the goal of these algorithms is to minimize a given cost function, which is usually the energy of a target Hamiltonian. This problem finds extensive applications in physics, chemistry, material science, as well as in classical combinatorial optimization problems that are reformulated as ground state minimization problems on spin variables. Algorithms for approximate optimization have come to be known as variational algorithms. The word 'variational' here is derived from the variational principle (a version of the Rayleigh-Ritz, or simply Ritz, method [22]), which asserts that the observed energy of a physical state provides an upper bound to the energy of the system's ground state.

One of the topics we explore in this thesis is the control of variational algorithms. While control theory is far from new [23], ideas from it have been only recently applied to variational algorithms [24, 25]. There are two parameterized circuit frameworks that perhaps best illustrate the difference in control strategy. The first is known as quantum adiabatic optimization (QAO) [26]. Historically, a closely related algorithm known as quantum annealing [27] was introduced around the same time, but here we will not distinguish them and focus on their similar approach. These algorithms

rely on the principle that physical systems remain in equilibrium under slow external changes. The quantum adiabatic theorem guarantees that a system that starts in the ground state of a time-dependent Hamiltonian with a finite spectral gap remains in the ground state at all times if the Hamiltonian varies infinitely slowly. (The spectral gap is the difference between the ground state energy and the energy of the first excited state.) For Hamiltonians varying at a finite rate, an approximate version of the theorem continues to hold if the rate of change is small compared to the spectral gap at any point in time. Therefore, the idea in QAO is to start in the ground state of a known Hamiltonian (canonically the transverse field $-\sum_{i=1}^{N} X_i$) and to tune the Hamiltonian slowly into the target Hamiltonian. If the gap condition is satisfied, then the final state is guaranteed to be close to the target ground state. The annealing curve, which dictates how the Hamiltonian is varied in time, is described by one or more control parameters that may be tuned to improve the convergence of the algorithm.

The second algorithm framework, proposed in 2014, is known as the quantum approximate optimization algorithm [28]. This too has a close relative, the independently proposed variational quantum eigensolver, or VQE [29]. Like QAO, the goal of QAOA is to evolve a known initial state (usually the transverse field ground state, $|+\rangle^N$) into the target state. However, unlike QAO, the evolution is not described by a smoothly time-varying Hamiltonian but rather by a sequence of alternating Hamiltonian evolutions. In the original formulation, one alternates between the initial Hamiltonian (known as the mixer) and the target Hamiltonian (the driver). The variational parameters in this framework are the evolution times (or "angles") in each round, as well as the total number of rounds, p.

The two frameworks mentioned above are by no means an exhaustive list, but they capture an essential difference in the design of NISQ circuits. In the algorithms of annealing type, the time-variation is slow and (usually) smooth. On the other hand, QAOA and VQE employ bang-bang control, where the parameters are piecewise constant functions of time and take either the maximum value or the minimum value of 0. In fact, it is possible to formulate both QAOA and QAO as two special instances of the same underlying algorithm, but with different control schedules.

In Chapter 5, we investigate the difference in performance achievable with the different forms of control present in QAOA, QAO, as well as two classical analogues of these frameworks. We show that on crafted toy instances, it is possible to obtain an exponential separation in runtime between annealing-type control and bang-bang control. This chapter is based upon Ref. [30]. (In subsequent work not covered in this thesis [31], we argue through a careful analysis that the optimal control in the variational setting is in general not bang-bang as previously believed, but instead of bang-anneal-bang form.)

In Chapter 6, we report on an implementation of VQE on an analog, trapped-ion quantum simulator to find the ground state of a critical, one-dimensional Ising model with long-ranged interactions. We show that by discovering patterns in the variational angle sequences, one can design classical heuristics for guessing good parameters that scale well with problem size and the depth of the variational circuit. This work is based upon Ref. [32] and was done in collaboration with the QSim laboratory in the Monroe group at UMD.

Finally, in Chapter 7, we study a classical optimization algorithm that is inspired by the structure of QAOA. Starting with the original construction of Hastings [33], we tune

the algorithm on benchmarking instances of the MaxCut problem. In the process, we learn heuristics for choosing the variational hyperparameters that are physically motivated. Surprisingly, the algorithm can be shown to mimic imaginary-time Schrödinger evolution under the problem Hamiltonian. This chapter is based upon work published in Ref. [34].

Chapter 2: Unitary entanglement construction in hierarchical networks

2.1 Introduction

As quantum computers grow from the small, few-qubit machines currently deployed to the large machines required to realize useful, fault-tolerant computations, it will become increasingly difficult for every physical qubit to be part of a single contiguous piece of hardware. Just as modern classical computers do not rely on a single unit of processing and memory, instead using various components such as CPUs, GPUs, and RAM, we expect that a quantum computer will likewise use specialized modules to perform different functions. At a higher level, computers can be organized into clusters, data centers, and cloud services which allow for a distributed approach to computational tasks, another paradigm quantum computers will no doubt emulate. Already, there has been significant interest in how quantum algorithms for elementary operations such as arithmetic perform in distributed-memory situations [35, 36] and how to automate the design of quantum computer architectures [37]. In addition, the construction of a fault-tolerant quantum computer naturally suggests a separation of physical qubits into groups corresponding to logical qubits, which makes modularity an attractive framework for building fault-tolerant computers [38]. Modular and scalable computing architectures have been explored for both ion trap [7, 39] and superconducting platforms [9, 40, 41].

In this chapter, we use tools from graph theory to discuss benefits and drawbacks of different potential architectures for a modular quantum computer. A graph-theoretic approach allows us to flexibly examine a wide range of possible arrangements quantitatively and allows for convenient numerical simulation using existing software packages designed for network analysis [42]. We especially wish to focus on families of graphs that can scale with the desired number of qubits. In general, we assume that connectivity, i.e., being able to quickly perform operations between nodes, is desirable in an architecture, but that building additional graph edges is in some way costly or difficult, and so will try to minimize the number of needed edges to achieve a highly communicative graph.

We will make use of a previously described graph-theoretic binary operation known as the hierarchical product [43, 44]. We will use this iteratively to describe a new family of graphs we dub "hierarchies." We will show that hierarchies perform well by many commonsense graph metrics and argue that they would serve as a plausible and efficient basis for a quantum computing architecture. Furthermore, we will demonstrate that these graphs allow for easily-implemented heuristic procedures to assist in the compilation of quantum algorithms.

We will examine the performance of graphs in generating large entangled states such as the multi-qubit Greenberger-Horne-Zeilinger (GHZ) state (also known as a cat state). The GHZ state has perfect quantum correlations between different qubits; it thus can be used to perform high-precision metrology [45, 46]. In addition, the creation of a GHZ state can be used as part of a state-transfer protocol, which may be useful as part of large quantum computations [47].

An additional property of GHZ state preparation and state transfer which makes them a useful starting point is that, in nearest-neighbor connected systems, performing these tasks using unitary processes from an initial product state is limited by the Lieb-Robinson bound [48, 49]. It takes a time proportional to the distance between two points to establish maximal quantum correlation between them. This is also a direct corollary of the fact that in a quantum circuit consisting of geometrically local gates, the light cone of any qubit grows linearly in the depth of the circuit. By examining these tasks on a range of different graphs, we hope to understand how the graph structure can affect the limitations on quantum processes caused by locality considerations. Prior work has characterized the difficulty of creating graph states [50], but preparation of such states is not limited by Lieb-Robinson considerations.

Our work in this chapter should be contrasted with work on entanglement percolation [51, 52]. Entanglement percolation describes the process of using low-quality entanglement between adjacent nodes on a graph to create one unit of long-range, high-quality entanglement (e.g., a Bell pair). The use of entanglement percolation to prepare large cluster states on a lattice was considered in Ref. [53]. The nature of entanglement growth in complex networks was considered in Refs. [54, 55], showing that so-called "scale-free" networks are particularly easy to produce large entangled states in. We are interested in the overall capability of different graph structures to perform large computations and in the use of graph eigenvalue methods to understand the spread of quantum information [56]. GHZ state preparation and state transfer are just two possible benchmark tasks, and it is possible that other tasks would result in different evaluations of relative performance between graphs.

Our work should also be considered in the context of classical network theory, where much is known about complicated graph structures [57–59]. It remains to be seen to what degree classical network theory can be easily exported to the quantum domain. Quantum effects such as the no-cloning theorem may limit our ability to distribute information, or conversely we can take advantage of teleportation to distribute quantum bandwidth in anticipation of it actually being needed. As further examples of how quantum and classical networks differ, it has been shown that entanglement swapping may be used to permit quantum networks to reshape themselves into interesting and useful topologies [60]. It has also been shown that, in general, the optimal strategy for entanglement generation in quantum networks can be difficult to calculate because many aspects of classical control theory do not apply [61].

The structure of this chapter is as follows. In Sec. 2.2, we will introduce a binary operation on graphs known as the hierarchical product, describe how it can be used to produce families of graphs we call hierarchies, and discuss the properties of these hierarchies. In Sec. 2.3, we will compare hierarchies to other families of graphs, examining how certain graph-theoretic quantities scale with the total number of included qubits. Readers who are not interested in graph theoretic details may wish to skip much of these first two sections. In Sec. 2.4, we will use analytic and numerical methods to examine how long is required to construct GHZ states spanning our graphs or to transfer states across them, using Lieb-Robinson bounds to connect graph-theoretic quantities to bounds on quantum computing performance. Finally, in Sec. 2.5, we will show how the unique structure of hierarchies allows for simple heuristics to map qubits in an algorithm into physical locations in hardware.

2.2 Hierarchical Products of Graphs

2.2.1 Background and Notation

One of the defining features of modularity in a network is the presence of clusters of nodes that are well-connected. Qualitatively, a modular network can be partitioned into such node clusters, or *modules*, that have a sparse interconnectivity. In quantum networking, it is believed that fully connected architectures will suffer greatly decreasing performance or increasing costs as the number of nodes becomes larger, and this motivates the search for alternative network designs. For instance, Ref. [8] estimates that a single module of trapped-ion qubits will likely contain no more than 10 to 100 ions, noting that the speed at which gates are possible becomes slower as the module is expanded. On the network scale, we might imagine a network of nodes over longer distances connected by quantum repeaters [62]. In such a network, establishing direct links between every possible pair of N nodes would require $\Theta(N^2)$ sets of quantum repeaters, a prohibitive cost as N becomes large.

The state of the art in quantum technologies, such as ion traps and superconducting qubits, is the ability to control a small number ($\approx 10 - 100$) of physical qubits using certain fixed sets of one- and two-qubit operations. Instead of increasing the size of these modules, one could instead build a network out of many small modules that are connected at a higher level in a sparse way, perhaps by optical communication links [8].

Our first goal will be to describe modular architectures in the language of graph theory. This will then allow us to quantify and compare their connectivity properties

against other network designs, notably the nearest-neighbor grid architecture.

Our detour into graph theory in this chapter serves two purposes. First, it will allow us to develop a rigorous way to construct families of graphs which we believe are promising quantum computing architectures. Second, we will later (beginning in Sec. 2.4) use these graph properties to connect directly to physical bounds on the generation of states with long-range quantum correlations; phrasing the properties of quantum architectures as graphs allows us to make a direct application of the Lieb-Robinson bound to these cases.

An unweighted graph $G = (V, E)$ is conventionally specified by a set of vertices V, and a set of edges between the vertices E, where an edge between distinct vertices i and j will be denoted by the pair (i, j). In this chapter, we use the terms "vertex" and "node" synonymously. The *order* of a graph is the total number of vertices in the graph, $|V|$. It will be useful for the purposes of this chapter to work with *weighted* graphs, where we specify a weight $w_{ij} \in \mathbb{R}$ for each pair of vertices $(i, j) \in V \times V$. Two vertices i and j are said to be *disconnected* if $w_{ij} = 0$, and connected by an edge with weight $w_{ij} \neq 0$ otherwise. Thus, unweighted graphs may be thought of as graphs with unit weight on every edge.

Finally, the graphs we consider here will be *simple*, meaning:

- The edges have no notion of direction. In other words, $w_{ij} = w_{ji}$ for all $i, j \in V$.
- There are no self-edges, i.e., $w_{ii} = 0$ for all $i \in V$.
- Any two vertices have at most one edge between them.

Henceforth, graphs will be simple and weighted, unless otherwise specified.

The information contained in a graph can be represented as a matrix known as the *adjacency matrix*, whose rows and columns are labeled by the vertices in V and whose entries hold edge weights. Thus, the adjacency matrix is an $n \times n$ matrix where $|V| = n$. The adjacency matrix A_G (or simply A for shorthand) for a graph G is given by

$$A_{ij} = \begin{cases} 0, & \text{if } i = j, \\ w_{ij}, & \text{if } i \neq j. \end{cases} \tag{2.1}$$

An important measure of local connectivity is given by the *valency* v_i of a node i, with $v_i = \sum_{j=1}^{n} w_{ij}$. For unweighted graphs, the valency of any node is simply the number of edges incident at that node, otherwise known as the *degree* of the node. We will also define the graph diameter, $\delta(G)$, as the maximization of the shortest distance between two nodes on the graph over all pairs of nodes.

Graphs may also be described by the *Laplacian*. The algebraic Laplacian L is given by

$$L_{ij} = \begin{cases} v_i, & \text{if } i = j, \\ -w_{ij}, & \text{if } i \neq j. \end{cases} \tag{2.2}$$

The algebraic Laplacian is closely related to the adjacency matrix, since we may write $L = \Delta - A$, where $\Delta = \text{diag}\,(v_1, \ldots, v_n)$ is the diagonal matrix of vertex valencies. The eigenvalues of the algebraic Laplacian give us bounds on various graph properties, as discussed further in Sec. 2.2.2.4.

Finally, we remark that the algebraic Laplacian should not be confused with the normalized Laplacian $\mathcal{L} = \Delta^{-\frac{1}{2}} L \Delta^{-\frac{1}{2}}$, which is frequently seen in the network theory

literature. The algebraic properties discussed in the next section (such as associativity of the hierarchical product) apply to the adjacency matrix as well as the algebraic Laplacian, but not to the normalized Laplacian.

2.2.2 Hierarchical Product

Here, we will define the hierarchical product and illustrate it with simple examples. For a fuller exposition, see Ref. [43], where the hierarchical product of graphs was introduced. Note that, in some contexts, the hierarchical product is also known as the rooted product [44].

Given a graph G, let $\mathbb{1}_G$ denote the identity matrix on $n = |V|$ vertices. We will denote by D_G an $n \times n$ diagonal matrix with 1 as the first entry and zero everywhere else. Note that there is no natural notion of order to graph vertices, so the choice of "first" vertex must be specified explicitly. Graphs with such a specified first vertex are called *rooted graphs* [63]. We write these matrices as

$$\mathbb{1} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & & & & \\ & 0 & & & \\ & & 0 & & \\ & & & \ddots & \\ & & & & 0 \end{pmatrix}. \tag{2.3}$$

Definition 2.2.1. Given graphs G and H, the *hierarchical product* $P = G \amalg H$ is the graph on vertices $V_P = V_G \times V_H$ and edges $E_P \subseteq V_P \times V_P$ specified by the adjacency

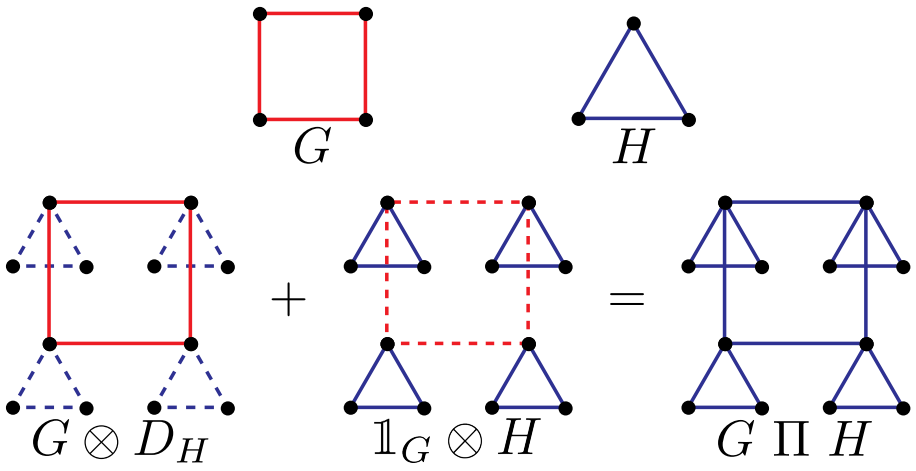


Figure 2.1: A simple example of the hierarchical product $G \Pi H$ between the cycle graphs $G = C_4$ and $H = C_3$. The first term in Eq. (2.4), $A_G \otimes D_H$, creates one copy of G on the vertex set formed by the first vertices of each H copy, while the second term $\mathbb{1}_G \otimes A_H$ creates the four copies of H.

matrix

$$A_P = A_G \otimes D_H + \mathbb{1}_G \otimes A_H, \tag{2.4}$$

or, equivalently, by the algebraic Laplacian

$$L_P = L_G \otimes D_H + \mathbb{1}_G \otimes L_H. \tag{2.5}$$

We will often use the shorthand $A_P = A_G \,\Pi\, A_H$ and $L_P = L_G \,\Pi\, L_H$.

If G and H are graphs, then $G \,\Pi\, H$ may be thought of as one copy of G with $|G|$ copies of H, each attached to a different vertex of G (see Fig. 2.1). Thus, $G \,\Pi\, H$ is a graph which has $|G|$ *modules* of $|H|$ nodes each. The modules' internal connectivity is described by H, and the modules are connected to one another in a manner described by G. The hierarchical product formalism therefore naturally produces modular graphs. Its main advantage comes from the convenience of working with the algebra at the level of adjacency matrices and Laplacians, which in turn makes the computation of important

properties of such graphs straightforward.

We now present some properties of the hierarchical product which make it an attractive formalism for practical applications in quantum networking.

2.2.2.1 Structural Properties

At the level of adjacency matrices, the hierarchical product is *associative*. Let A, B, C be three adjacency matrices. Then,

$$(A \,\Pi\, B) \,\Pi\, C = A \,\Pi\, (B \,\Pi\, C)\,. \tag{2.6}$$

For a proof, we refer the reader to Ref. [43].

Associativity implies that a product of multiple graphs does not depend on the order of evaluation. Therefore, we can unambiguously take the hierarchical product over many graphs to produce a graph of the form $G_k \,\Pi\, G_{k-1} \,\Pi \cdots \Pi\, G_1$. We will refer to such graphs as *hierarchies*, and the i-th graph in the product G_i as the i-th level of the hierarchy, enumerated from the bottom level upwards (symbolically, from right to left). In particular, if all G_i are equal to some graph G, then we write

$$G^{\Pi k} := \underbrace{G \,\Pi \cdots G \,\Pi}_{k-1 \text{ times}} G. \tag{2.7}$$

and refer to $G^{\Pi k}$ as a depth-k (or k-level) hierarchy.

Note that the hierarchical product *does not* satisfy many properties which are commonly assumed for operations on matrices. In particular,

1. Bilinearity: $(A_1 + A_2)\Pi B = A_1 \otimes D_B + A_2 \otimes D_B + \mathbb{1}_{(A_1+A_2)} \otimes B \neq A_1 \Pi B + A_2 \Pi B$. Similarly, $A \,\Pi\, (B_1 + B_2) \neq A \,\Pi\, B_1 + A \,\Pi\, B_2$.

2. Scalar multiplication: For any scalar α, $(\alpha A) \,\Pi\, B = \alpha A \otimes D_B + \mathbb{1}_A \otimes B \neq \alpha\,(A \,\Pi\, B) \neq A \,\Pi\, (\alpha B)$. Note however that scalar multiplication is distributive in the following way: $\alpha\,(A \,\Pi\, B) = (\alpha A) \,\Pi\, (\alpha B)$.

Hierarchical graphs are also special cases of hyperbolic graphs. The Gromov hyperbolicity [64], which measures curvature and is small for a graph with large negative curvature, is only a constant for hierarchical graphs. Since the hyperbolicity in general is at most half the graph diameter, whereas in this case it is independent of the diameter, it is termed *constantly hyperbolic* in the parlance of Ref. [65]. Hyperbolic graphs are seen in several real-world complex networks [66, 67], most notably the internet [68, 69]. Hyperbolic lattices have also been realized recently in superconducting circuits [70].

Finally, hierarchies have low tree-, clique- and rank-widths, which are each measures of the decomposibility of a graph [71]. These structural properties imply efficient algorithms for optimization problems expressible in monadic second-order (MSO) logic – a class which, for arbitrary graphs, includes several NP-hard problems. This feature could be useful in solving circuit layout and optimization problems on modular architectures without resorting to heuristics. We refer the reader to Ref. [72] for details on these structural results.

2.2.2.2 Scalability

So far we have discussed hierarchies in which the edges in different levels of the hierarchy are equally weighted. However, one useful generalization would be to allow the weight of edges at each layer of the hierarchy to vary. The meaning of this weight could vary depending on the context. In some cases, weights can be used to quantify the costs of an edge (*cost weight*). In others, we may wish to use weighted edges to quantify the power or performance of a network, interpreting edge weights as the strength of terms in a Hamiltonian or, inversely, the time required to communicate between nodes (*time weight*).

In this work, we prefer to remain agnostic to the meaning of the weights as much as is possible. When we calculate graph properties in Sec. 2.3, we will do so without reference to the meaning of the weights. In general, we will allow a graph to assign multiple kinds of weights to its edges, and each type of weight might scale differently. For now, we define a generalization of the hierarchical product which will allow us to construct hierarchies that incorporate different weights at different levels of the hierarchy.

Definition 2.2.2. Given graphs G and H, and $\alpha \in \mathbb{R}_+$, the *α-weighted hierarchical product* $P = G \,\Pi_\alpha\, H$ is a graph on vertices $V_P = V_G \times V_H$ and edges $E_P \subseteq V_P \times V_P$ specified by the adjacency matrix

$$A_P = \alpha A_G \otimes D_H + \mathbb{1}_G \otimes A_H, \tag{2.8}$$

or, equivalently, by the algebraic Laplacian

$$L_P = \alpha L_G \otimes D_H + \mathbb{1}_G \otimes L_H. \tag{2.9}$$

We will often use the shorthand $A_P = A_G \, \Pi_\alpha \, A_H$, and $L_P = L_G \, \Pi_\alpha \, L_H$.

As before, we may construct a k-level, *weighted* hierarchy out of k base graphs $G_1, \ldots, G_k$, and k weights $(\alpha_i, \ldots, \alpha_k) \equiv \boldsymbol{\alpha}$, so that the edges of the i-th level graph G_i are weighted by the i-th component of $\boldsymbol{\alpha}$, α_i. The adjacency matrix of such a hierarchy may be written as

$$A^{\Pi_{\boldsymbol{\alpha}} k} := \sum_{i=1}^{k} \alpha_i \mathbb{1}_{[i+1..k]} \otimes A_i \otimes D_{[1..i-1]}, \tag{2.10}$$

where the subscripts $[a \ .. \ b]$ on $\mathbb{1}$ and D are shorthand for the Kronecker product of matrices over all descending indices in the integer interval $[a \ .. \ b]$. For instance, $D_{[1..i-1]} := D_{G_{i-1}} \otimes D_{G_{i-2}} \otimes \cdots \otimes D_{G_1}$.

Defined as above, a weighted hierarchy $G^{\Pi_{\boldsymbol{\alpha}} k}$ is uniquely and efficiently specified by a real vector of weights $\boldsymbol{\alpha} \in {\mathbb{R}_+}^k$ and an ordered tuple of graphs $(G_1, \ldots, G_k)$. It will be the case that our analyses are unaffected by an overall scaling of the weight vector, so that one may identify $\boldsymbol{\alpha} \equiv c\boldsymbol{\alpha}$ for any real scalar c. As convention, we will always normalize by setting $\alpha_1 = 1$, which corresponds to assigning a unit-weight multiplicative factor to the lowest-level graphs in the hierarchy. Then, as stated previously, the edges at the i-th level of the hierarchy are weighted by α_i. We will shortly narrow our focus to weighted hierarchies where the weights follow a geometric progression, $\alpha_i = \alpha^{i-1}$.

We can construct the adjacency matrix of the graph $G^{\Pi_{\boldsymbol{\alpha}} k}$ by repeated application

of the two-fold product (Def. 2.2.2) in some well-defined way, analogous to Eq. (2.7). However, unlike before, the weighted product is non-associative, so we must first define an order of operations for manifold weighted products. Unless otherwise specified, we will always evaluate a manifold product from *right to left*, which corresponds to building the hierarchies from the bottom up, and is required in order to ensure that this definition matches Eq. (2.10). For example, in the 3-fold product $A_3 \,\Pi_{\alpha_3}\, A_2 \,\Pi_{\alpha_2}\, (\alpha_1 A_1)$, we will first evaluate the product $A_2 \,\Pi_{\alpha_2}\, (\alpha_1 A_1)$, and then take the product of A_3, weighted by α_3, with the resulting graph. The final result is

$$\alpha_3 A_3 \otimes D_2 \otimes D_1 + \alpha_2 \mathbb{1}_3 \otimes A_2 \otimes D_1 + \alpha_1 \mathbb{1}_3 \otimes \mathbb{1}_2 \otimes A_1. \tag{2.11}$$

In fact, a k-fold product, when evaluated this way, matches the right hand side of Eq. (2.10). Therefore, the k-level weighted hierarchy can also be written unambiguously as

$$A^{\Pi_{\boldsymbol{\alpha}} k} = A_k \,\Pi_{\alpha_k}\, A_{k-1} \,\Pi_{\alpha_{k-1}} \cdots \Pi_{\alpha_2}\, (\alpha_1 A_1). \tag{2.12}$$

Henceforth, the weight α_1, which scales the lowest-level adjacency matrix A_1, will be dropped due to our normalization choice of $\alpha_1 = 1$.

An important class of hierarchy graphs is one where the level weights follow a geometric progression of weights, i.e., $\alpha_i = \alpha^{i-1}$. We will denote such hierarchies by $G^{\Pi_\alpha k}$, where the scalar subscript α will be understood to mean the mutual weighting between successive hierarchies. For $\alpha > 1$, this leads to a "fat tree" structure, while for $\alpha < 1$, we instead get a "skinny tree" for which the edge weights decrease between

consecutive levels from the leaves to the root. These constructions are illustrated in Fig. 2.2, and mentioned because fat trees are known to be a commonly used architecture in classical networks [73].

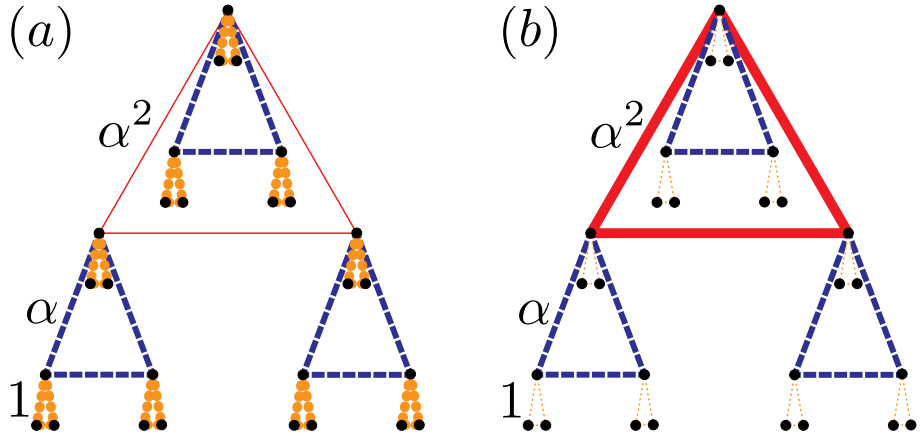


Figure 2.2: An illustration of the use of the hierarchical product to produce (a) "skinny" and (b) "fat" trees. In each case, the hierarchy $K_3^{\Pi_\alpha 3}$ is drawn, with the thickness of edges illustrating the weight of those edges. Depending on whether $\alpha < 1$ or $\alpha > 1$, this can lead to either lower-weighted high-level edges as in (a) or higher-weighted ones as in (b). Note that, for ease of visualization, here we break the usual convention of taking the lowest-level edges as unit weight.

Allowing a clear separation of the modular system into hierarchical levels, each of which can be assigned unique edge weight, enables straightforward discussion of computation that occurs both within and between modules in a unified framework. When two nodes interact, we can assign this a cost that depends on the edges between them.

2.2.2.3 Node Addressal

A hierarchy on N nodes gives a natural labeling of the nodes. Suppose the hierarchy H contains k levels and each level is described by a graph G with $|G| = n$ nodes, where $n^k = N$. Label the vertices of G by indices $j = 0, 1, \ldots, n-1$. Then, the adjacency matrix $\mathbb{1}_G \otimes G$ (which corresponds to n disjoint copies of G) has vertices which may be labeled as (jk), where $j, k = 0, 1, \ldots n-1$. The first label identifies

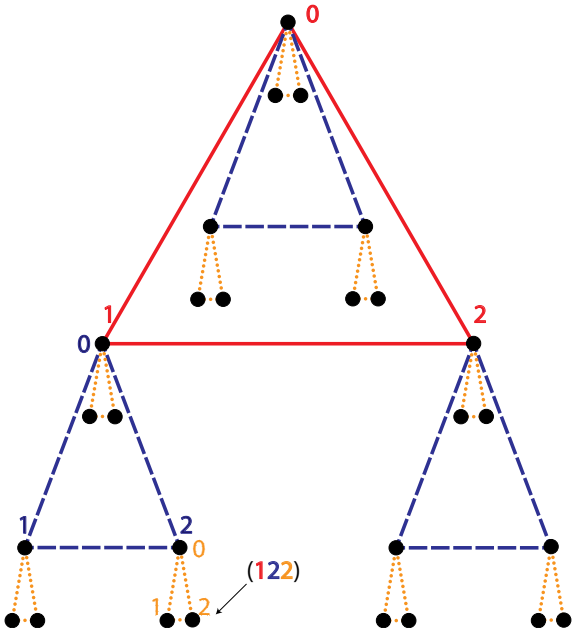


Figure 2.3: Addressing nodes in the hierarchy, layer by layer. Shown is a three-level hierarchy with the triangle graph K_3 as its base. Each vertex is represented as a 3-digit number in base 3. The first digit points to a node at the top level (red solid triangle), the second to a location in the second level (blue dashed triangle), and finally, the last digit (yellow dotted triangle) specifies the node location completely.

which copy of G the node occurs in, while the second identifies where in G it appears. The same vertex labeling can then be used for the 2-level hierarchy $G \,\Pi\, G$. In this manner, the k-level hierarchy has n^k vertices with labels of the form $(b_1 b_2 \cdots b_k)$, where $b_i \in \{0, 1, \ldots, n-1\}$ for all i. This is essentially a k-digit, base-n representation of numbers from 0 to $N = n^k - 1$, as illustrated in Fig. 2.3.

This node addressal scheme allows for each node to be uniquely identified in a way that simultaneously describes its connectivity to other nodes and allows for easy counting of how many nodes lie in either the entire graph or in particular subgraphs. This addressal scheme will be important for describing a variant of hierarchies in Sec. 2.2.2.5 and for implementing the graphs in software, e.g. as used to generate the numerical results in Sec. 2.4.3.

2.2.2.4 Spectral Properties

One of the tools frequently used in analyzing large networks is the spectral decomposition of the Laplacian. The behavior of the largest eigenvalue, the first eigenvalue gap, and the distribution of eigenvalues as a function of the network parameters are some of the diagnostics that can provide key information about dynamical processes on the network, and can also be used as points of comparison between competing network topologies [74].

The smallest eigenvalue of a Laplacian is always $\lambda_1 = 0$, which corresponds to the uniform eigenvector $\mathbf{e}_1 = (1, 1, \ldots, 1)$. In ascending order, the eigenvalues of L may be denoted by $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$. We now state some graph properties that can be related to the spectrum of L [74, 75].

The second eigenvalue λ_2 is known as the *algebraic connectivity* of the graph and is closely related to the expansion and connectivity properties of the graph. Broadly, the larger the value of λ_2, the better the connectivity of the network. To illustrate this point, consider the graph diameter, $\delta(H)$, which can be bounded using λ_2 as follows [74, Theorems 4.1.4, 4.1.10]:

$$\frac{4}{N\lambda_2} \leq \delta\left(H\right) \leq 2\left\lceil \frac{\Delta + \lambda_2}{4\lambda_2} \ln\left(N-1\right) \right\rceil, \tag{2.13}$$

where Δ is the maximum degree of H. It can be seen that a larger value for λ_2 will lead to a smaller graph diameter. We also have the following asymptotic bound on the mean

distance between nodes [75, Equations 6.13, 6.14], $\bar{\rho}(H)$:

$$\frac{2}{(N-1)\lambda_2(H)} + \frac{1}{2} \lesssim \bar{\rho}(H) \lesssim \left\lceil \frac{\Delta + \lambda_2}{4\lambda_2} \ln(N-1) \right\rceil. \tag{2.14}$$

Another important diagnostic of a network is given by the *Cheeger constant* $h(H)$ [76], also called the edge isoperimetric number or the graph conductance. This graph invariant is a measure of how difficult the graph is to disconnect by cutting edges, and is defined as follows:

Definition 2.2.3. For any node subset A of a graph $H = (V, E)$, let $\delta(A, \bar{A})$ denote the set of edges with exactly one node in A. Then, the Cheeger constant of H, denoted $h(H)$, is given by

$$h(H) = \min_{\substack{A \subset V \\ |A| \leq N/2}} \frac{|\delta(A, \bar{A})|}{|A|}. \tag{2.15}$$

For a connected graph, the Cheeger constant is always positive. As benchmark values, the complete graph K_N has Cheeger constant $N/2$ while a cycle graph C_N has Cheeger constant $4/N$. The relationship between λ_2 and $h(H)$ can be seen through the following bounds:

$$\frac{\lambda_2}{2} \leq h(H) \leq \sqrt{\lambda_2 (2\Delta - \lambda_2)}. \tag{2.16}$$

Many other graph properties may be derived from the Laplacian spectrum as well (see, e.g., Refs. [74, 75]).

For a large network, finding the eigenvalues can be numerically expensive. However, hierarchies have a special structure which can be exploited for the evaluation of graph spectra. Here, we show (in Theorem 2.2.1) that if the spectra of the base

graphs L_i are known, then one can derive the spectrum of the k-level hierarchy efficiently using a recursive procedure. We first present two lemmas. The first lemma generalizes Theorem 3.10 from Ref. [43], which states that the characteristic polynomial $\phi_P(x)$ $(= \det[x\mathbb{1} - P])$ of an unweighted hierarchical product of adjacency matrices A, B is given by

$$\phi_P(x) = \phi_{B'}(x)^{n_A} \phi_A\left(\frac{\phi_B(x)}{\phi_{B'}(x)}\right), \tag{2.17}$$

where A' (resp. B') is the matrix A (resp. B) with the first row and first column removed, and $n_A = |G_A|$ is the order of the graph A. In fact, Eq. (2.17) applies to Laplacians as well as adjacency matrices. The lemma below further generalizes this statement to a weighted product of Laplacians.

Lemma 2.2.1. Let K and L be two graph Laplacians with characteristic polynomials given by $\phi_K(x)$ and $\phi_L(x)$, respectively. Then, the characteristic polynomial $\phi_\Pi(x)$ of the hierarchical product $K\ \Pi_\alpha\ L$ is given by

$$\phi_\Pi(x) = [\alpha\phi_{L'}(x)]^{n_K} \phi_K\left(\frac{1}{\alpha}\frac{\phi_L(x)}{\phi_{L'}(x)}\right), \tag{2.18}$$

where $n_k = \dim\{K\}$, and L' is defined similar to A' and B' above.

Proof. Denote the spectra of K and L by $\{\kappa_j\}$ and $\{\lambda_j\}$, respectively. Recall that the α-weighted hierarchical product may be written as

$$K\ \Pi_\alpha\ L = \alpha K \otimes D_L + \mathbb{1}_K \otimes L. \tag{2.19}$$

If U_K is a unitary that diagonalizes K, we conjugate the above equation with the unitary $U_K \otimes \mathbb{1}_L$, and look at the resulting block matrix. Each block corresponds to an eigenvalue of K, and thus the j-th block is given by $\alpha\kappa_j D_L + L$. The full spectrum may then be expressed as a disjoint union of the block spectra,

$$\operatorname{spec}\left(K \,\Pi_\alpha\, L\right) = \bigsqcup_{j=1}^{|K|} \operatorname{spec}\left(\alpha\kappa_j D_L + L\right). \tag{2.20}$$

Now, we apply Eq. (2.17) to $K \,\Pi_\alpha\, L \equiv (\alpha K) \,\Pi\, L$ and use the fact that $\phi_{\alpha K}(x) = \det\left[x\mathbb{1} - \alpha K\right] = \alpha^{n_K} \det\left[\frac{x}{\alpha}\mathbb{1} - K\right] \equiv \alpha^{n_K}\phi_K\left(\frac{x}{\alpha}\right)$. This yields Eq. (2.18), as desired.

□

Now we show that if the eigenvalues of K and the polynomials ϕ_L and $\phi_{L'}$ are known, then there is a straightforward procedure to compute the eigenvalues of $K \,\Pi_\alpha\, L$.

Lemma 2.2.2. Let K and L be graph Laplacians, as before. Each eigenvalue of the product characteristic polynomial ϕ_Π can be found as a solution of the equation

$$\alpha\kappa_i = \frac{\phi_L\left(x\right)}{\phi_{L'}\left(x\right)} \tag{2.21}$$

for some K-eigenvalue κ_i.

Proof. Any eigenvalue of the product graph must be a zero of the left-hand side of Eq. (2.18) and, by equality, a zero of the right-hand side. Now, the degree of polynomial ϕ_K is n_K, which implies that the term of degree n_K must be nonzero. Thus, in the product $\phi_{L'}\left(x\right)^{n_K}\,\phi_K\left(\frac{1}{\alpha}\frac{\phi_L(x)}{\phi_{L'}(x)}\right)$, there must be a term which is *indivisible* by the polynomial

$\phi_{L'}(x)$. Therefore, the zero of the right-hand side cannot be a root of the polynomial $\phi_{L'}$.

We are seeking values of x such that the polynomial $\phi_K\left(\frac{1}{\alpha}\frac{\phi_L(x)}{\phi_{L'}(x)}\right)$ evaluates to zero. In other words, we are looking for x such that the term $\frac{1}{\alpha}\frac{\phi_L(x)}{\phi_{L'}(x)}$ is a root of ϕ_K. Therefore, we solve Eq. (2.21) for x, for all roots κ_i of K. □

If the forms of ϕ_L and $\phi_{L'}$ are known (and if each have sufficiently low degree), then computing the roots of ϕ_Π becomes tractable, even if K is a large matrix. This suggests a recursive procedure for computing the spectrum of a k-level hierarchy, by writing it as a product of the $(k-1)$-level hierarchy with the k-th base graph. We now frame this as our main result of this section:

Theorem 2.2.1. Suppose we have a k-level hierarchy $L^{\Pi_{\boldsymbol{\alpha}} k}$ described by base graph Laplacians $L_1, L_2, \ldots, L_k$ and weights $\boldsymbol{\alpha} = (1, \alpha_2, \ldots, \alpha_k)$ as follows,

$$L^{\Pi_{\boldsymbol{\alpha}} k} = L_k \,\Pi_{\alpha_k}\, L_{k-1}\, \Pi_{\alpha_{k-1}} \cdots \Pi_{\alpha_3}\, L_2\, \Pi_{\alpha_2}\, L_1. \tag{2.22}$$

Define a new set of weights $\boldsymbol{\beta} = (1, \beta_2, \ldots, \beta_k)$ with $\beta_i = \alpha_i/\alpha_{i-1}$, and a new set of Laplacians $M_k, M_{k-1}, \ldots, M_1$ recursively as

$$M_k = L_k,$$

$$M_i = M_{i+1}\, \Pi_{\beta_{i+1}}\, L_i.$$

Then, the following hold:

1. $M_1 = L^{\Pi_{\boldsymbol{\alpha}} k}$.

2. Any eigenvalue of M_i (for $i < k$) may be found as a solution to the equation

$$\beta_{i+1}\mu^{(i+1)} = \frac{\phi_{L_i}(x)}{\phi_{L_i'}(x)} \tag{2.23}$$

for some $\mu^{(i+1)} \in \text{spec}\,\{M_{i+1}\}$.

Proof. First, we prove statement 1. It can be seen that

$$\begin{aligned} M_{k-1} &= M_k \,\Pi_{\beta_k}\, L_{k-1} = L_k \,\Pi_{\beta_k}\, L_{k-1} \\ &= \frac{1}{\alpha_{k-1}} \left(\alpha_k L_k \otimes D_{k-1} + \alpha_{k-1} \mathbb{1}_k \otimes L_{k-1}\right), \end{aligned} \tag{2.24}$$

$$\begin{aligned} M_{k-2} &= M_{k-1} \,\Pi_{\beta_{k-1}}\, L_{k-2} \\ &= \frac{1}{\alpha_{k-2}} (\alpha_k L_k \otimes D_{k-1} \otimes D_{k-2} + \\ &\alpha_{k-1} \mathbb{1}_k \otimes L_{k-1} \otimes D_{k-2} + \alpha_{k-2} \mathbb{1}_{k-1} \otimes \mathbb{1}_{k-2} \otimes L_{k-2}), \end{aligned} \tag{2.25}$$

and so on, until we have an $\boldsymbol{\alpha}$-weighted sum over all k of the base graphs (with an overall denominator of $\alpha_1 = 1$), which is precisely $L^{\Pi_{\boldsymbol{\alpha}} k}$.

The proof of statement 2 follows as a direct consequence of Lemma 2.2.2, with $K = M_{i+1}, L = L_i$, and $\alpha = \beta_{i+1}$. □

Theorem 2.2.1 provides an algorithm to compute the spectrum of $L^{\Pi_{\boldsymbol{\alpha}} k}$, namely:

1. Compute the relative weight vector $\boldsymbol{\beta}$ from $\boldsymbol{\alpha}$.

2. Start with $i = k$, where the spectrum of $M_k = L_k$ is known. Decrease i by one.

3. Compute the spectrum of M_i from the known spectrum of M_{i+1} and Eq. (2.23). Decrease i by one.

4. Perform step 3 repeatedly, halting at $i = 0$. Return the spectrum of $M_1 = L^{\Pi_\alpha k}$.

Therefore, given a large hierarchy, one can efficiently compute the Laplacian eigenvalues and use them to find bounds on important graph properties. This is a scalable technique for obtaining figures of merit efficiently for hierarchies. Later, in Sec. 2.3, we will present analytic results for some of these figures of merit for simple hierarchies, but the results of the current section can be used even in more complicated cases, such as hierarchies that do not use the same G at every layer or that have heterogeneous scaling parameters.

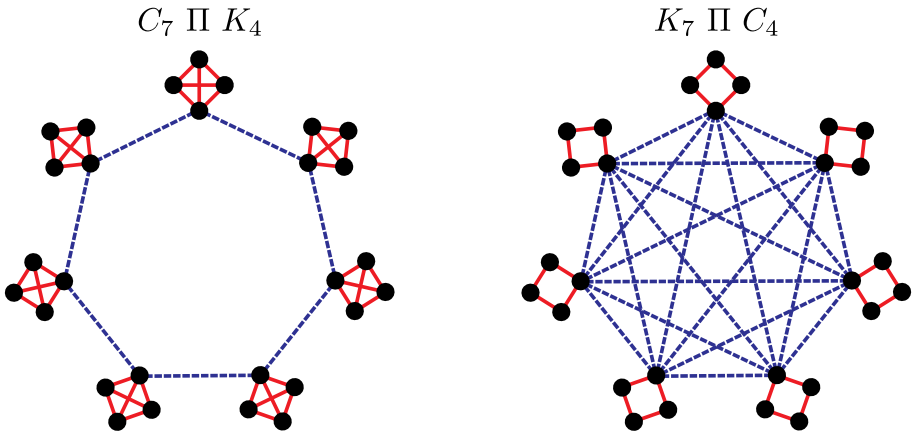


Figure 2.4: Two topologies with the same number of nodes (28) and edges (49). While the diameters for the two graphs are the same, are they equally well-connected? A comparison of the Cheeger constants (see Table 2.1) suggests that the left graph is less interconnected. This is consistent with the spectral gap, which is smaller for the left graph, indicating poorer connectivity.

Due to the structural richness and heterogeneity of graphs, it is not always easy to decide whether one graph is, for instance, more connected than another graph. One aspect of connectivity is how close the nodes are to one another, which is captured by quantities like the diameter and mean distance. In Fig. 2.4, we compare two graphs, $C_7 \ \Pi \ K_4$ and $K_7 \ \Pi \ C_4$, which have an identical number of nodes (28) and edges (49). The two graphs also have identical diameters (5 each), but the mean distance for the left graph is smaller (see Table 2.1). Under these measures, the left graph appears better connected.

Better connectivity also corresponds to having fewer bottlenecks in the graph, which corresponds to a larger Cheeger constant. In Fig. 2.4, the graph on the right has a larger Cheeger constant, as one would expect given that it has complete connectivity between the seven modules. Note that this metric of connectivity need not agree with the mean distance, as seen in this example.

Similarly, a parameter-by-parameter comparison of the two hierarchy graphs C_{13} Π K_5 and K_{13} Π C_5 (Table 2.1) reveals that, while both graphs are two-level hierarchies with the same number of nodes and edges, K_{13} Π C_5 has the smaller diameter, smaller mean distance, larger cheeger constant, and a larger spectral gap, all of which indicate better connectivity. While structural comparisions for the above examples can be carried out simply by inspection or a quick calculation of graph quantities, general hierarchies may be far too complex to compare this way. In practice, when choosing a modular topology with the best connectivity, one might hope for a single, balanced measure of connectivity that relates to aspects such as node distance and bottleneckedness and is easy to compute. The spectral gap λ_2 meets these requirements. It is asymptotically related to the other invariants discussed here via upper and lower bounds in Eqs. (2.13), (2.14) and (2.16). Furthermore, λ_2 can be efficiently computed using the recursive procedure described earlier in this section.

2.2.2.5 Truncated Hierarchical Product

In some scenarios, there may be physical or technological limitations on the total number of interconnections allowed at a single node of a quantum computer. In our

Graph Invariant	$C_7 \,\Pi\, K_4$ vs.	$K_7 \,\Pi\, C_4$	$C_{13} \,\Pi\, K_5$ vs.	$K_{13} \,\Pi\, C_5$
Number of edges	49	49	143	143
Number of nodes	28	28	65	65
Diameter	<u>5</u>	<u>5</u>	8	<u>5</u>
Mean distance	<u>2.68</u>	2.71	4.77	<u>3.23</u>
Cheeger constant	0.17	<u>1.0</u>	0.07	<u>1.4</u>
Spectral gap λ_2	0.16	<u>0.46</u>	0.04	<u>0.34</u>

Table 2.1: Comparison of topologies by connectivity measure. In each case, the graphs being compared have an identical number of nodes and edges. The better value for each comparison is underlined.

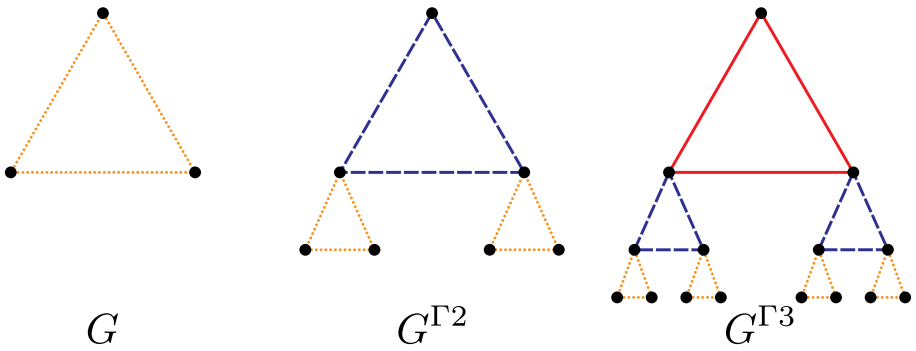


Figure 2.5: A demonstration of how our hierarchical product can be truncated to avoid requiring many interconnections at one node. As the hierarchy grows, the graph is duplicated and then attached to a subset of nodes in a larger version of the base graph, G.

framework, this manifests as a restriction on the maximum degree of a node. We believe that hierarchical structures can still prove useful in this context, but (as we will see in Sec. 2.3) the hierarchy we have described thus far has a maximum degree which grows linearly with the number of levels of the hierarchy.

We now introduce an architecture which maintains the hierarchical properties but also has a bounded maximum node degree (i.e. maximum node degree that does not go to infinity as the number of levels goes to infinity). To model such an architecture, we modify the hierarchical product $G_1 \,\Pi\, G_2$. Whereas previously, $|G_1|$ copies of G_2 were connected according to G_1, we now bring together $|G_1| - 1$ copies, which we connect according to G_1, and add the root node of G_1 without an associated subhierarchy (see Fig. 2.5). When extended to a many-level hierarchy, this means that every node will be connected to, at most, two levels, and so its degree will not grow as the hierarchy grows. We will denote this *truncated* hierarchical product by $G_1 \,\Gamma\, G_2$, and its weighted version as $G_1 \,\Gamma_\alpha\, G_2$. It can be written algebraically in terms of adjacency matrices by adopting a more general definition of the hierarchical product.

Definition 2.2.4. Given rooted graphs G and H, the *weighted truncated hierarchical product* $P = G \,\Gamma_\alpha\, H$ is a graph on vertices $V_P = V_G \times V_H$ and edges $E_P \subseteq V_P \times V_P$ specified by the adjacency matrix

$$A_P = \alpha A_G \otimes D_H + P_G \otimes A_H, \tag{2.26}$$

or, equivalently, the algebraic Laplacian

$$L_P = \alpha L_G \otimes D_H + P_G \otimes L_H. \tag{2.27}$$

Here, P_G is a projector onto all nodes in G except the root node. At the level of adjacency matrices, we may also write $A_P = A_G \, \Gamma_\alpha \, A_H$. An unweighted version, $G \, \Gamma \, H$, can be obtained by setting $\alpha = 1$.

An illustration of this architecture can be found in Fig. 2.5. From this definition, we naturally derive both unweighted and weighted truncated hierarchies, $G^{\Gamma k}$ and $G^{\Gamma_\alpha k}$. We note that a generalization of this definition to allow an arbitrary projector (rather than one that only excludes the root node) is possible, but we do not consider such a case in this chapter.

The addressing scheme outlined in Sec. 2.2.2.3 can also be used for truncated hierarchies. However, since many nodes do not sit atop sub-hierarchies in this case, not all node addresses are valid. We will assume that the node in the i-th level which connects to the level above it has a zero in the i-th digit of its address. In a truncated hierarchy, each node whose address contains a zero (representing the "root" of a hierarchy) must have only zeros in all following positions, as it does not contain any further sub-hierarchies. The base-n addressal scheme can thus be used to specify which nodes are present in a truncated hierarchy.

Note that the truncated hierarchical product adds nodes more slowly than (although with the same scaling as) the hierarchical product structure specified at the beginning of Sec. 2.2.2. When we perform graph comparisons in Sec. 2.3, we will consider all

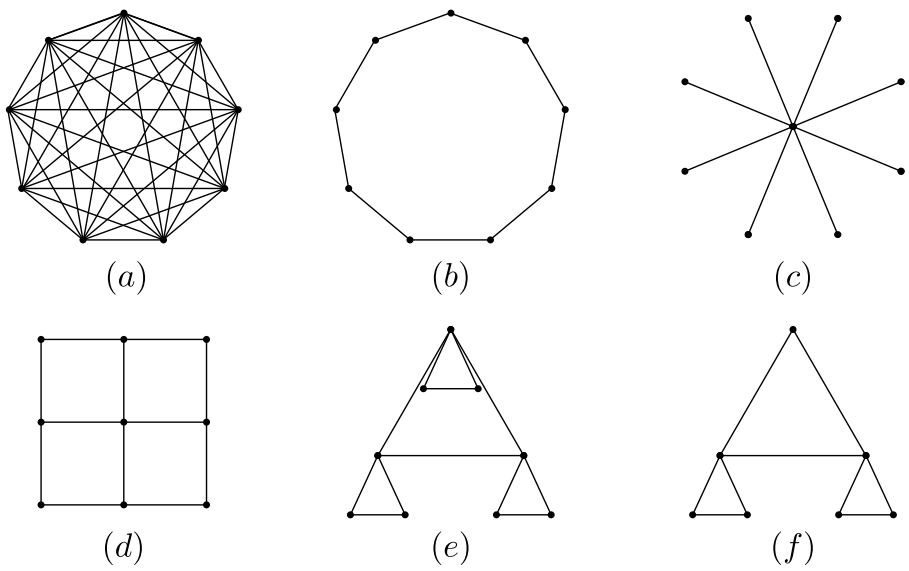


Figure 2.6: Illustration of the graph structures considered in this section, each with nine nodes except (f). (a) The complete graph K_9. (b) The cycle graph C_9. (c) The star graph S_9. (d) The nearest-neighbor grid in two dimensions. (e) The hierarchical product $K_3^{\Pi 2}$. (f) The truncated hierarchical product of Sec. 2.2.2.5, $K_3^{\Gamma 2}$.

cost functions and optimizations in terms of the total number of nodes so that the two architectures can be compared fairly.

2.3 Graph Comparisons

Having developed the machinery to construct hierarchies, we will now evaluate them against other potential architectures. Any evaluation is impossible to do in an absolute sense, since what properties are desirable in a graph and how serious the cost of improving them is will depend on both the application as well as the physical system under consideration. In general, we assume that the most desirable quality of a graph is some measure of connectivity or the ease with which the graph can transport information between nodes. Note that it is always possible to translate between quantum circuit architectures with some overhead. A detailed atlas summarizing these overheads can

be found in Ref. [77].

We will look at the scenario of state transfer, which is an important subroutine that may need to be carried out if an algorithm requires gates to be performed between two qubits that are not directly connected. We consider the worst-case state transfer time on a given graph, which allows us to evaluate graphs without reference to any particular quantum algorithm. If we are interested in the time taken for state transfer in the graph, an appropriate metric can be the diameter of the graph, $\delta(H)$, under the assumption that information transfer takes unit time along any edge in the graph. The diameter then captures the maximum distance, and hence the maximum time required for information to travel between any two nodes in the system.

For graphs produced by the weighted hierarchical product, we will also consider a diameter which takes into account edge weight. This "weighted diameter," $\delta_w(H)$, can be found by considering all pairs of nodes j, k and identifying the two whose least-weighted connecting path has the highest sum weight of edges. If we consider a path between two nodes j and k to be a set of nodes $P = \{j, v_1, v_2 \dots v_n, k\}$ with a weight $W(P)$ given by the sum $w_{j,v_1} + w_{v_1,v_2} + \cdots + w_{v_n,k}$, then the weighted diameter can be written as:

$$\delta_w(H) = \max_{j,k} \min_{P} W(P). \tag{2.28}$$

One way to grasp why the weighted diameter is a useful quantity is to consider the time weights of edges, where the weight signifies the time required to perform a gate between two connected qubits. In this case, the weighted diameter is the maximum time it will take us to perform a chain of two-qubit gates that connects two different qubits (for instance,

using SWAP operations to bring the two qubits to adjacent positions and then performing the final desired operation).

However, optimizing *only* with respect to connectivity yields a trivial result, because a fully connected graph is obviously most capable of communicating information between any two points. Therefore, we will consider a number of different possible "costs" associated with physical implementations of graphs. One potential input to the cost function is the maximum degree of a graph, $\Delta(H)$. As discussed in the previous section, we want to avoid needing to connect too many different communication channels to a single node. Another is total edge weight $w(H)$ – if it costs time, energy, money, coherence, or effort to produce communication between two nodes, we should try to use as few communication channels as possible.

We now walk through the calculations for several important graph quantities for several graphs: an all-to-all connected graph, a cycle graph, a star graph, a square grid, a hierarchy graph with scaling parameter α, and a truncated version of that same hierarchy graph. We calculate how quantities scale with the total number of nodes N. For ease of calculation, we assume that N nodes fit in the architecture of the current graph; for instance, we assume $N = \ell^d$ for some integer ℓ for a d-dimensional square graph. All results of this section are compiled in Table 2.2, and examples of the graphs for small N are illustrated in Fig. 2.6.

Graph H	δ	δ_w	Δ	$w(H)$
K_N	const.	const.	N	N^2
S_N	const.	const.	N	N
C_N	N	N	const.	N
Square grid, d-dim	$dN^{1/d}$	$dN^{1/d}$	d	dN
$K_n^{\Pi_\alpha k}, \alpha \neq n$	$\log_n N$	$\max\left(\frac{2}{1-\alpha}, N^{\log_n \alpha}\right)$	$n \log_n N$	$nN^{\max(1,\log_n \alpha)}$
$K_n^{\Pi_\alpha k}, \alpha = n$	$\log_n N$	$\max\left(\frac{2}{1-\alpha}, N^{\log_n \alpha}\right)$	$n \log_n N$	$nN \log_n N$
$K_{n+1}^{\Gamma_\alpha k}, \alpha \neq n$	$\log_n N$	$\max\left(\frac{2}{1-\alpha}, N^{\log_n \alpha}\right)$	n	$nN^{\max(1,\log_n \alpha)}$
$K_{n+1}^{\Gamma_\alpha k}, \alpha = n$	$\log_n N$	$\max\left(\frac{2}{1-\alpha}, N^{\log_n \alpha}\right)$	n	$nN \log_n N$

Table 2.2: Summary of scalings of important graph properties with total node number, N. For every graph H, the columns contain (from left to right): diameter δ, weighted diameter δ_w, maximum degree Δ, and total edge weight $w(H)$. All entries describe only the scaling of the leading coefficient with d, n, and N.

2.3.1 Graph Calculations

2.3.1.1 Complete Graph, K_N

Since all nodes in a complete graph [Fig. 2.6(a)] have edges between them, the diameter is simply 1. This comes at the cost of very high maximum degree, $N-1$, as every node is connected to all $N-1$ other nodes. The total weight of every edge is the same, and there are $N(N-1)/2$ edges because every pair of nodes has a corresponding edge. Therefore, the total edge weight scales as $\Theta(N^2)$.

2.3.1.2 Cycle Graph, C_N

In a cycle graph [Fig. 2.6(b)], the diameter is $\lfloor N/2 \rfloor$, the distance to the opposite side of the circle. The maximum degree is only 2, and the total weight of the edges is likewise only N. This graph is thus able to reduce the cost factors associated with the complete graph, but at the cost of a much higher asymptotic diameter.

2.3.1.3 Star Graph, S_N

The star graph is the graph which has a single central node connected to all others [Fig. 2.6(c)]. Like the complete graph, it also has a constant diameter, although this diameter is two rather than one. The maximum degree of the star graph is $N-1$, the same as the complete graph. However, the star graph improves over the complete graph, as it has a lower total edge weight of $N-1$ rather than $\binom{N}{2}$. Thus, we have improved the cost asymptotically without affecting the overall scaling of the diameter of the graph.

The example of S_N raises a complication which we do not attempt to quantify in this chapter. In a realistic distributed quantum computer, we expect that a significant amount of operations need to be performed at the same time and need to be scheduled on the graph. But in the star graph, all operations between nodes must pass through the single central hub. This is likely to lead to a scheduling bottleneck when performing general quantum algorithms. While we do not attempt to treat scheduling of such algorithms on the network in this chapter, in future work we hope to consider these complications, which will at times make the star graph unsuitable for real-world use. An experimental comparison of the star graph and the complete graph in existing five-qubit quantum computers can be found in Ref. [78]. In those experiments, the requirement that all information be shuttled through a central node for the S_N connectivity made high-fidelity execution of quantum algorithms more difficult.

2.3.1.4 Square Grid Graph

We consider now a square grid (i.e., a hypercubic lattice) in d dimensions [Fig. 2.6(d)]. Here, the diameter is $d(N^{1/d} - 1)$, since this is the distance from the point in one corner labeled $(1, 1, 1, \dots)$ to the opposite corner at $(N^{1/d}, N^{1/d}, \dots)$ (note that diagonal moves are not allowed). The maximum degree depends on the dimension, as each interior node is connected to $2d$ other nodes. The total edge weight can be found by considering that each node on the interior of the graph corresponds with exactly d edges, and it is these edges that dominate as $N \to \infty$. Therefore, the total edge weight scales as $\Theta(dN)$.

2.3.1.5 Hierarchy Graph, $G^{\Pi_\alpha k}$

As the hierarchy graph [Fig. 2.6(e)] is built recursively, it is easiest to calculate its properties using recursion relations. We consider a graph that has k levels to it, so that given a base graph G and $n = |G|$, then the overall graph has n^k nodes.

First, we calculate the unweighted diameter of a k-level hierarchy, which we denote by $\delta\left(G^{\Pi_\alpha k}\right)$. Since all sub-hierarchies are rooted at their first vertex, we will need to keep track of the *eccentricity* of the root node, which we denote by $\varepsilon(F)$ for any subhierarchy F. The eccentricity of any graph node is defined as the maximum distance from that node to any other node in the graph F. Here, we fix $\varepsilon(F)$ to be the root eccentricity for the graph in question.

Now, we write recursion relations for two quantities, the unweighted diameter $\delta(G^{\Pi_\alpha i})$ of an i-level hierarchy for some intermediate i, and the eccentricity $\varepsilon(G^{\Pi_\alpha i})$

of the top-level root node of the current i-level hierarchy.

Consider a diametric path in an i-level hierarchy. This path must ascend and descend the entire hierarchy. That is, using the notation of Sec. 2.2.2.3, two maximally separated qubits have addresses that are different in their first digit. Such a path can always be partitioned into 3 disjoint pieces, the terminal two of which each lie in some $(i-1)$-level subhierarchy, while the middle piece lies in the current top (i.e. i-th) level. These three pieces must be independently maximal, since the path is diametric. The middle piece maximizes to the diameter of the top-level graph, which is simply $\delta(G)$. The two sub-level pieces each maximize to the root eccentricity of the $(i-1)$-th level subhierarchy, which is precisely the quantity $\varepsilon(G^{\Pi_\alpha(i-1)})$. Therefore, our first recursion reads

$$\delta(G^{\Pi_\alpha i}) = 2\varepsilon(G^{\Pi_\alpha(i-1)}) + \delta(G). \tag{2.29}$$

The i-th level root eccentricity may be found by a similar argument. Partition the most eccentric path (starting at the top level root node) into two pieces, one which lies at the top level, and the other which lies exclusively in the lower levels. Maximizing both pieces, one gets

$$\varepsilon(G^{\Pi_\alpha i}) = \varepsilon(G^{\Pi_\alpha(i-1)}) + \varepsilon(G). \tag{2.30}$$

Solving the second relation, we get $\varepsilon(G^{\Pi_\alpha i}) = i\varepsilon(G)$. By substitution, the first recursion has the solution

$$\delta(G^{\Pi_\alpha k}) = 2(k-1)\varepsilon(G) + \delta(G). \tag{2.31}$$

Since the total number of levels is given by $k = \log_n N$, and the graph diameter is no

greater than twice the eccentricity of any node, we conclude that the diameter scales as $\Theta(\varepsilon(G)\log_n N)$ for a general graph G. If we specifically examine the case when G is a complete graph of order n, $\delta(G) = 1$ and $\varepsilon(G) = 1$, and the exact expression is $\delta\left(G^{\Pi\boldsymbol{\alpha}k}\right) = 2\log_n(N) - 1$.

Next we calculate the maximum degree. Again, we proceed by recursion. Iterating the hierarchical product to some level i can be viewed as attaching a copy of the graph $G^{\Pi\boldsymbol{\alpha}(i-1)}$ to every point in the graph G. Therefore, the degree of every root node in the $(i-1)$-level subhierarchies increases by the degree of the corresponding node in graph G. The maximal increase achievable thus is the maximum degree $\Delta(G)$ of graph G. Since the root node for an i-level subhierarchy has i distinct copies of G attached to it, its degree is given by $i \cdot \deg(g_1)$, where g_1 is the root node of G. Then, the i-level maximum degree can be expressed as

$$\Delta(G^{\Pi\boldsymbol{\alpha}i}) = \max\left\{(i-1)\deg(g_1) + \Delta(G), \Delta(G^{\Pi\boldsymbol{\alpha}(i-1)})\right\} \tag{2.32}$$

$$\ldots = \max_{0\le j\le i-1}\left\{j\deg(g_1) + \Delta(G)\right\} \tag{2.33}$$

$$= (i-1)\deg(g_1) + \Delta(G), \tag{2.34}$$

where the second step was obtained by recursion. For a general G, this gives the maximum degree scaling as $\Delta(G^{\Pi\boldsymbol{\alpha}k}) = \Theta(\log_n N)$. For $K_n^{\Pi\boldsymbol{\alpha}k}$, the root degree and the maximum degree of the base graph K_n are both $n-1$, so $\Delta(K_n^{\Pi\boldsymbol{\alpha}k}) = (n-1)\log_n N$.

Now we consider the total edge weight of the hierarchy. We compute this by a recursion relation, first by duplicating the existing edge weight at $i-1$ levels by n (the number of smaller hierarchies we must bring together) and then adding new edges. If the

edges at level i have weight α_i, we can write this as:

$$w(G^{\Pi_{\boldsymbol{\alpha}} i}) = nw(G^{\Pi_{\boldsymbol{\alpha}}(i-1)}) + \alpha_i w(G). \tag{2.35}$$

By counting the number of subhierarchies with different weights, we find the following form for the total edge weight of the weighted hierarchy:

$$w\left(G^{\Pi_{\boldsymbol{\alpha}} k}\right) = w(G) \sum_{i=1}^{k} \alpha_i \left|G\right|^{k-i}. \tag{2.36}$$

This can be verified by checking that it satisfies the recursion relation Eq. (2.35). If we now specialize to the case where $G = K_n$ and $\alpha_i = \alpha^{i-1}$, we find

$$w\left(K_n^{\Pi_\alpha k}\right) = \frac{n(n-1)}{2} \sum_{i=1}^{k} \alpha^{i-1} n^{k-i}. \tag{2.37}$$

This behavior can be broken into three regimes. For $\alpha = n$, the sum is constant, and the overall scaling is $\Theta(nN \log_n N)$. Otherwise, we can perform the geometric sum to obtain

$$w\left(K_n^{\Pi_\alpha k}\right) = \frac{n(n-1)}{2} \frac{n^k - \alpha^k}{n - \alpha}. \tag{2.38}$$

Here, the scaling will depend on the relative size of n and α. For $n > \alpha$, the first term in the numerator dominates, and $w\left(K_n^{\Pi_\alpha k}\right) = \Theta(nN)$. Otherwise, we can write $\alpha^k = N^{\log_n \alpha}$ and find $w\left(K_n^{\Pi_\alpha k}\right) = \Theta(nN^{\log_n \alpha})$.

Finally, we calculate the weighted diameter of a k-level hierarchy $\delta_w(G^{\Pi_{\boldsymbol{\alpha}} k})$, just as for the unweighted diameter, by solving recursion relations for the quantities

$\delta_w(G^{\Pi_{\boldsymbol{\alpha}} i})$ and $\varepsilon_w(G^{\Pi_{\boldsymbol{\alpha}} i})$, which are, respectively, the weighted diameter and weighted root eccentricity for an i-level weighted hierarchy. Here, note that the top level (at any intermediate stage i) is weighted by α_i. Therefore, the recursion for the weighted diameter is modified to

$$\delta_w(G^{\Pi_{\boldsymbol{\alpha}} i}) = 2\varepsilon_w(G^{\Pi_{\boldsymbol{\alpha}}(i-1)}) + \alpha_i \delta_w(G). \tag{2.39}$$

Similarly, the recursion for the weighted eccentricity becomes

$$\varepsilon_w(G^{\Pi_{\boldsymbol{\alpha}} i}) = \varepsilon_w(G^{\Pi_{\boldsymbol{\alpha}}(i-1)}) + \alpha_i \varepsilon_w(G), \tag{2.40}$$

which has the solution $\varepsilon_w(G^{\Pi_{\boldsymbol{\alpha}} i}) = \varepsilon_w(G) \sum_{j=1}^{i} \alpha_j$. Finally, we have

$$\delta_w(G^{\Pi_{\boldsymbol{\alpha}} k}) = 2\varepsilon_w(G) \sum_{j=1}^{k-1} \alpha_j + \delta_w(G)\alpha_k. \tag{2.41}$$

For $G = K_n$ and $\alpha_i = \alpha^{i-1}$, this becomes:

$$\delta_w(K_n^{\Pi_\alpha k}) = 2 \sum_{i=1}^{k-1} \alpha^{i-1} + \alpha^{k-1} \tag{2.42}$$

$$= \frac{\alpha^k + \alpha^{k-1} - 2}{\alpha - 1}. \tag{2.43}$$

Therefore, the scaling of the weighted diameter with N has two regimes, depending on α. For $\alpha < 1$ the geometric sum converges as $i \to \infty$ to $\frac{2}{1-\alpha}$. This means that for $\alpha < 1$, a constant time suffices to traverse the entire hierarchy no matter how large it is. For $\alpha = 1$ the weighted diameter is equal to the (unweighted) diameter, which we have already computed. For $\alpha > 1$, δ_w scales as $\alpha^{k-1} = N^{\log_n \alpha}/\alpha \sim N^{\log_n \alpha}$. Note that the

last scaling only applies if α does not scale with n. Since $n > 1$ and $\alpha > 1$, this exponent $\log_n \alpha$ is always positive. Therefore, the total edge weight is asymptotically always either constant (for $\alpha < 1$) or growing (for $\alpha \geq 1$), as expected.

2.3.1.6 Truncated Hierarchy, $G^{\Gamma_\alpha k}$

Finally, we look at how the results above are modified if we use the truncated hierarchical product discussed in Sec. 2.2.2.5 [Fig. 2.6(f)]. Although many of the calculations in terms of the number of levels k are similar to those for the non-truncated hierarchy, it is no longer the case that $k = \log_n N$ exactly. In order to compare graphs fairly, we will need to recalculate the order of $G^{\Gamma_\alpha k}$ so that results in this section can be written in terms of the total number of nodes, N.

Under the node addressal scheme of Sec. 2.2.2.3, the nodes of a truncated hierarchy are in one-to-one correspondence with base-n strings of length k that only have trailing zeros. As before, a 0 label points to a root node, but since root nodes do not bear subhierarchies due to truncation, all subsequent labels are forced to be 0. In other words, we only label nodes using strings of the form $(l_1 l_2 \dots l_i 00 \dots 0)$ for some $i \leq k$, and $l_j \neq 0$ for all $j \leq i$. The number of such strings with i nonzero labels followed by $(k-i)$ zero labels is $(n-1)^i$. Therefore, the total number of nodes is

$$N = \sum_{i=0}^{k} (n-1)^i. \tag{2.44}$$

Since $N = \Theta\left((n-1)^k\right)$, many quantities of a truncated hierarchy with a base graph of order $n+1$ have the same scaling with the number of nodes N as those for a non-truncated

hierarchy with a base graph of order n.

In terms of the number of levels k, the maximum diameter will be proportional to k, just as it was in Sec. 2.3.1.5. It follows that the diameter scales with the total number of nodes as $\delta = \Theta\left(\log_{n-1} N\right)$ for a truncated hierarchy.

On the other hand, truncation offers a large improvement in the maximum degree of the hierarchy. As discussed in Sec. 2.2.2.5, the maximum degree of the truncated hierarchy is $\Delta(G^{\Gamma_{\boldsymbol{\alpha}} k}) = 2\Delta(G)$, which is constant in N.

The edge weight recursion relation is simply $n-1$ copies of the current graph and then new, additional edges:

$$w(G^{\Gamma_{\boldsymbol{\alpha}} i}) = (n-1)w(G^{\Gamma_{\boldsymbol{\alpha}}(i-1)}) + \alpha_i w(G). \tag{2.45}$$

This is identical to the recursion relation for the standard hierarchy, Eq. (2.35), except that there are now only $n-1$ copies, and also, for a given number of qubits N, the number of levels k may be different by constant factors and terms. Thus, the only modification to the recursion relation is to replace n with $n-1$, and the solution of the relation is otherwise identical. This means that none of the asymptotic scaling with k is affected, and the scaling with N is only affected by changing the total number of levels required to construct a graph of N nodes.

The recursion relation for weighted diameter is similar to Eq. (2.39). Due to truncation, one needs to make a careful comparison of paths that do or do not terminate at the root node of the top level, but in any case the weighted diameter's scaling with k is the same as the non-truncated weighted diameter's scaling. The weighted diameter scaling

with N can thus be found from Eq. (2.43), using the appropriate value of k for truncated hierarchies with N nodes.

2.3.2 Choosing Among Graphs

2.3.2.1 Graph Embeddings

The long list of comparisons summarized in Table 2.2 can make it difficult to see exactly when different graphs are preferable. To make our calculations more concrete, we would like to compare concrete scenarios for the connection of qubits arranged on a grid in d dimensions. Specifically, in each dimension ($d = 1$, 2, and 3), we examine a hierarchy that is embedded into the grid, comparing its properties to the same grid but with nearest-neighbor connections. We consider building modules where each small module is a complete graph of size n, laid out in cubes on the grid so that the side-length of the cube is $n^{1/d}$. The $d = 1$ and $d = 2$ cases with $n = 2^d$ are illustrated in Fig. 2.7.

As shown, the length of an edge must increase by a factor of $n^{1/d}$ (2 in Fig. 2.7) at every level of the hierarchy in order to make these hierarchies possible. Therefore, to determine the total length of wire used, we can use a cost weight with $\alpha = n^{1/d}$. Keeping factors of N only, Table 2.2 shows that for $d = 1$, we expect a total cost weight $\Theta\left(N \log_n N\right)$, while for the higher-dimensional cases we expect a total cost weight $\Theta\left(N\right)$ [1]. For the d-dimensional grid, this total cost weight is always $\Theta(N)$.

Now, to consider the performance of the two graphs, we must fix a separate scaling factor for the time weight, β. There are several options which might be reasonable

[1] If, for the application at hand, a planar graph is required, cycles such as C_n can yield the same scaling.

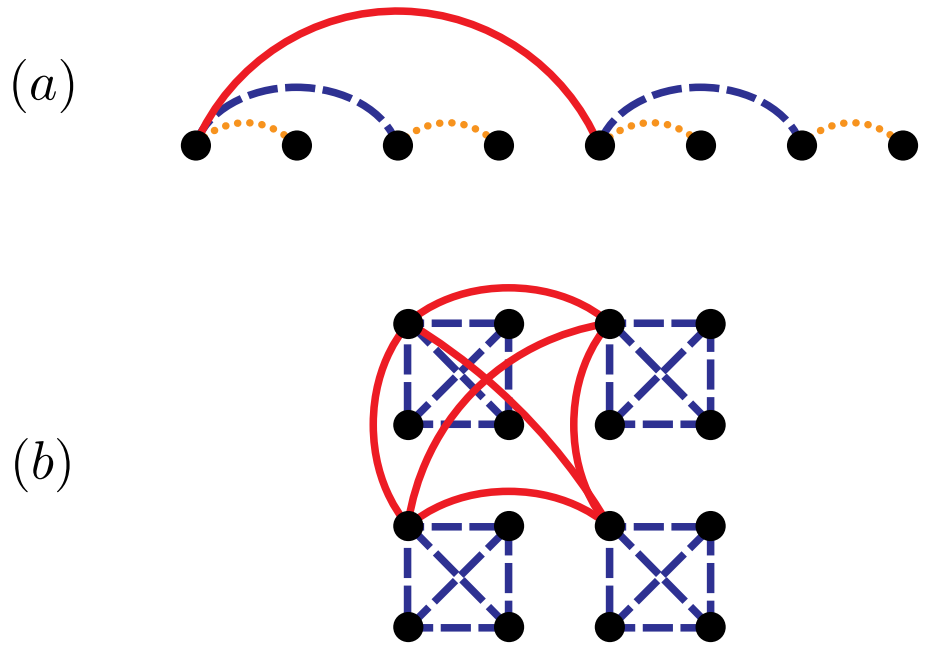


Figure 2.7: An illustration of the embedding of a hierarchy on a (a) one- or (b) two-dimensional lattice of qubits. In both cases, the length of an edge doubles at every level of the hierarchy, but the scaling in total edge length used changes from $\Theta(N \log_2 N)$ to $\Theta(N)$ when going from 1 to 2 dimensions. In $d = 3$, a similar hierarchy with doubling length scales connects modules of eight qubits.

for different physical applications. If $\beta = 1$, i.e., all links act identically in terms of time required to traverse them, then the weighted diameter of the hierarchy is simply $\Theta(\log_n N)$. Another option would be to take $\beta = \alpha$, i.e., to assume that links take as long to move through as they are long. In this case, we find that the hierarchy's weighted diameter scales as $\Theta\left(N^{1/d}\right)$, meaning that the hierarchy and nearest-neighbor graphs match in performance.

We may also want to allow hierarchies to make use of the "fat tree" concept to produce a better-performing graph [73]. Suppose that we allow ourselves to "spend more" on higher-level links, causing their cost weight to increase with a factor α, but improving their performance so that the time weight scales with the factor $\beta = 1/\alpha$. In this case, the question is what range of α allows for the hierarchy to perform better than the nearest-

neighbor grid (lower time-weighted diameter) for less cost (lower total edge cost weight)? (Note that this cost weight includes any contribution from "lengthening" wires at hire levels of the hierarchy.)

To answer the first, we compare the two asymptotic diameter scalings, $N^{\max(0,\log_n 1/\alpha)}$ and $N^{1/d}$. This suggests that if $\alpha \geq n^{-1/d}$, the hierarchy will allow for faster traversal than the nearest-neighbor grid. However, we wish to avoid causing the hierarchy to have a total cost weight that scales worse than $\Omega(N)$, which requires $\log_n \alpha < 1$. We find that a winning hierarchy can be constructed if α lies in the range $\alpha \in \left[n^{-1/d}, n\right)$. The optimal α is as large as possible but less than n; at that point an additional logarithmic factor is introduced to the total cost weight scaling.

In these cases, we have not allowed the nearest-neighbor grid to modify the weight (either kind) of its links. This is because any modification in its cost or time weight enters simply as a constant factor; if the individual links have weight c instead of 1, the overall weighted diameter is just $cN^{1/d}$ while the total cost weight is just cN. Of course, one can apply different constants to each figure of merit, or apply c to one and $1/c$ to the other. In order to make the nearest-neighbor grid match the performance of the hierarchy, the unit-length time weight would have to be $N^{\log_n(\alpha)-1/d}$ while the unit-length cost weight must not scale with N.

2.3.2.2 Pareto Efficiency

Our calculation of various graph parameters suggests that the hierarchy architecture offers significant advantages over others. One way to make this comparison more exact

is to appeal to the economics concept of Pareto efficiency, which is used to designate an acceptable set of choices in multiparameter optimization [79]. A choice is Pareto efficient if switching to a different choice will cause at least one parameter to become worse. Suppose we eliminate all constants to focus only on the scaling with N for three parameters: weighted diameter, maximum degree, and total edge weight. By removing these constants, we assume that the small multiplicative factors they provide will not influence decision making. For simplicity, we will assume that both cost and time weights scale with the same factor, α.

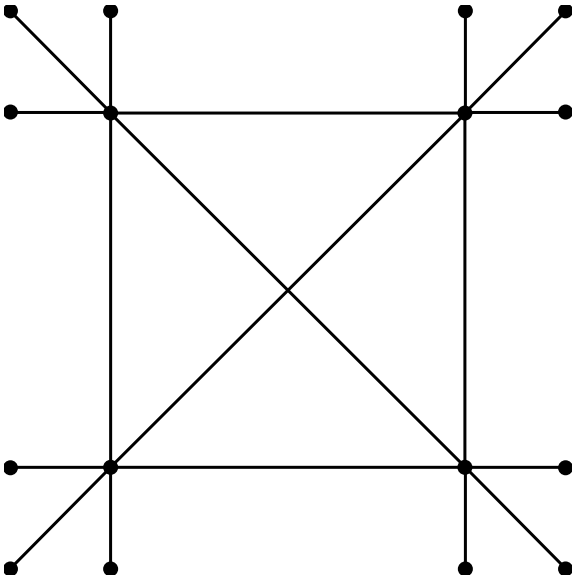


Figure 2.8: An example of a porcupine graph as defined in Ref. [80], in this case, $K_4 \amalg S_4$.

For comparison, one could ask: what minimum number of edges is required for a graph on N nodes to have maximum degree Δ and diameter δ? Reference [80] answers this optimization question partially, and constructs what are known as *porcupine graphs* which achieve the optimum, illustrated in Fig. 2.8. We observe here that qualitatively, porcupines are modular, since they may be described by attaching trees to the nodes of a complete graph. In particular, the graph $K_{\sqrt{N}} \amalg S_{\sqrt{N}}$ is a porcupine graph that achieves a

Graph	δ_w	Δ	w
K_N	const.	N	N^2
S_N	const.	N	N
C_N	N	const.	N
Square grid	$N^{1/d}$	const.	N
$\star$ $K_{\sqrt{N}} \amalg S_{\sqrt{N}}$	const.	$\sqrt{N}$	N
$\star$ $K_{n+1}^{\Gamma_\alpha k}$ $\alpha \neq 1$	$N^{\log_n \alpha}$	const.	N
$\star$ $K_{n+1}^{\Gamma_\alpha k}$ $\alpha = 1$	$\log_n N$	const.	N

Table 2.3: An illustration of the scaling with N of three key parameters to be used in Pareto optimization. Here δ_w is the weighted diameter, Δ is the maximum degree, and w is the total edge weight of the graph. A star ($\star$) has been placed next to the two graphs we find to be Pareto efficient. We have also included the $\alpha = 1$ (unweighted) hierarchy in the final row, as it has a different scaling for the weighted diameter. Our Pareto efficiency judgment is made assuming $n^{1/d} \geq \alpha \geq 1$.

diameter $\delta = 3$ and a maximum degree of $\Delta = 2(\sqrt{N} - 1)$ with the minimal number of edges.

We summarize the scalings of these graphs in Table 2.3. Assume that $n^{1/d} \geq \alpha \geq 1$. In this case, we can find the Pareto-efficient solutions by noting which options can be eliminated. We see that K_N is strictly worse than S_N and can be eliminated; S_N is then dominated by the porcupine. C_N is dominated by the square grid, which has identical scaling of total weight and degree but lower diameter. The square grid, in turn, is dominated by the hierarchy due to the assumptions we have made on α. This means that the two Pareto-efficient choices in this case are the truncated hierarchy and the porcupine graph. If we chose any option besides these two, we could improve the scaling with respect to N without any trade-off by switching to one of them. While this framework does not offer a decision rule to choose between the porcupine and $K_n^{\Gamma_\alpha k}$, the latter is clearly preferable if our aim is to create a modular quantum system that does not rely on a few centralized nodes. We stress that this optimization procedure is only intended to evaluate the quantities and graphs introduced, and the Pareto-efficient choices will change

if other figures of merit or other graphs are included in the optimization.

2.3.2.3 Optimality of diameter for hierarchical graphs

The use of $K_n^{\Gamma_\alpha k}$ may be further motivated via the degree-diameter problem [81] (for a survey, see Ref. [82]). Given a graph with a maximum allowed degree Δ on each node and diameter no greater than δ, the degree-diameter problem asks for the maximum number of nodes $N(\Delta, \delta)$ that such a network could hold. This problem is practically well-motivated in the design of networks, and may be answered for special classes of graphs. The Moore bound, which is a bound for general graphs, states that the number of nodes N is at most $\frac{\Delta(\Delta-1)^\delta-2}{\Delta-2}$. This means that for a constant maximum degree $\Delta \geq 3$, the diameter satisfies $\delta = \Omega(\log N)$, meaning that hierarchical graphs have optimal diameter up to a constant factor. Tighter bounds on the number of nodes may be shown, for instance, when the *tree-width* of the graph is bounded. Ref. [83] shows that graphs with small tree-widths t and an odd diameter δ satisfy

$$N(\Delta, \delta; t) \sim t(\Delta - 1)^{\frac{\delta-1}{2}}. \tag{2.46}$$

As discussed towards the end of Sec. 2.2.2.1, hierarchies have low tree-widths. In particular, the tree-width of the truncated hierarchy $K_n^{\Gamma_\alpha k}$ is at most $n-1$. Next, the diameter of the truncated hierarchy $K_n^{\Gamma_\alpha k}$ is $\delta(k) = 2k-1$ (which is odd), and the maximum degree is $\Delta(k) = 2(n-1)$. Comparing the number of nodes in this hierarchy

$N(k)$ to the node capacity $N\left(\Delta(k), \delta(k); n-1\right)$ as in Eq. (2.46), we get

$$\frac{N(k)}{N\left(\Delta(k), \delta(k); n-1\right)} \gtrsim \frac{n^k}{(n-1)\left(2n-3\right)^{k-1}}. \tag{2.47}$$

Keeping the total number of nodes N fixed, consider two limits: one, a shallow hierarchy in which the number of levels k is $O(1)$, and two, a deep hierarchy, in which the size n of the base graph is $O(1)$ [i.e., $k = O(\log N)$]. We see that when the hierarchy is shallow, the right side of Eq. (2.47) is $\Theta(1)$, which indicates optimality. For a deep hierarchy, the above ratio scales as $2^{-\log_n N} = N^{\frac{-1}{\log(n)}}$, which is polynomially suboptimal. However, when $n = 3$, the ratio in Eq. (2.47) is again $\Theta(1)$, and the truncated hierarchy $K_3^{\Gamma_\alpha k}$ is degree-diameter optimal in this case.

2.4 Entangled State Construction

2.4.1 Setup

Although some of the graph properties calculated in the previous section give a heuristic sense for the capabilities of the hierarchical graph versus the nearest-neighbor or all-to-all graphs, we would like to examine their performance directly in terms of a quantum information processing task. The task we have chosen as a benchmark is the creation of a many-qubit GHZ state. Since this entangled state is difficult to create across long distances when using nearest-neighbor interactions, we hope that it can serve as a useful yet basic benchmark for processing quantum information with unitary evolution [47]. As shown in Ref. [47], preparation of a GHZ state also provides a means

of transferring a state across the graph. Thus, the results of this section also bound state transfer time. However, in this work, unlike Ref. [47], we focus on the use of discrete unitary operations (gates) rather than Hamiltonian interactions. This means that we cannot take advantage of the many-body interference which provided a speed-up in Ref. [47].

Using GHZ state creation as a benchmark for potential quantum architectures allows us to use physical limitations (represented by the Lieb-Robinson bound) to place computational limits on information processing. The GHZ state is directly useful on its own [45–47], but even in systems which do not directly produce the GHZ state, it is likely that quantum operations will require the creation of long-range correlations between distant sites. For example, the same physical bounds which govern the creation of the GHZ state also restrict the speed at which topological order can be produced [48]. We focus on the GHZ state as an easy-to-analyze example for the problem of creating these nonlocal correlations, but we stress that our results generalize to any state which possesses non-local correlations of the kind whose creation is limited by the Lieb-Robinson bound.

We adopt a framework in which every vertex of the graph represents one logical qubit, while an edge of the graph represents the ability to perform a two-qubit gate between nodes. For the purposes of this work, we assume that we can ignore single-qubit operations, instead focusing on the cost imposed by the required two-qubit gates between nodes.

2.4.2 Analytical Results for Deterministic Entanglement Generation

In order to create the GHZ state, we assume that we begin with all qubits in the state $|0\rangle$ except for one qubit that we place in the initial state $|+\rangle$. By performing controlled-NOT operations between this qubit and its neighbors, a GHZ state of those qubits is created. The state can be expanded by continuing to use further CNOT operations to expand the "bubble" of nodes contained in the GHZ state until it eventually spans the entire graph. For state transfer, we instead assume the initial state $|\psi\rangle$ to be transferred sits on one qubit, which is then transferred through the graph using SWAP operations until it reaches its destination.

We first consider a graph which has been assigned time weights, so that a gate between two linked edges can be performed deterministically in a time given by the weight of the edge between them. We assume that one node can act as the control qubit for several CNOT operations at once. Therefore, according to our protocol above, the time t_{GHZ} required to construct the GHZ state is found by identifying the qubit that will take the longest to reach from the initial qubit by hopping on the graph. A similar argument holds for the state transfer time.

This implies that a GHZ state can be created, or a state transferred, in time that scales like the (time-)weighted eccentricity of the node we choose as the initial $|0\rangle + |1\rangle$ state. However, if we take the further step in our analysis of maximizing over weighted eccentricities (identifying the worst-case starting node), then the time will simply be the weighted diameter of the graph as calculated in the previous section. Note that the difference between the best-case weighted eccentricity (the weighted graph radius) and

the worst-case weighted eccentricity (the weighted graph diameter) over all nodes is at most a factor of two – if we look at the midpoint of the path that realizes the graph diameter, its distance to the endpoints of the path is bounded by the radius – so from the perspective of how this time scales asymptotically with N, the two are interchangeable.

2.4.3 Numerical Results for Probabilistic Entanglement Generation

As shown in the previous subsection, in a deterministic setting of entanglement generation where a gate between two nodes of our graph H can be performed in fixed time, the time required to create a GHZ state is equal to the weighted diameter $\delta_T(H)$. However, in many situations in long-distance quantum information processing, probabilistic or heralded methods might be used instead. We might suppose that, in a small time step, the network succeeds in performing a desired two-qubit gate with probability p (and that we know whether the gate succeeded or not). Upon failure, one can try performing the gate again in the next time step without having to rebuild the state from the beginning. In this setting, we expect that the scaling will likely be similar to the deterministic case but more difficult to calculate exactly. Fortunately, it is easy to re-interpret the meanings of the edge weights to account for this.

The main complication arising from the inclusion of unitaries that do not get completed in a fixed amount of time is that multiple paths between two nodes can all contribute to the total probability that entanglement has been produced, making it a harder problem to solve exactly. However, we can turn to numerical simulation to get an idea of the behavior. In the following, we define a new edge weight called the probability weight,

p_{ij}, which is the probability of success of edge (i, j) in one time step.

The algorithm for simulating the creation of a GHZ state is as follows:

- At each time step t, identify the subgraph F of nodes that have already joined the GHZ state.
- For each edge between a GHZ node $i \in F$ and a non-GHZ node $j \notin F$, identify the probability edge weight p_{ij}. With probability p_{ij}, allow node j to join the GHZ state in the current time step, t.
- Once all edges have been tested, repeat the procedure for the next time step on the new, possibly larger, set of GHZ nodes.

A single number p_0 is chosen as the base probability, so that the probability weights on the lowest level are p_0, and edges on the i-th level of the hierarchy succeed with probability $p_0\alpha^{i-1}$. Note that we must fix $\alpha < 1$. As a first step toward evaluating the performance of a graph, we estimate its time weights as $w_{ij} = 1/p_{ij}$, the time required to perform a two-qubit unitary on average. The overall estimate of the expected time taken is then δ_T/p_0, where δ_T is the time taken for the deterministic case with time weights scaling by a factor $\beta = 1/\alpha$ at each level. We find that this predicts very well the rate at which the GHZ state can be constructed over a wide range of α values (Fig. 2.9). The expected time remains $\Theta\left(N^{\log_n(1/\alpha)}\right)$.

For graphs with multiple potential paths between two nodes, such as a two-dimensional grid, the expected time is not simply the deterministic time scaled by the extra time factor the probabilistic setup requires in each step. We can however still bound the expected time to build the GHZ state $\mathbb{E}[t_{\mathrm{GHZ}}]$ above and below for a graph H. We

will bound it above by considering a modified graph in which the only path between the initial qubit and the qubit farthest from the starting point has distance $d_w(H)$. Such a path completes in time $d_w(H)/p_0$ on average. Since H has strictly more paths than this, the expected time will be lower. However, the shortest path between the initial and final qubits has total distance $d_w(H)$, which would take time $d_w(H)$ to complete even if $p_0 = 1$ and all gates were deterministic. Therefore, no path can finish faster than this, and the expected outcome over all possible paths cannot improve over $d_w(H)$. We can therefore write the following restriction on the expected time:

$$d_w(H) \leq \mathbb{E}\left[t_{\mathrm{GHZ}}\right] \leq \frac{d_w(H)}{p_0}, \tag{2.48}$$

where $\mathbb{E}[\cdot]$ denotes the expected value. This implies $\mathbb{E}\left[t_{\mathrm{GHZ}}\right] = \Theta\left(d_w(H)\right)$. Therefore, although the prefactor is difficult to calculate, we can tell that the time required to complete the creation of a GHZ state on the nearest-neighbor graph with $d = 2$ is $\Theta(\sqrt{N})$. This scaling implies that the condition for the hierarchy to outperform the nearest-neighbor grid in 2D is $\alpha \geq n^{-1/2}$, which is reflected in Fig. 2.9.

Using the GHZ-creation time and state transfer as examples, we can see many of the advantages of hierarchical graphs as network topologies. Such architectures are able to rapidly incorporate a very large number of qubits (exponential in the number of hierarchy levels), while the time-weighted diameter (and thus communication time) grows linearly with the number of levels. Since the weighted diameter is not substantially changed even if we use the truncated hierarchical product of Sec. 2.2.2.5, these benefits can also be realized in that setup.

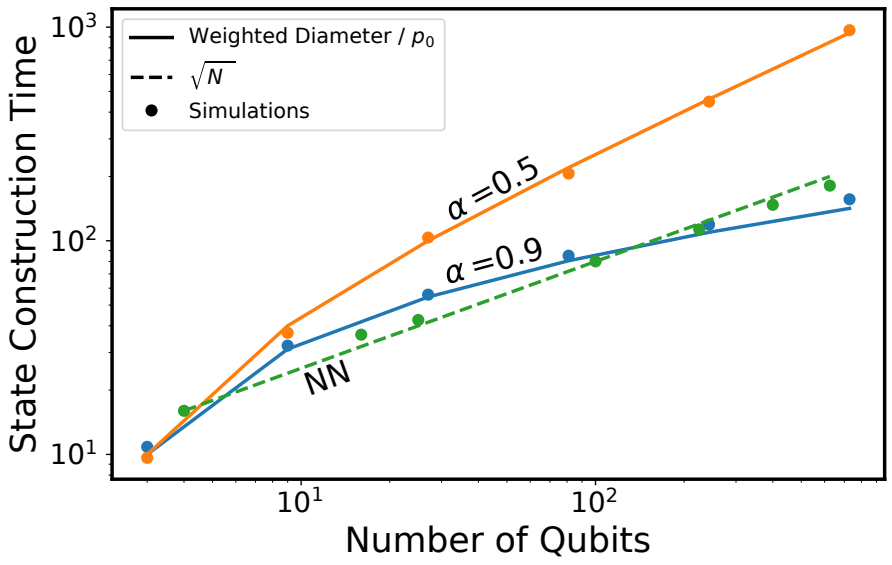


Figure 2.9: Graph-theoretic predictions and simulation of t_{GHZ} for the hierarchy $K_3^{\Pi_\alpha k}$ at various α, and a two-dimensional nearest-neighbor (NN) grid; $p_0 = 0.1$. The $\sqrt{N}$ fit shows the scaling of t_{GHZ} for the nearest-neighbor case, with a prefactor in the range suggested by the text's argument. Note that since $n = 3$, the crossover for the hierarchy to asymptotically outperform the nearest-neighbor grid is at $\alpha \geq 1/\sqrt{3} \approx 0.58$, which is seen in the numerical results. Code for generating this figure can be found at [84].

2.5 Circuit Placement on Hierarchies

A final reason we believe hierarchies could be a useful way to organize modular quantum systems is that they may be able to take advantage of straightforward methods for circuit placement. Circuit placement is a problem that arises when a quantum circuit or algorithm must be translated onto a physical system [85]. Suppose we are given a specification for a quantum algorithm in the form of a circuit diagram, and we wish to run that algorithm on a given quantum computer (which presumably has enough quantum memory to perform that algorithm). In order to translate the circuit into instructions for our machine, we must identify each algorithm qubit with a machine qubit and then

determine how the individual quantum gates can be realized in our machine [2].

Circuit placement is an important part of the quantum software stack, just as the compilation to machine code is in classical computers. By placing qubits which must operate on each other often close together in the real-world machine, we can minimize the amount of time spent performing long-range quantum gates. However, this problem is generally quite difficult for arbitrary instances and in fact has been shown to be NP-complete [85].

However, since we are interested in the sub-problem of circuit placement on hierarchies, it is possible that the hardness results of Ref. [85] do not apply and the exact solution can be found in polynomial time, just as the problem can be solved tractably in linear qubit chains [86]. Whether or not an exact algorithm exists, we can appeal to heuristics to efficiently place circuits as well as possible. Such an approach is promising because hierarchies are extremely structured with clear prioritization of clustering between small groups of qubits, which can be recognized in the algorithm and matched to the physical architecture.

To explain further, we consider the following model. We suppose that we begin with a weighted circuit graph C with a vertex set V_C and an edge set E_C, in which an edge exists between two vertices if there is at least one two-qubit gate between them in the circuit, with the weight of the edge corresponding to the number of gates. We then specify a machine graph, M, with vertex set V_M and edge set E_M, in which each edge (u, v) indicates that the machine can perform two-qubit gates between u and v.

[2] We studiously avoid referring to the machine qubits as "physical" in this chapter, as we do not want to confuse this conceptual distinction with the physical/logical qubit divide in error correction. All of the qubits referenced in this section are logical qubits in the error-correcting sense.

We now seek a mapping $f : V_C \rightarrow V_M$ that assigns algorithm qubits to machine qubits. A mapping f has a total cost found by considering, for every edge in E_C between vertices c_i and c_j, the shortest-path distance between $f(c_i)$ and $f(c_j)$ in M, multiplying that distance by the weight of the edge in C and summing over all edges. Thus, it captures the total distance that must be traversed by all gates in order to execute the circuit when the current mapping is used. Reducing this is expected to reduce the amount of time spent performing SWAP gates in order to connect two distant qubits. Performing this mapping is an important subroutine in any quantum programming framework, and at least one existing quantum compiler has a "mapper" phase that takes into account the actual graph that a program must be compiled onto [87, 88].

Our cost function is a choice made from convenience, and others are possible. Using this cost function ignores several important aspects of quantum circuits. First, our cost function does not account for the fact that a different mapping might allow for more parallelism, since it evaluates the cost of each gate individually. In addition, we take the circuit graph C as a given, when in fact many different circuits exist for any given quantum operation. In fact, it is likely that optimization of C could be performed, possibly by using the structure of M itself. A more realistic model for circuit placement may require a back-and-forth in which a circuit is first placed, then optimized, then re-placed, and so forth. A more advanced placement algorithm may even permit the swapping of qubits throughout the circuit, thus optimizing the placement of the quantum algorithm without constructing a circuit connectivity graph as an intermediate step.

For this chapter, we will ignore these concerns and proceed with a heuristic approach to circuit placement for hierarchies. We describe our algorithm as "partition and

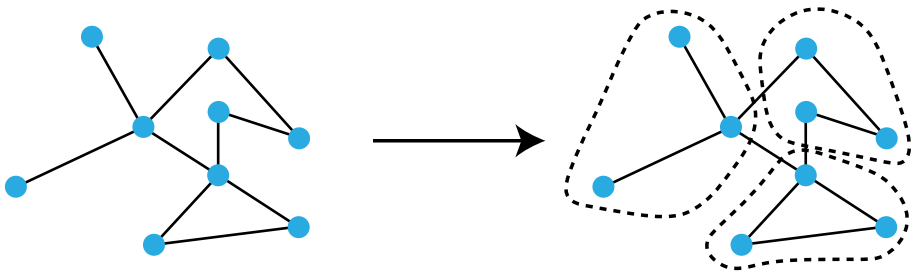


Figure 2.10: Illustration of how we might divide a hypothetical graph into smaller clusters. This process is repeated many times, recursively.

rotate," as it requires these two basic subroutines. First, qubits are partitioned into sub-hierarchies by examining whether they are connected by many gates in C. This process continues recursively, with each partition being subdivided and so on until every qubit is identified with its point in the hierarchy. This top-down process is then followed by a bottom-up process in which each small cluster is rotated so that its most-communicative qubit is at the root of the sub-hierarchy, and then the partitions themselves are rotated, and then clusters of clusters, etc. Ideally, this results in a mapping in which every qubit is (a) placed close to qubits it needs to communicate with and (b) placed in easy access to other modules if that qubit requires such access. We will now explore in detail these subroutines and the circuit speed-ups that result. We will place algorithms on a machine graph M which we take to be defined by $K_n^{\Pi k}$ for some integer k. Note that we examine unweighted hierarchies, but these methods can be applied to weighted hierachies as well.

2.5.1 Partitioning

For the first step of our algorithm, we wish to divide the computational graph C into n subgraphs which are as disconnected as possible. In addition, since we wish to assign each node in C to physically separate and limited qubit registers, it is important

that each of the subsets has precisely $|C|/n$ nodes. This problem is known as *balanced graph partitioning*, and the problem of finding the optimal solution is NP-complete for $n \geq 3$ [89]. However, heuristic methods exist which approximate the solution, and are widely used in the field of parallel computing and circuit design [90]. We have illustrated this process in Fig. 2.10.

Our method for performing circuit placement on hierarchies relies on a subroutine that performs balanced graph partitioning. There are many algorithms and software packages from which to choose. Here, we have used a software package called Metis, which implements an algorithm called recursive bipartitioning [90].

We begin by supposing that we have the circuit graph C and we wish to identify groups of $|C|/n$ nodes which have high connection to each other but low connection outside of the group. This is accomplished by finding a balanced graph partition in which the weight of the edges connecting each group is minimized. If we call the initial set of all nodes S, then we wish to identify subsets $S_0, S_1, \ldots, S_n$. In terms of the addressal scheme of Sec. 2.2.2.3, all the nodes in set S_i will have have digit i in their base-n representation. In the next section, we will discuss the choice of which digit to assign to each set.

Once the subsets S_i are found, partitioning can be run again on that relevant subgraph, creating n new subsets of this subset. Eventually, every node in the graph will be identified with a lowest-level module of size n, a next-level module of size n^2, and so forth.

Here we have used a generalized, pre-existing algorithm for graph partitioning. It is possible that the specifics of this problem, and the possibility of co-designing the precise quantum circuit implementing the algorithm (and thus C) with the architecture, enable

Figure 2.11: An illustration of how and why the process of rotation works in our circuit placement algorithm. In this diagram, red links represent gates to be performed (edges in C) and black ones are available communicative links (edges in M). In the graph C, the qubits 1, 2, and 3 are all connected, and 3 is connected with 4. These qubits have been correctly placed into clusters $(1, 2, 3)$ and (4). However, if they are not rotated correctly (see left), the link between 3 and 4 can become quite long, necessitating a long-range quantum gate. By properly rotating (right), the gate between links 3 and 4 becomes much shorter, improving the placement.

more specific, better-performing approaches.

2.5.2 Rotation

Drawing partitions between qubits is not enough to fully specify their placement into a hierarchy. If we consider using the i-digit representation, we can imagine that partitioning essentially describes the process of deciding, from a set of qubits, which ones will share a digit in the next level. However, these digits are more than arbitrary markers, because there is one node in any sub-hierarchy which connects to the hierarchy above. This node (which we say has digit 0) has privileged access to communication with other sub-hierarchies. Therefore, in order for our circuit placement to succeed, we should ensure that the qubit on top of each sub-hierarchy is the one which requires the most access.

In order to do this, we implement a second subroutine, the "rotate" part of the algorithm. This is called rotation because, once we know which qubits will be together

in a module, we must choose how to orient them relative to the larger modular structure. Whereas partitioning is top-down (the full graph is broken into small subgraphs which are then themselves partitioned), rotation is bottom-up. Suppose the modular structure is $K_n^{\Pi k}$. We begin with sets of n qubits and must choose which will be the top of each smallest instance of K_n. We then take each partition of n instances of K_n and decide which instance of K_n will connect to the next level up, and so on. This process is illustrated in Fig. 2.11.

Note that the general structure of our algorithm is to first go down the hierarchy, partitioning nodes, and then to go up, re-arranging sub-hierarchies in the proper order. We perform this procedure only once to obtain our circuit mapping.

2.5.3 Results

Now that the placement algorithm is specified, we turn toward examining its performance on quantum circuits. We consider two separate questions. First, we investigate whether the algorithm is effective – does it actually reduce, relative to a random assignment, the amount of distance that must be traversed in a circuit to execute all the requested gates? Second, we will examine whether the algorithm executes efficiently on a classical computer. This second point is important because in general the problem can be solved by brute-force search, but such a search requires a time $\mathcal{O}(N!)$ to perform (although, as we stated earlier, it is possible that an exact algorithm exists with a lower time cost for the special case of hierarchies).

To investigate the above concerns, we examine the algorithm's performance on

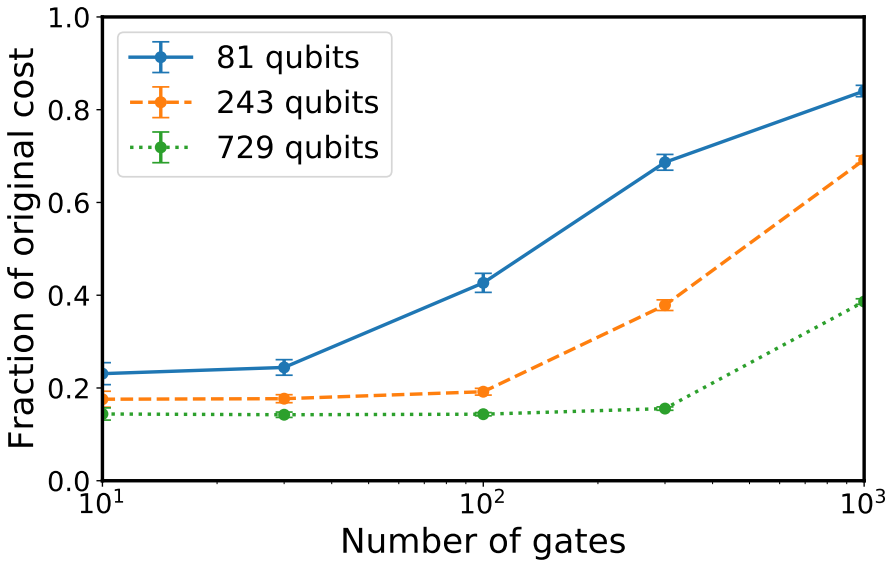


Figure 2.12: Plot of the average ratio (total gate distance after partition-and-rotate)/(total gate distance before) given 100 trials each for different numbers of random gates and random qubits. Error bars represent one standard deviation. As the number of gates begins to saturate the number of qubits, the possible improvement from optimization begins to decrease.

random circuits. For each trial, we first generate a random circuit of N_g two-qubit gates on N total qubits. The precise type of two-qubit gate is irrelevant in this framework. Likewise, single-qubit gates require no communication overhead, so we do not consider them. The random circuit then implies a computational graph C, where, as described above, the vertices represent the algorithm qubits and the edge weights represent the number of gates that must be applied between each pair of qubits. Once this computational graph has been generated, we first attempt to map it blindly to the hierarchy graph, using the addressing scheme of Sec. 2.2.2.3 and an arbitrary order of the graph C. Then, we apply partition-and-rotate and calculate the new cost function. By comparing the cost function between these two, we develop an idea of how much long-range quantum information processing is eliminated by partition-and-rotate. We perform this several times to build up statistics on average time costs and average improvement.

Code which performs circuit placement and generates the profiling figures included in this section can be found at [84].

In our simulations, we test hierarchies $K_3^{\Pi k}$ up to 729 qubits ($k = 6$). We find that as gates are added, the improvement over the initial cost is decreased. This is sensible, because as more randomly placed gates are present, different node mappings become more similar. Such an effect will likely not be present for quantum algorithms which do not have their gates placed randomly. For cases in which the number of gates is significantly fewer than the number of qubits, partition-and-rotate is able to significantly reduce the cost function. We find that 100 gates can be placed on a 729 qubit hierarchy with a total cost less than 20 % of the original on average. When 1000 gates are placed on a 729 qubit hierarchy, the final cost is still only 40 % of the initial one. Results for $K_3^{\Pi 4}$, $K_3^{\Pi 5}$, and $K_3^{\Pi 6}$ can be seen in Fig. 2.12.

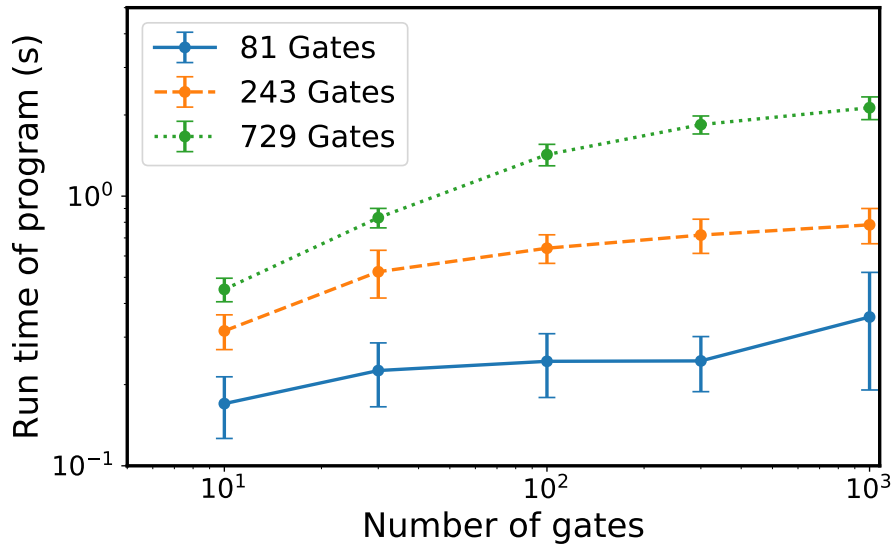


Figure 2.13: Average run times over 100 trials for partition-and-rotate on a 2015 MacBook Pro with a 2.6 GHz processor. Each line represents an increasing number of gates for a constant circuit size as measured by the number of qubits. Error bars represent one standard deviation.

Next, we examine the time required to place such a circuit. Our code, most of which

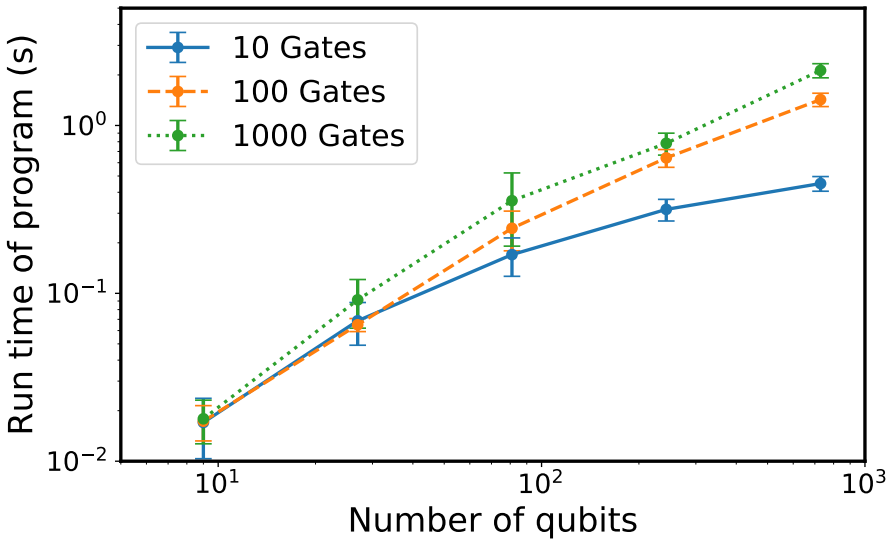


Figure 2.14: Average run times over 100 trials for partition-and-rotate on a 2015 MacBook Pro with a 2.6 GHz processor. Each line represents an increasing number of qubits for a constant number of gates. Error bars represent one standard deviation.

is written in Python3 but which uses a C implementation of Metis for graph partitioning, can place 1000 gates on a 729-qubit hierarchy in roughly two seconds when running on a 2015 MacBook Pro. Although the algorithm seems naturally suited to parallelization, our implementation uses only a single core. Our current implementation appears to scale with the number of qubits as $\mathcal{O}(N)$ and not to depend on the number of gates included at all once there are a sizable number of gates. We illustrate these two relationships in Figs. 2.13 and 2.14. These times compare favorably to the times reported in Ref. [85], with much optimization still possible in our implementation.

Note that using random graphs as described above means that our results may not be valid for more general quantum algorithms. It is possible that practical quantum algorithms have structure that makes them either particularly amenable or particularly difficult for partition-and-rotate algorithms to place, depending on the actual algorithm being examined.

2.6 Conclusions and Outlook

In this chapter, we have developed the theory of hierarchies using the existing binary operation of the graph hierarchical product. We have shown that hierarchies may be a promising architecture for large quantum information processing systems. To demonstrate this, we analyzed both properties of the underlying graph (such as diameter, maximum degree, total edge weight) as well as the time it would require to perform a representative quantum information process (constructing the GHZ state/state transfer) in both deterministic and probabilistic settings. We have also computed and tabulated these properties for many other graphs which appear as potential architectures, for comparison. We have shown that, for much of parameter space, hierarchies have favorable scalings in cost and performance with the total number of qubits N compared to these competitors. Also, since hierarchical graphs are hyperbolic, they share many of the advantages of hyperbolic graphs such as efficient routing schemes [91], network security [92], and node addressal [93].

We have also presented a conceptually simple circuit placement algorithm which allows for simple optimization using existing graph-partitioning software packages. Our partition-and-rotate algorithm scales well with the number of qubits and gates in the circuit and reliably reduces the total distance that needs to be traversed by random quantum circuits, which we verified by simulation.

One significant limitation of our analysis in this chapter has been that we remained confined to unitary operations. Non-unitary operations (for instance, measurements which are then fed forward to choose future unitary operations) are capable of establishing

long-range correlations like those present in the GHZ state much more quickly than unitary ones if measurements and classical communication are fast. In the future, we hope to extend our results into non-unitary domains [94].

In addition, we have made the assumption that the primary way in which quantum architectures will differ is the speed with which two qubits can communicate (as represented by our time weights on edges). Another important case might be one in which the primary way edges are enhanced is by improving bandwidth or duplicating nodes to provide parallel routes rather than affecting gate speed directly. For some schemes, our abstract notion equating the time of a two-qubit gate with the edge weight may still be a useful tool of analysis, but in other cases bandwidth and speed may not be interchangeable. We intend to undertake the analysis appropriate for this case in a future manuscript [94].

In this chapter, we limited ourselves to consideration of a few quantum processes (generation of a large entangled state, or transfer of a state across the graph), which might not be representative of other, more general distributed quantum information tasks. Some algorithms, such as Shor's algorithm, are known to be able to run with little additional overhead even on one-dimensional, nearest-neighbor graphs [95]. Therefore, when selecting an architecture for a practical quantum computer, care will need to taken to select the proper benchmarking task.

In future work, we hope to look at a wider variety of quantum circuits and use those to better benchmark different modular architectures. In addition, we hope to gain a better understanding of the treatment of probabilistic links for general graphs. For instance, as we discussed briefly when assessing the star graph S_N, one real concern in a networked

setting is whether some parts of the network will form bottlenecks. To analyze the impact of this in a general way will require a better understanding of realistic quantum algorithms and the demands they place on a network. Analyzing more complex quantum algorithms could also shed light on the performance of partition-and-rotate placement algorithms in realistic settings when sequencing and scheduling also enter into consideration.

Finally, in addition to asking ourselves how current circuits and algorithms can be executed on highly modular systems, we also hope to explore the possibility that highly modular architectures open up new possibilities for parallelized quantum algorithms. For instance, Ref. [96] shows that quantum fan-out gates can be used to parallelize gate sequences, decreasing the time to perform an algorithm at the cost of requiring additional memory qubits. Hierarchies could implement such schemes by using high-level connections to perform the initial fan-out gates and then performing the various parallelized operations in each individual module.

Chapter 3: Nearly optimal time-independent reversal of a spin chain

Quantum information transfer is a fundamental operation in quantum physics, and fast, accurate protocols for transferring quantum states across a physical system are likely to play a key role in the design of quantum computers [97, 98]. For example, quantum information transfer can be used to establish long-range entanglement and is also useful for qubit routing in quantum architectures with limited connectivity [15, 99]. Extensive work has studied the implementation of various information transfer protocols, often via Hamiltonian dynamics on spin chains [100].

Information transfer in Hamiltonian systems is governed by the spread of entanglement and has close links to Lieb-Robinson bounds [101], entanglement area laws [102], and algorithms for quantum simulation [103]. Fundamental limits to the rate of entanglement growth are set by bounds on the *asymptotic entanglement capacity* [104–107] and more recent small incremental entangling theorems [108–111]. We show that these limits can also be used to obtain lower bounds on the execution time of Hamiltonian protocols for information transfer. This raises the question of whether a protocol can achieve optimality by saturating the bound.

Quantum state transfer studies protocols for moving qubits through a spin chain [112]. Long-range interactions can be used to speed up protocols [113], but here

we consider only nearest-neighbor interactions. State transfer protocols usually assume the intermediate medium to be in a known initial state [114–117] or allow it to change in an unknown or non-trivial manner [118, 119]. Such protocols are not directly applicable when some or all spins in the chain contain data qubits that need to be transferred or maintained.

Protocols for *state reversal*, also known as *state mirroring* [120], take steps towards addressing this issue. State reversal reverses any input state on a spin chain about the center of the chain. Specifically, with qubit labeling $1, 2, \ldots, N$, state reversal corresponds to the unitary

$$\mathrm{R} \coloneqq \prod_{k=1}^{\lfloor \frac{N}{2} \rfloor} \mathrm{SWAP}_{k,N+1-k} \tag{3.1}$$

up to a *global phase*, which is independent of the state. State reversal is potentially a useful subroutine for the more general task of qubit routing, where we wish to apply arbitrary permutations to the qubits. Early results in this area require the state to be in the single-excitation subspace [121] or introduce phases in the final state that depend on a non-local property such as the number of qubits in state $|1\rangle$ [120, 122]. The protocol in [120] introduces a phase $(-1)^{M(M-1)/2}$ that is a function of the excitation number $M \pmod 4$. This is non-trivial to correct: Consider the task of signaling the value of the left bit to the right end of a chain with zeros in the bulk. A right edge state prepared in $|+\rangle$ is flipped to $|-\rangle$ by the phase correction procedure, conditioned on the value of the left bit. By the signaling lower bound, we incur a time over- head linear in N to correct these phases and implement a reversal for a general state. These limitations were later removed by time-dependent protocols for state reversal [123–125]. Concepts from Refs. [123, 124]

can also be used for translation-invariant universal quantum computation by, for example, modelling a quantum cellular automaton [123, 126–128].

In this chapter, we propose the first *time-independent* protocol for state reversal using nearest-neighbor interactions. We show that the execution time of our protocol is nearly optimal, comparable to the time-dependent protocol given in Ref. [123]. However, as our protocol does not require dynamical control or intermediate measurements but only engineered nearest-neighbor couplings, we expect it to be more experimentally feasible on near-term quantum systems such as superconducting qubits [129]. Through simulations, we also find that the protocol has reduced error scaling in system size to noise by static disorder caused by imperfect fabrication (see Sec. 3.6). Therefore, we expect that the protocol presented here has applications in noisy, connectivity-limited quantum devices at intermediate scale.

Before presenting our state reversal protocol in more detail, let us elaborate on the claim that it is *nearly optimal*—specifically, that it has an evolution time within a factor $1.502(1 + 1/N)$ of the shortest possible. In order to compare fairly between the runtimes of Hamiltonian protocols (such as the one we will introduce shortly) and gate-based protocols (such as SWAP-based routing algorithms), it is necessary to standardize the notions of time in the two models. Time in a gate-based model is measured by circuit depth, which implicitly assigns a time of 1 to one application of a gate. However, it is possible to provide a rigorous lower bound on the evolution time required to simulate two-qubit gates using a two-qubit Hamiltonian whose spectral norm is bounded by 1 [130]. Therefore, for any nearest-neighbor spin Hamiltonian H, a time scale follows from a normalization that limits the strength of every two-qubit interaction but allows fast local

operations. Up to local unitaries, we can write any two-qubit Hamiltonian in the *canonical form* [130]

$$K \coloneqq \sum_{j \in \{x,y,z\}} \mu_j \sigma_j \otimes \sigma_j \,, \tag{3.2}$$

where $\mu_x \geq \mu_y \geq |\mu_z| \geq 0$ and σ_j are the Pauli matrices. We impose the normalization condition that $\|K\| = \sum_j |\mu_j| \leq 1$ for all interactions, where $\|\cdot\|$ is the spectral norm. Under this normalization, a SWAP can be optimally implemented in time $3\pi/4$ [131], and our protocol achieves state reversal in time

$$t_N \coloneqq \pi \sqrt{(N+1)^2 - p(N)}/4 \,, \tag{3.3}$$

where $p(N)$ is the parity function, which is 0 when N is even and 1 when N is odd. This is equivalent in time to a SWAP gate circuit of depth $\sim N/3$. As state reversal using only SWAPs requires depth at least $N - 1$ [132], our protocol is faster than any SWAP-based protocol by an asymptotic factor of 3. Similarly, we can compare to other time-independent Hamiltonian protocols that use nearest-neighbor interactions: [114] implements state transfer in time $N\pi/4$ and [120] implements state reversal in time $N\pi/2$ but introduces relative phases in the state as mentioned earlier. Our time-independent protocol (and some time-dependent protocols [123–125]) thus improve upon these previous protocols for state transfer and state reversal except for a subleading term.

We lower-bound the time for state reversal, which can generate entanglement across a bipartition, by using bounds on the asymptotic entanglement capacity in a more general model [105, 107]. The asymptotic entanglement capacity bounds the

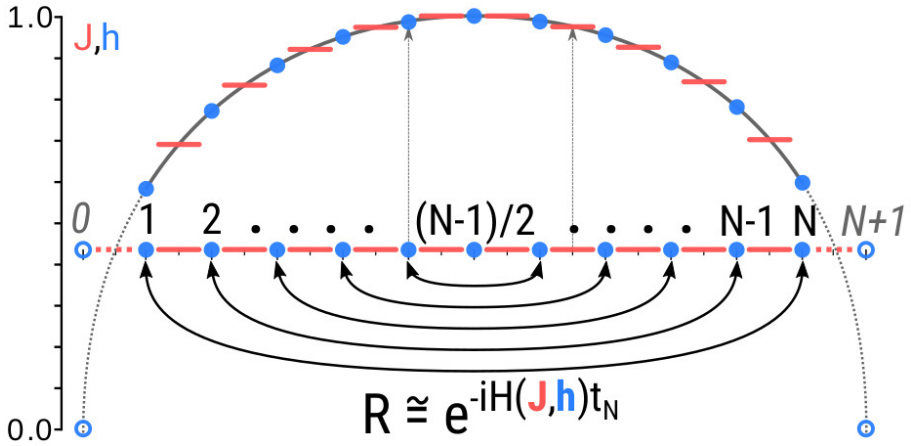


Figure 3.1: The state reversal operation R (depicted by arrows) and an illustration of our time-independent protocol to implement it. The nearest-neighbor $\sigma_x^k \sigma_x^{k+1}$ couplings (J_k, dashes) and on-site σ_z^k fields (h_k, dots) are plotted on the y-axis. Sites $0, N+1$ are ancilla qubits, which are not part of the protocol and are used purely in the analysis.

rate at which entanglement can be generated by any evolution of a given bipartite Hamiltonian interspersed with arbitrary local operations and classical communication (LOCC) and with arbitrary finite local ancilla spaces. We give an explicit example of entanglement generated by state reversal and lower-bound the time using the capacity of a normalized two-qubit interaction in canonical form (3.2), even allowing for LOCC. Nonetheless, our state reversal protocol is able to nearly saturate this bound without classical communication, without ancillas, and with only nearest-neighbor interactions throughout the chain.

We propose a state reversal protocol with Hamiltonian of the form

$$H(\mathbf{J},\mathbf{h}) = J_0\sigma_x^1 + \sum_{k=1}^{N-1} J_k \sigma_x^k \sigma_x^{k+1} + J_N \sigma_x^N - \sum_{k=1}^{N} h_k \sigma_z^k, \tag{3.4}$$

where the coefficients $\mathbf{J}, \mathbf{h}$ are engineered as follows. Letting

$$a_k := \pi\sqrt{(N+1)^2 - (N+1-k)^2}/(4t_N)\,, \tag{3.5}$$

for $k \in \mathbb{N}$, our protocol is defined as (see also Fig. 3.1)

Protocol 3.0.1. Let $J_k = a_{2k+1}, h_k = a_{2k}$ for all sites k, and let $H := H(\mathbf{J}, \mathbf{h})$. Apply $U := e^{-it_N H}$ to the input state.

We show in the following sections that our protocol implements state reversal exactly, up to a global phase (we denote this equivalence by $\cong$). In other words,

Theorem 3.0.2. $U \cong \mathrm{R}$.

3.1 Proof and analysis of the protocol

We prove the correctness of our protocol (i.e., Theorem 3.0.2) by mapping the spin chain to a doubled chain of Majorana fermions via a Jordan-Wigner transformation, describing the action in the Majorana picture, and then mapping back to the spin picture. To help with the analysis, we extend the chain with two ancillary sites $\{0, N+1\}$ called the *edge*, E, and refer to the sites $\{1, \ldots, N\}$ as the *bulk*, B. We define the transverse-field Ising model (TFIM) Hamiltonian

$$\widetilde{H} := \sum_{k=0}^{N} a_{2k+1}\sigma_x^k\sigma_x^{k+1} - \sum_{k=1}^{N} a_{2k}\sigma_z^k. \tag{3.6}$$

on the extended chain that reduces to H when the edge is initialized to state $|{+}{+}\rangle$. Similarly, we define $\widetilde{U} := e^{-i\widetilde{H}t_N}$. Note that the operator $\widetilde{H}$ (and hence $\widetilde{U}$) acts trivially

on $|{+}{+}\rangle_E$, so this edge state does not change through the course of the evolution. (Our results also hold using the edge state $|{-}{-}\rangle_E$, which is equivalent to negating the sign of the longitudinal fields in (3.4).) We then prove that in the Heisenberg picture, Pauli matrices on site k map to the corresponding Pauli on site $N+1-k$ for all sites k in the chain.

First, we map to the doubled chain of Majorana fermionic operators by defining

$$\gamma_{2k} \coloneqq P_{[0,k-1]} \cdot \sigma_x^k, \quad \gamma_{2k+1} \coloneqq P_{[0,k-1]} \cdot \sigma_y^k \tag{3.7}$$

at each site, where we have used the notation $P_{[a,b]} \coloneqq \prod_{j=a}^{b}(-\sigma_z^j)$ for the Jordan-Wigner parity string between sites a and b. The γ_k are Hermitian and satisfy the Majorana anti-commutation relations $\{\gamma_j, \gamma_k\} = 2\delta_{jk}$. We also see that $\sigma_z^k = -i\gamma_{2k}\gamma_{2k+1}$ and $\sigma_x^k\sigma_x^{k+1} = i\gamma_{2k+1}\gamma_{2k+2}$, leading (3.6) to take the form

$$\widetilde{H} = i\sum_{k=1}^{2N+1} a_k\gamma_k\gamma_{k+1}\,. \tag{3.8}$$

The Majoranas γ_0, γ_{2N+3} do not appear in the sum, since $a_0 = a_{2N+2} = 0$. In the following lemma, we show how $\widetilde{U}$ transforms the Majorana operators. Our main technique is an analogy with the dynamics of the y component of the spin operator for a spin $N+\frac{1}{2}$ particle, similar to Refs. [114, 120]. Here, the same analogy provides a stronger protocol which gives state reversal on all spins in the chain without introducing relative phases.

Lemma 3.1.1. The operation $\widetilde{U}$ acts on the Majorana operators as

$$\widetilde{U}\gamma_k\widetilde{U}^\dagger = \begin{cases} \gamma_k & \text{if } k = 0, 2N+3, \\ (-1)^{k-1}\gamma_{2N+3-k} & \text{otherwise.} \end{cases} \tag{3.9}$$

Proof. For the first case, $\widetilde{H}$ has no overlap with operators γ_0 and γ_{2N+3}, so they are stationary under evolution by $\widetilde{H}$.

For the remaining cases, we use the analogy with a spin $s = N + \frac{1}{2}$ particle. The Heisenberg evolution of γ_k corresponds to the rotation of the S_z eigenstate $|s, k-s-1\rangle$ of magnetization $k-s-1$. Observing that

$$\frac{i\pi}{4t_N} \langle s, m| S_y |s, m'\rangle = a_{s+m+1}(\delta_{m'(m+1)} - \delta_{m(m'+1)}) \tag{3.10}$$

(with $\hbar = 1$), we can express (3.8) in the bilinear form $\widetilde{H} = \frac{1}{2}\boldsymbol{\gamma}^\dagger A\boldsymbol{\gamma}$, for the vector $\boldsymbol{\gamma} \coloneqq \begin{bmatrix} \gamma_1 & \gamma_2 & \dots & \gamma_{2N+2} \end{bmatrix}$ and the matrix $A \coloneqq -\pi/(2t_N)S_y$ expressed in the S_z basis. Using the Majorana commutation relations, we have $\dot{\boldsymbol{\gamma}} = i[\widetilde{H}, \boldsymbol{\gamma}] = 2iA\boldsymbol{\gamma}$, so $\boldsymbol{\gamma}(t) = e^{2iAt}\boldsymbol{\gamma}(0)$. The Heisenberg evolution of γ_k under $\widetilde{H}$ for time t_N is exactly analogous to the (Schrödinger) time evolution of the state $|s, k-s-1\rangle$ under S_y for time π. A π-rotation under S_y maps

$$|s, -s+k-1\rangle \mapsto (-1)^{k-1} |s, s-k+1\rangle , \tag{3.11}$$

and correspondingly, $\gamma_k(t_N) = (-1)^{k-1}\gamma_{2N+3-k}$. □

Note that Eq. (3.11) can easily be verified for a spin-$1/2$ particle. Similarly, a spin-s

particle may be viewed as a system of $2s$ spin-$\frac{1}{2}$ particles with maximal total spin. In this picture, a π-rotation under S_y corresponds to independent π-rotations of each small spin. Since the state $|s, k-s-1\rangle$ is represented by a permutation-symmetric state with $k-1$ up spins, the π-rotation maps it to a state with $2s-(k-1)$ up spins and introduces a phase (-1) for each up spin, which is precisely (3.11).

Due to the signed reversal of the Majoranas in Lemma 3.1.1, the parity string $P_{[0,k]} = i^{b+1-a}\prod_{j=2a}^{2b+1}\gamma_j$ is (with the exception of γ_0) reflected about the center of the chain with an overall phase that exactly cancels when the product is reordered by increasing site index. The invariance of the edge Majoranas is crucial, as it provides a phase factor that cancels the state-dependent phases when we revert to the spin picture. In particular, we have the following lemma.

Lemma 3.1.2. The operation $\widetilde{U}$ acts on the parity strings as $\widetilde{U}P_{[0,k]}\widetilde{U}^\dagger = i\sigma_x^0\sigma_x^{N+1}P_{[0,N-k]}$ for all k.

Proof. Applying Lemma 3.1.1, we have

$$\widetilde{U}P_{[0,k]}\widetilde{U}^\dagger = i^{k+1}(-1)^{k(2k+1)}\gamma_0\prod_{j=1}^{2k+1}\gamma_{2N+3-j}\,. \tag{3.12}$$

$$= \gamma_0 P_{[0,N]}P_{[0,N-k]}\gamma_{2N+2} \tag{3.13}$$

where we reordered the product and used $P_{[N+1-k,N]} = P_{[0,N]}P_{[0,N-k]}$. From the Majorana anti-commutation relations and (3.7), the result follows. □

Now we prove the main theorem.

Proof of Theorem 3.0.2. $U \cong \mathrm{R}$ holds iff all bulk observables on the chain transform

identically under U, R. For any operator $\mathcal{O}^k$ supported on bulk site $k \in \{1, \ldots, N\}$, we show that $U\mathcal{O}^k U^\dagger = \langle{+}{+}| \widetilde{U}\mathcal{O}^k\widetilde{U}^\dagger |{+}{+}\rangle_E = \mathcal{O}^{N+1-k}$. (Henceforth we drop the edge subscript E.) By Eq. (3.7) and Lemmas 3.1.1 and 3.1.2, σ_x^k is mapped to

$$U\sigma_x^k U^\dagger = \langle{+}{+}| \widetilde{U} P_{[0,k-1]}\gamma_{2k}\widetilde{U}^\dagger |{+}{+}\rangle \tag{3.14}$$

$$= -i \langle{+}{+}| \sigma_x^0\sigma_x^{N+1} P_{[0,N+1-k]}\gamma_{2N+3-2k} |{+}{+}\rangle \tag{3.15}$$

$$= -i\sigma_z^{N+1-k}\sigma_y^{N+1-k} = \sigma_x^{N+1-k}\,. \tag{3.16}$$

Next, we use Lemma 3.1.2 to show that σ_z^k is mapped to

$$U\sigma_z^k U^\dagger = -\langle{+}{+}| \widetilde{U} P_{[0,k-1]}P_{[0,k]}\widetilde{U}^\dagger |{+}{+}\rangle \tag{3.17}$$

$$= \langle{+}{+}| \sigma_x^0\sigma_x^{N+1} P_{[0,N+1-k]}\sigma_x^0\sigma_x^{N+1} P_{[0,N-k]} |{+}{+}\rangle \tag{3.18}$$

$$= \sigma_z^{N+1-k}\,. \tag{3.19}$$

All other observables can be written in terms of the on-site Pauli operators σ_x^k, σ_z^k, so U is identical to R, up to global phase. □

3.2 Time lower bound

We now prove a lower bound on the optimal time, t^*, to implement state reversal using normalized local interactions. Let the entanglement entropy between systems A and B of a bipartite state $|\psi\rangle_{AB}$ be $\mathrm{E}(|\psi\rangle)$, defined as the local von Neumann entropy $S(\rho) \coloneqq -\operatorname{Tr}[\rho \log_2 \rho]$, for $\rho = \operatorname{Tr}_B[|\psi\rangle\langle\psi|]$. Then, the asymptotic entanglement capacity

of a Hamiltonian H that couples systems A and B was shown to equal [107]

$$E_H = \sup_{|\psi\rangle \in \mathcal{H}_{AA'BB'}} \lim_{t \to 0} \frac{\mathrm{E}\left(e^{-iHt}|\psi\rangle\right) - \mathrm{E}(|\psi\rangle)}{t}, \tag{3.20}$$

where $\mathcal{H}_{AA'BB'}$ is the Hilbert space of the bipartite systems A and B with arbitrarily large ancilla spaces A' and B', respectively. In particular, for a Hamiltonian of the form $\sigma_x \otimes \sigma_x$, [104, 105] showed that

$$\alpha := E_{\sigma_x \otimes \sigma_x} = 2 \max_y \sqrt{y(1-y)} \log_2 \frac{y}{1-y} \approx 1.912. \tag{3.21}$$

This is tighter than the more general small incremental entangling bound $E_H \leq c\|H\| \log_2 d = 2$ for the conjectured $c = 2$ [108] (best known $c = 4$ [110]) and where the smallest dimension of A or B gives $d = 2$. Since E is invariant under local unitaries, a direct corollary is that $E_{\sigma_y \otimes \sigma_y} = E_{\sigma_z \otimes \sigma_z} = \alpha$.

We now show that Protocol 3.0.1 is close to the shortest time possible.

Theorem 3.2.1. It holds that $\frac{t_N}{t^*(1+1/N)} \leq \alpha\pi/4 < 1.502$.

Proof. We prove the time lower bound via an upper bound on the rate of increase of entanglement across a cut in the center of the chain (allowing differences of one qubit for odd N). Designate the left half of the cut as subsystem $\mathcal{A}$ and the right half as subsystem $\mathcal{B}$. $\mathcal{A}$ consists of subsystem A given by the qubit at site $\lfloor N/2 \rfloor$ adjacent to the cut, and subsystem A' consisting of the remaining qubits to the left of the cut as well as a finite but arbitrary number of ancilla systems that are not part of the chain. Similarly, $\mathcal{B}$ consists of subsystem B, the qubit at site $\lfloor N/2 \rfloor + 1$, and B', the remaining qubits in the right half

with an arbitrary finite number of ancilla.

Consider Hamiltonians of the form $H(t) = K(t) + \bar{K}(t)$ specifying the evolution of the $\mathcal{AB}$ system, where $K(t)$ is a two-qubit Hamiltonian supported on systems AB (i.e., the cut edge), while $\bar{K}$ contains terms supported on AA' or BB' but not the cut edge AB. For brevity, we drop the time parameter t even though we allow the Hamiltonian to be time-dependent. We assume that K is expressed in canonical form (3.2) due to equivalence under local unitaries. Aside from its support, we make no assumptions about the form of $\bar{K}$ (so the resulting bound is more general than nearest-neighbor interactions). We call H satisfying these conditions *divisible* and also call protocols using divisible Hamiltonians divisible.

Observing that E_H is the supremum over a time derivative of the von Neumann entropy of $\rho = \mathrm{Tr}_{\mathcal{B}} |\psi\rangle \langle\psi|$, we have

$$E_H = \sup_{|\psi\rangle} \mathrm{Tr}\left(-\frac{d\rho}{dt}\log\rho - \rho\frac{d\log\rho}{dt}\right) \tag{3.22}$$

$$= \sup_{|\psi\rangle} \mathrm{Tr}\left(-\frac{d\rho}{dt}\log\rho\right). \tag{3.23}$$

The reduced density matrix ρ has time evolution

$$\frac{d\rho}{dt} = -i\,\mathrm{Tr}_{\mathcal{B}}\left[H, |\psi\rangle\langle\psi|\right]. \tag{3.24}$$

We substitute $H = \bar{K} + \sum_{j\in\{x,y,z\}} \mu_j\sigma_j \otimes \sigma_j$ in the commutator and substitute the time-dependence of ρ into Eq. (3.23). By linearity of the trace and sublinearity of the

supremum, we get

$$E_H \leq E_{\bar{K}} + \sum_{j\in\{x,y,z\}} \mu_j E_{\sigma_j\otimes\sigma_j} \leq \alpha\,, \tag{3.25}$$

where we observe that $E_{\bar{K}} = 0$ since $\bar{K}$ does not have support across the cut, and use the normalization condition $\sum_j |\mu_j| \leq 1$. This bound holds for all divisible Hamiltonians H, with nearest-neighbor Hamiltonians as a special case.

The entanglement generated by any divisble protocol can now be bounded in time. We observe that if the protocol contains local measurements then these cannot increase entanglement $\mathrm{E}(|\psi\rangle)$ and that feedback may be viewed as a particular time-dependence of H conditioned on measurement outcomes. Therefore, (3.25) bounds the total increase in entanglement across bipartition $\mathcal{AB}$ over a time t^* by

$$\mathrm{E}(|\psi(t^*)\rangle) - \mathrm{E}(|\psi(0)\rangle) \leq \alpha t^* \tag{3.26}$$

for any initial state $|\psi(0)\rangle$ acted on by a divisible protocol and LOCC.

Finally, we give an explicit bound on the worst-case time of divisible state reversal protocols by specifying an initial state. Let the system start in the product state $|\phi\rangle_{\mathcal{A}}\otimes|\phi\rangle_{\mathcal{B}}$ where each qubit forms a Bell state with a local ancilla not part of the chain. Clearly, $\mathrm{E}(|\phi\rangle_{\mathcal{A}} \otimes |\phi\rangle_{\mathcal{B}}) = 0$. We perform a reversal R on the chain and get the state $|\psi\rangle_{\mathcal{AB}} \coloneqq \mathrm{R}(|\phi\rangle_{\mathcal{A}} \otimes |\phi\rangle_{\mathcal{B}})$, which is maximally entangled, i.e., $\mathrm{E}(|\psi\rangle_{\mathcal{AB}}) = N$. Then, (3.26) gives the bound

$$t^* \geq \frac{\mathrm{E}(|\psi\rangle_{\mathcal{AB}}) - \mathrm{E}(|\phi\rangle_{\mathcal{A}} \otimes |\phi\rangle_{\mathcal{B}})}{\alpha} \geq \frac{N}{\alpha} \tag{3.27}$$

on any divisible state reversal protocol. Comparing this to our protocol time (3.3), we

have

$$\frac{t_N}{t^*} \leq \frac{\alpha\pi\sqrt{(N+1)^2 - p(N)}}{4N} \leq \frac{\alpha\pi(1+1/N)}{4}. \qquad \square$$

3.3 Discussion

The time-dependent protocol in Ref. [123] is closely related to our time-independent protocol, and both can be described within the same framework (see Sec. 3.4). In the time-dependent case, the state is evolved alternately under two restrictions of the Hamiltonian (3.4): $H(\mathbf{1}, \mathbf{0})$ (uniform Ising) and $H(\mathbf{0}, \mathbf{1})$ (uniform transverse field), each for time $\pi/4$, for a total of $N+1$ rounds. In the Majorana picture, these Hamiltonians carry out a simultaneous braiding of neighboring Majoranas along even (resp. odd) edges of the doubled Majorana chain. The resulting map matches Lemma 3.1.1 exactly, implying that the two protocols are identical at the level of Majorana operators. Indeed, any protocol achieving the map in Lemma 3.1.1 is guaranteed to implement state reversal.

In fact, as shown in Sec. 3.5, there is an infinite family of nearest-neighbor, time-independent Hamiltonian protocols for state reversal that generalizes Protocol 3.0.1. This family is parameterized by a non-negative integer m, with modified $\sigma_x^k \sigma_x^{k+1}$ coupling $J_k^{(m)} \propto \sqrt{(2N+1-2k+4m)\,(2k+1+4m)}$ and unmodified σ_z^k field strength. Protocol 3.0.1 corresponds to the special case of $m = 0$. By choosing large m, the coupling strength can be engineered to be nearly uniform throughout the chain, which may be a desirable feature in experimental implementations of the protocol [122].

Moreover, we show by simulations that our protocol is more robust to noise by static disorder, caused by imperfect fabrication or tuning, in NISQ implementations

(see Sec. 3.6). The spectral distance of the gate-based and time-independent reversal protocols show a reduced scaling over SWAP when fitted to $\exp(N^a \delta^b)$ in the a parameter by (standard error) $0.33(0.020)$ and $0.25(0.020)$, respectively. This implies, for example, that with strong disorder and an error threshold of 0.03, a SWAP protocol can only reverse 4 sites, whereas the time-independent protocol can reverse 8 sites. We give an operational meaning to the spectral distance by relating it to the completely bounded trace norm (diamond norm) and state fidelity.

In general, we would like to know how fast we can perform qubit routing on graphs. Qubit routing is a key subroutine in quantum architectures with incomplete connectivity [15], and can potentially improve runtimes of general quantum algorithms by overcoming limitations imposed by the underlying qubit connectivity. Indeed, we later showed [19] that a constant factor speedup over a SWAP-based approach is achievable for qubit routing on the chain using our fast reversal protocol as a primitive. While a superconstant speedup is not possible in one dimension, higher dimensional systems may offer such opportunities. State transfer in these systems has been studied [133] but the general question of routing remains open. Our techniques show that routing protocols for higher dimensional systems may exist by exploiting similar mappings between spins and fermions [134–136]. It is also interesting to consider an alternative model for routing that includes local measurements and fast classical communication. While the time lower bound presented here rules out superconstant speedups for the chain in this alternative model, faster protocols are possible in other geometries [137]. Future work will explore these more general schemes for qubit routing.

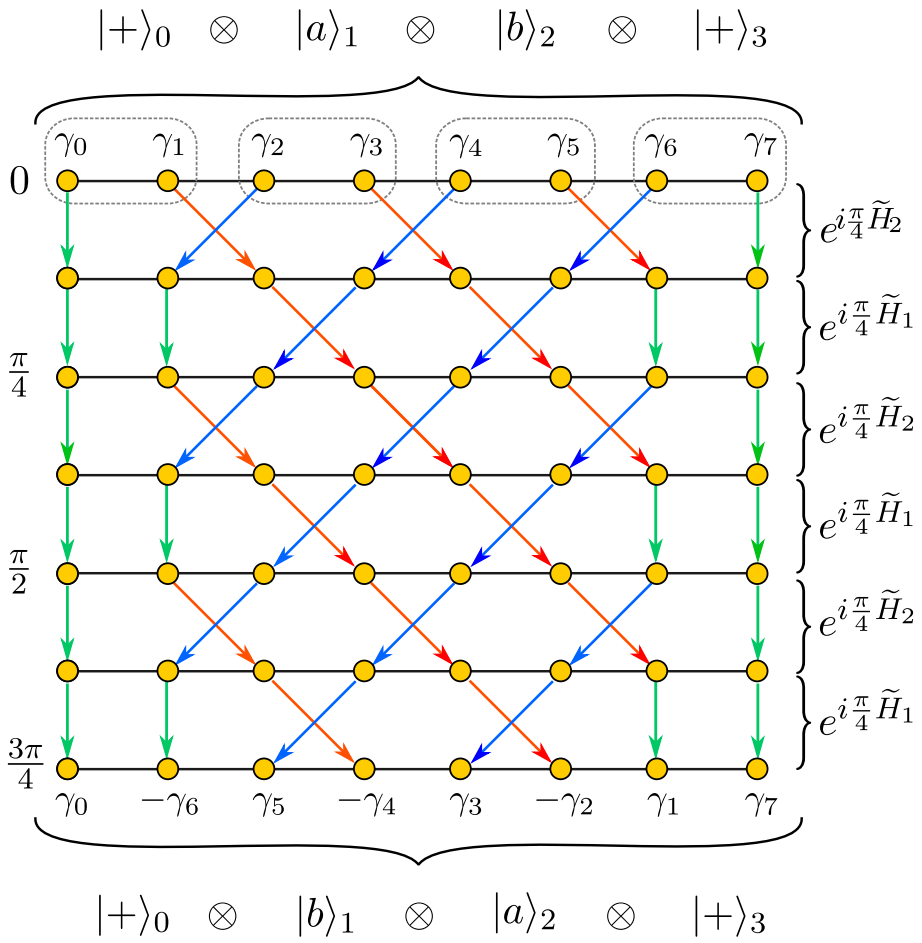


Figure 3.2: Time-dependent reversal protocol for $N = 2$ (with two edge ancillas). For any bulk state $|ab\rangle_{12}$ (with edge state $|{+}{+}\rangle_E$), alternating $\pi/4$ evolutions under $\widetilde{H}_2$, $\widetilde{H}_1$ are applied a total of $2N + 2$ times. Each step braids neighboring Jordan-Wigner Majoranas as indicated by the arrows; the right-movers keep the same sign while the left-movers gain a minus sign. The edge Majoranas γ_0, γ_7 are unchanged (a crucial feature that ensures the correct parity phases), while the intermediate Majoranas undergo reversal of position with alternating sign. The final state in the bulk of the chain is $|ba\rangle_{12}$.

3.4 Time-dependent protocol for reversal

In this section, we give a simple analysis of the time-dependent protocol given in Refs. [123, 124] using our methods. The strategy is to prove that this protocol satisfies Lemma 3.1.1. Lemma 3.1.2 and Theorem 3.0.2 are then automatically satisfied. First, we re-introduce the protocol using our notation.

Protocol 3.4.1. Let $H_h := H(\mathbf{0}, \mathbf{1})$ and $H_J := H(\mathbf{1}, \mathbf{0})$, where $\mathbf{1} = (1, 1, \ldots, 1)$ and $\mathbf{0} = (0, 0, \ldots, 0)$. Explicitly,

$$H_h = \sum_{k=1}^{N} Z_k \,, \tag{3.28}$$

$$H_J = X_1 + \sum_{k=1}^{N-1} X_k X_{k+1} + X_N \,. \tag{3.29}$$

Apply $V := \left(e^{i\frac{\pi}{4}H_h} e^{i\frac{\pi}{4}H_J}\right)^{N+1}$ to the input state.

As before, we extend the chain with two ancillary sites $\{0, N+1\}$ that constitute the edge E. The unitary V extends to an operator $\widetilde{V} := \mathbb{1}_E \otimes V$ on the extended chain. Then the following lemma holds.

Lemma 3.4.1. The operation $\widetilde{V}$ acts on the Majorana operators as

$$\widetilde{V}\gamma_k\widetilde{V}^\dagger = \begin{cases} \gamma_k & \text{if } k = 0, 2N+3, \\ (-1)^{k-1}\gamma_{2N+3-k} & \text{otherwise.} \end{cases} \tag{3.30}$$

Proof. We use Eq. (3.7) to write V as a product of alternating $\pi/4$-rotations under two Hamiltonians $\widetilde{H_J} = i\sum_{k=0}^{N} \gamma_{2k+1}\gamma_{2k+2}$ and $\widetilde{H_h} = i\sum_{k=1}^{N} \gamma_{2k}\gamma_{2k+1}$. Since $e^{-\pi/4\gamma_i\gamma_j}$ is a braiding unitary that maps $\gamma_i \mapsto \gamma_j, \gamma_j \mapsto -\gamma_i, \gamma_{k\neq i,j} \mapsto \gamma_k$, it follows that the operator $e^{i\frac{\pi}{4}\widetilde{H_h}}$ braids nearest-neighbor Majoranas along all odd edges of the chain (except the first and last edge), while $e^{i\frac{\pi}{4}\widetilde{H_J}}$ braids along the even edges. Therefore, alternating $\pi/4$ rotations under $\widetilde{H_J}$ and $\widetilde{H_h}$ implement an even-odd sort [138] on the chain, as shown in Fig. 3.2. Accounting for sign changes, the Majoranas map as follows: $\gamma_k \mapsto (-1)^{k+1}\gamma_{2N+3-k}$, while γ_0, γ_{2N+3} remain unchanged. □

3.5 Infinite family of Hamiltonians for state reversal

Reference [122] shows that there is an infinite family of XY Hamiltonians that generalize the protocol introduced in [120]. In fact, Protocol 3.0.1 is also a special case of an infinite family of protocols parameterized by a single non-negative integer m, as given below.

Protocol 3.5.1. Let $m \in \mathbb{Z}_{\geq 0}$, and

$$J_k^{(m)} := \frac{\pi}{4}\sqrt{(2k+1+4m)\,(2N+1-2k+4m)} \tag{3.31}$$

$$h_k^{(m)} := \pi\sqrt{k\,(N+1-k)} \tag{3.32}$$

for all sites $k = 1, \ldots, N$. Let $H^{(m)} = H(\mathbf{J}^{(m)}, \mathbf{h}^{(m)})$. Apply $U^{(m)} := e^{-iH^{(m)}}$ to the input state.

The protocol modifies only the couplings $J_k^{(m)}$ as a function of m, while the field terms $h_k^{(m)} = h_k$ are invariant with m. Note that $U^{(0)} = U$, so Protocol 3.0.1 is indeed a special case of Protocol 3.5.1. For convenience, we have rescaled the coefficients so that the evolution time is 1. To prove the correctness of this family of protocols, write the Hamiltonian $H^{(m)}$ in terms of Majorana fermions obtained by Jordan-Wigner transformation on the spin chain (extended to edge sites $\{0, N+1\}$). We have

$$H^{(m)} = \frac{1}{2}\boldsymbol{\gamma} \cdot A^{(m)} \cdot \boldsymbol{\gamma}, \tag{3.33}$$

where $\boldsymbol{\gamma} = \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_{2N+2} \end{bmatrix}$ and $A^{(m)}$ is a $(2N+2) \times (2N+2)$ tridiagonal matrix

with entries

$$A^{(m)} = i \begin{pmatrix} 0 & J_0^{(m)} & & & & & \\ -J_0^{(m)} & 0 & h_1 & & & & \\ & -h_1 & 0 & J_1^{(m)} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -h_N & 0 & J_N^{(m)} \\ & & & & -J_N^{(m)} & 0 \end{pmatrix}. \tag{3.34}$$

As before, the Heisenberg evolution of the Majoranas under $H^{(m)}$ is given by $\boldsymbol{\gamma}(t) = e^{2iA^{(m)}t}\boldsymbol{\gamma}(0)$. Lemma 3.1.1 shows that the operator $e^{2iA^{(0)}}$ implements reversal. Here we show that $e^{2iA^{(m)}} = e^{2iA^{(0)}}$ for all m, which implies that $U^{(m)}$ implements state reversal for all m. We state the following lemma (due to Refs. [139, 140]) on the spectrum of $A^{(m)}$.

Lemma 3.5.1. Let $A^{(m)}$ be as given in Eq. (3.34), and $s_k \coloneqq \mathrm{sgn}(2N+3-2k)$. Then $A^{(m)}$ has spectrum

$$E_k^{(m)} = \frac{\pi}{4}\left(2k - 2N - 3 + 4s_k m\right) \tag{3.35}$$

for $k = 1, \ldots, 2N+2$. The corresponding eigenvectors v_k satisfy the property $v_{kj} = (-1)^{N+k-j+1/2} v_{k(2N+3-j)}$.

Proof. The first claim follows from Ref. [139]. Via a transformation of the off-diagonals that preserves the spectrum, $A^{(m)}$ can be converted to a matrix $B(n,a)$ of Sylvester-Kac

type

$$B(n,a) \coloneqq \frac{\pi}{4}\begin{pmatrix} 0 & 1+a & & & & & \\ n+a & 0 & 2 & & & & \\ & n-1 & 0 & 3+a & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 2 & 0 & n+a & \\ & & & & 1+a & 0 & \end{pmatrix}, \tag{3.36}$$

for $n = 2N+1, a = 4m$. As shown in [139], the eigenvalues of $B(n,a)$ are given by the formula $\lambda_{\pm,j} = \pm\frac{\pi}{4}|2j+1+a|$ for $j \in \{0,\dots,n\}$, and the first claim follows.

For the second claim, we observe again that $A^{(m)}$ may be converted to a real, symmetric, tridiagonal matrix $C^{(m)}$ with positive off-diagonal entries via the similarity transformation $C^{(m)} \coloneqq DA^{(m)}D^{-1}$ where $D = \operatorname{diag}\left(i, i^2, \dots, i^{2N+2}\right)$. Reference [140] shows that the eigenvectors $u_k = Dv_k$ of $C^{(m)}$ (ordered by ascending eigenvalue) satisfy $u_{kj} = (-1)^{k-1}u_{k(2N+3-j)}$ for $k = 1,\dots,2N+2$. Correspondingly, the eigenvalues of $A^{(m)}$ satisfy $v_{kj} = (-1)^{k-1}i^{2N+3-2j}v_{k(2N+3-j)} = (-1)^{N+k-j+1/2}v_{k(2N+3-j)}$. □

Finally, we show that $e^{2iA^{(m)}}$ implements reversal.

Theorem 3.5.2. For all $m \in \mathbb{Z}_{\geq 0}$, $A^{(m)}$ satisfies $\left[e^{2iA^{(m)}}\right]_{jl} = (-1)^{j-1}\delta_{j(2N+3-l)}$.

Proof. Write

$$e^{2iA^{(m)}} = \sum_{k=1}^{2N+2} e^{2iE_k^{(m)}} v_k v_k^\dagger = \sum_{k=1}^{2N+2} (-1)^{k-N-3/2} v_k v_k^\dagger, \tag{3.37}$$

where we dropped the trivial phase $2\pi i m s_k$. The matrix elements of $e^{iA^{(m)}}$ are

$$\left[e^{iA^{(m)}}\right]_{jl} = \sum_{k=1}^{2N+2} (-1)^{k-N-3/2} v_{kj} v_{kl}^* \tag{3.38}$$

$$= \sum_{k=1}^{2N+2} (-1)^{2N+2-l} v_{kj} v_{k(2N+3-l)}^* \tag{3.39}$$

$$= (-1)^{j-1} \delta_{j(2N+3-l)} \,, \tag{3.40}$$

where in the second step we used Lemma 3.5.1 as $v_{kl}^* = (-1)^{l-k-N-1/2} v_{k(2N+3-l)}^*$. Therefore, $e^{2iA^{(m)}}$ maps $\gamma_k \mapsto (-1)^{k-1}\gamma_{2N+3-k}$, which implies that the protocol $U^{(m)}$ implements state reversal for all $m \in \mathbb{Z}_{\geq 0}$. □

When normalized so that all two-qubit terms are bounded by unity in spectral norm, $H^{(m)}$ implements state reversal in time $t_N^{(m)} = \frac{(N+1+4m)\pi}{4}$. Therefore, the time cost increases linearly in m and is minimal for Protocol 3.0.1 where $m = 0$. Next, observe that if we choose $4m \gg N$, the variation in coupling coefficients $J_k^{(m)}$ is small and on the order $\sim \frac{1}{8}\left(\frac{N+1}{2m}\right)^2$. Therefore, the parameter m quantifies a trade-off between reversal time and the non-uniformity of $J_k^{(m)}$. Setting $m = N + 1$, for example, yields a variation in the couplings on the order of 3% for any N, and gives reversal in time $5N\pi/4$.

3.6 Robustness of the protocol

Protocol 3.0.1 and its generalizations given in Sec. 3.5 are exact, i.e., any input state $|\psi\rangle$ maps perfectly to the output $\mathrm{R}\,|\psi\rangle$, assuming the interaction coefficients are implemented exactly as prescribed by the protocol. However, inherent in experimental systems is noise, and the usefulness of a given state transfer protocol is determined not

only by the time of operation and fidelity under perfect implementation, but also resilience to noise. There are many possible sources of noise, arising from device imperfections, interaction with the environment, or imperfect state preparation, execution or readout. Of these, imperfect fabrication can be modeled as a static noise term on every coefficient in the Hamiltonian. We compare our time-independent protocol with a swap-based protocol for reversal (odd-even sort) and a gate-based protocol [123].

Stochastic noise can be modeled as a perturbation to the Hamiltonian coefficients. For the case of disorder, we draw a single noise term for every coefficient from the normal distribution $\mathcal{N}$. We assume that the noise is multiplicative, so that the noise strength scales proportional to the magnitude of the coefficient. The perturbed Hamiltonian H' for our time-independent protocol then looks like

$$H' = J'_0\sigma_x^1 + \sum_{k=1}^{N-1} J'_k\sigma_x^k\sigma_x^{k+1} + J'_N\sigma_x^N - \sum_{k=1}^{N} h'_k\sigma_z^k\,, \tag{3.41}$$

where $J'_i = J_i \cdot (1+\delta J_i), h'_i = h_i \cdot (1+\delta h_i)$, where $\delta h_i \sim \mathcal{N}(\delta_h), \delta J_i \sim \mathcal{N}(\delta_J)$ for specified standard deviations $\delta_h, \delta_J \geq 0$. Evolution under this Hamiltonian gives a noisy reversal $\mathrm{R}' := e^{-iH'_i t_N}$ that reduces to R when $\delta_h = \delta_J = 0$. For swap and gate-based protocols, we compute the equivalent Hamiltonian formulation and similarly add noise terms.

A natural and widely used metric for the distinguishability of outputs of two quantum channels is the completely bounded trace norm (or diamond norm), which is the supremum over the trace distance between outputs over all inputs, for arbitrary identity extensions of the channels (3.55) [141]. The computation of the diamond norm can be

expressed as the solution to a particular optimization problem, making it a somewhat non-trivial quantity to compute. We consider unitary noisy models, where the diamond distance is equivalent to a simpler notion of distinguishability, the *spectral distance*

$$\Delta \equiv || \, \mathrm{R}' - \mathrm{R} \, || \, , \tag{3.42}$$

for the perfect and noisy state reversals R and R'. The diamond distance is at most two times as large as the spectral distance [142]. We also lower bound the output fidelity by the spectral distance, giving a relation to more practical figures of merit.

The spectral distance bounds the state distance

$$\| (\mathrm{R} - \mathrm{R}') \, |\psi_0\rangle \| \leq \Delta, \tag{3.43}$$

for an arbitrary input pure state $|\psi_0\rangle$. In fact, Δ can be used to bound the fidelity between perfect and noisy output states by

$$\begin{aligned}
\|\Delta\|^2 &= \|(\mathrm{R} - \mathrm{R}')^\dagger (\mathrm{R} - \mathrm{R}')\| && (3.44)\\
&= \max_{|\psi\rangle} | \, \langle\psi| \, (\mathrm{R} - \mathrm{R}')^\dagger (\mathrm{R} - \mathrm{R}') \, |\psi\rangle \, | && (3.45)\\
&= \max_{|\psi\rangle} | \, \langle\psi| \, 2\mathbb{1} - \mathrm{R}^\dagger \, \mathrm{R}' - \mathrm{R}'^\dagger \, \mathrm{R} \, |\psi\rangle \, | && (3.46)\\
&= \max_{|\psi\rangle} |2 - 2 \, \mathrm{Re} \, \langle\psi| \, \mathrm{R}^\dagger \, \mathrm{R}' \, |\psi\rangle \, | && (3.47)\\
&= 2 - \min_{|\psi\rangle} 2 \, \mathrm{Re} \, \langle\psi| \, \mathrm{R}^\dagger \, \mathrm{R}' \, |\psi\rangle && (3.48)\\
&\geq 2 - 2 \min_{|\psi\rangle} | \, \langle\psi| \, \mathrm{R}^\dagger \, \mathrm{R}' \, |\psi\rangle \, | \, , && (3.49)
\end{aligned}$$

where we used the fact that for any unitary U, $\mathrm{Re}\,\langle\psi|\,U\,|\psi\rangle \leq 1$, and $\mathrm{Re}\,[z] \leq |z|$ for any $z \in \mathbb{C}$. Next, note that the fidelity between perfect and noisy output states for input $|\psi\rangle$ is given by $|\,\langle\psi|\,R^\dagger R'\,|\psi\rangle\,|^2$. Let F_{min} denote the worst-case fidelity over all pure input states. Then,

$$\frac{1}{2}\Delta^2 \geq 1 - \sqrt{F_{min}} \implies F_{min} \geq \left(1 - \frac{1}{2}\Delta^2\right)^2 . \tag{3.50}$$

When $\Delta \leq \sqrt{2}$, we have $(1 - \Delta^2/2)^2 \geq 1 - \Delta^2$ and we get the bound

$$F_{min} \geq 1 - \Delta^2 . \tag{3.51}$$

Therefore, for small Δ, the infidelity between the perfect and noisy output states for any pure input state is at most Δ^2. Later in Theorem 3.6.1, we show a more general bound that holds for mixed inputs. For the numerics presented here, however, we assume pure input states.

We estimate the spectral distance dependence on noise and system size in the three candidate protocols. For each protocol, we probe the distance as a function of similar on-site and coupling disorder $\delta = \delta_h = \delta_J$, and increasing number of spins N. Since the spectral distance is computed by exact diagonalization, its runtime scales as an exponential in N and it is possible to probe system sizes up to $N = 14$ with the resources available. At these sizes, we can already see differences between the protocols, shown in Fig. 3.3. At each error rate δ, the swap protocol has the worst performance, followed by the time-independent protocol, and lastly, the gate-based protocol. We note that the gate-based and time-independent protocols perform within a standard deviation of one

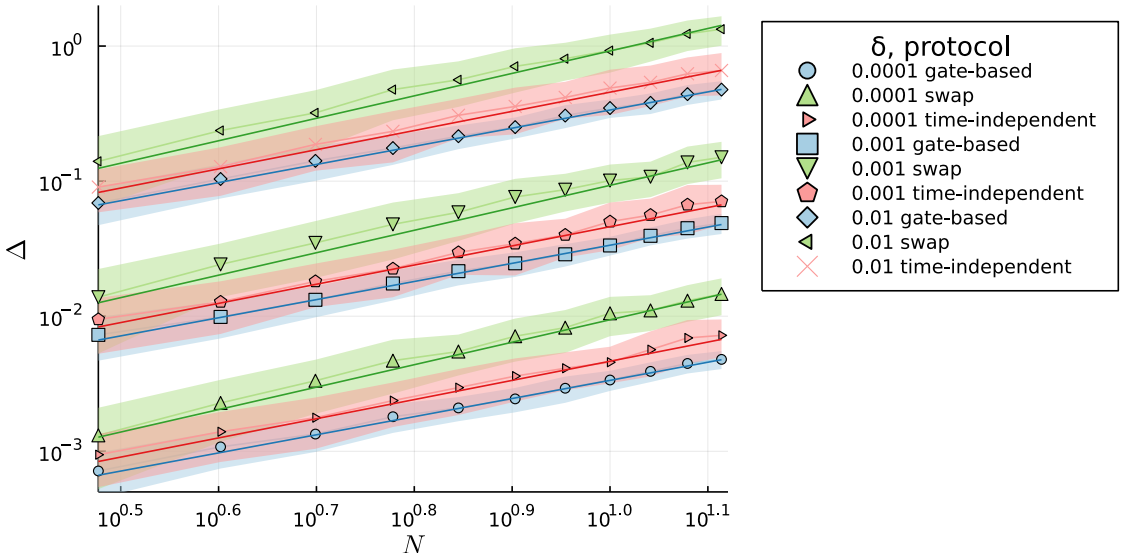


Figure 3.3: Spectral distance with standard deviation for different protocols under varying strengths of noise. We take 100 samples for each datapoint and use a linear fit for a power-law $\Delta = \exp(N^a \delta^b)$ controlled on the protocol, i.e., fitting $\log \Delta = a \log N + b \log \delta + O(1)$, to find (standard error) $a \approx 1.66(0.012)$ and $b \approx 0.994(0.0028)$ for the SWAP-based protocol. The b coefficient changes insignificantly for time-independent and gate-based protocols but the a coefficient is reduced by $0.31(0.016)$ for gate-based and $0.23(0.016)$ for time-independent protocols, indicating more robust scaling of these protocols in system size, relative to a SWAP-based protocol.

another, but the SWAP protocol is significantly noisier. For example, at a threshold of $\Delta \leq 0.03$, the swap can reverse only up to 4 sites, while the time-independent protocol can successfully reverse 8 sites. Therefore, the specialized protocols for reversal improve upon SWAP-based protocols not only in runtime but also in accuracy.

The relative accuracy between time-independent and gate-based protocols is not entirely predicted by our simulations here. The time-independent protocol does not vary in time, and derives its error primarily from imperfect engineering of the coupling strengths and interactions with the environment. The gate-based protocol, however, requires dynamical control, which is an additional source of noise. Since this noise source is likely to worsen the performance of the discrete protocol, we cannot make a definite comparison between the two protocols in our model which does not capture this effect.

Previously we showed that for pure input states, the infidelity is at most the squared spectral norm difference between the perfect and noisy reversal unitaries. Now we show a bound relating the fidelity to the spectral norm difference for mixed input states.

Theorem 3.6.1. For unitary operators U and V acting on an initial state ρ_0

$$F(U\rho_0 U^\dagger, V\rho_0 V^\dagger) \geq (1 - \|U - V\|)^2 ,$$

where the fidelity function F of states ρ and σ is defined as

$$F(\rho, \sigma) \equiv \mathrm{Tr}(\sqrt{\rho}\sigma\sqrt{\rho})^2 .$$

Proof. The Fuchs-van de Graaff inequality implies [141, 143]

$$F(U\rho_0U^\dagger, V\rho_0V^\dagger) \geq (1 - \frac{1}{2}\|U\rho_0U^\dagger - V\rho_0V^\dagger\|_1)^2\,, \tag{3.52}$$

where we use the trace norm $\|X\|_1 \equiv \mathrm{Tr}\,\sqrt{X^*X}$, for linear operators X. Now we can define the trace norm on linear maps Φ as

$$\|\Phi\|_1 \equiv \max\{\,\|\Phi(X)\|_1 : \|X\|_1 \leq 1\,\}\,. \tag{3.53}$$

Given that $\|\rho\|_1 = 1$ for any density operator ρ, and defining the unitary channels $\mathcal{U}(X) = UXU^\dagger$ and $\mathcal{V}(X) = VXV^\dagger$, the trace distance is upper bounded by

$$\|U\rho_0U^\dagger - V\rho_0V^\dagger\|_1 = \|\mathcal{U}(\rho_0) - \mathcal{V}(\rho_0)\|_1 = \|(\mathcal{U} - \mathcal{V})(\rho_0)\|_1 \leq \|\mathcal{U} - \mathcal{V}\|_1\,. \tag{3.54}$$

The completely bounded trace norm (or diamond norm) extends a given mapping Φ with an identity mapping $\mathbb{1}$ of (at most) the same input dimension

$$|||\Phi|||_1 \equiv \|\Phi \otimes \mathbb{1}\|_1 \tag{3.55}$$

and results in a bounded distance under extension with a (possibly entangled) ancilla system. The trace norm is upper bounded by the completely bounded trace norm

$$\|\Phi\|_1 = \max\{\,\|\Phi(X)\|_1 : \|X\|_1 \leq 1\,\} \leq \max\{\,\|(\Phi \otimes \mathbb{1})(Y)\|_1 : \|Y\|_1 \leq 1\,\} = |||\Phi|||_1 \tag{3.56}$$

because the maximization over linear operators Y is on a larger space. Finally, we can bound the completely bounded trace norm by the spectral norm [142, Lemma 7]

$$\|\|\mathcal{U} - \mathcal{V}\|\|_1 \leq 2\|U - V\| \tag{3.57}$$

for the special case of unitary channels. Bounding Eq. (3.52) by Eqs. (3.54), (3.56) and (3.57) gives the result. □

Chapter 4: Routing using fast reversal

4.1 Introduction

Qubit connectivity limits quantum information transfer, which is a fundamental task for quantum computing. While the common model for quantum computation usually assumes all-to-all connectivity, proposals for scalable quantum architectures do not have this capability [7–9]. Instead, quantum devices arrange qubits in a fixed architecture that fits within engineering and design constraints. For example, the architecture may be grid-like [12, 144] or consist of a network of submodules [7, 8]. Circuits that assume all-to-all qubit connectivity can be mapped onto these architectures via protocols for *routing* qubits, i.e., permuting them within the architecture using local operations.

Long-distance gates can be implemented using SWAP gates along edges of the graph of available interactions. A typical procedure swaps pairs of distant qubits along edges until they are adjacent, at which point the desired two-qubit gate is applied on the target qubits. These swap subroutines can be sped up by parallelism and careful scheduling [86, 145–150]. Minimizing the SWAP circuit depth corresponds to the ROUTING VIA MATCHINGS problem [15, 132]. The minimal SWAP circuit depth to implement any permutation on a graph G is given by its *routing number*, $\mathrm{rt}(G)$ [132]. Deciding $\mathrm{rt}(G)$ is generally NP-hard [151], but there exist algorithms for architectures

of interest such as grids and other graph products [15, 132, 152]. Furthermore, one can establish lower bounds on the routing number as a function of graph diameter and other properties.

Routing using SWAP gates does not necessarily give minimal circuit evolution time since it is effectively classical and does not make use of the full power of quantum operations. Indeed, faster protocols are already known for specific permutations in specific qubit geometries such as the path [18, 123]. These protocols tend to be carefully engineered and do not generalize readily to other permutations, leaving open the general question of devising faster-than-SWAP quantum routing. In this chapter, we give a positive answer to this question.

Rather than directly engineering a quantum routing protocol, we consider a hybrid strategy that leverages a known protocol for quickly performing a specific permutation to implement general quantum routing. Specifically, we consider the reversal operation

$$\rho \coloneqq \prod_{k=1}^{\lfloor \frac{n}{2} \rfloor} \mathrm{SWAP}_{k,n+1-k} \tag{4.1}$$

that swaps the positions of qubits about the center of a length-n path. Fast quantum reversal protocols are known in the gate-based [123] and time-independent Hamiltonian [18] settings. The reversal operation can be implemented in time [18]

$$T(\rho) \leq \frac{\sqrt{(n+1)^2 - p(n)}}{3} \leq \frac{n+1}{3}, \tag{4.2}$$

where $p(n) \in \{0, 1\}$ is the parity of n. 'Time' here is to be understood as the evolution

time of a Hamiltonian with two-local interaction terms that are each bounded by 1 in spectral norm. This is a more general notion of time than gate depth, which only counts the number of circuit layers without accounting for the time required to implement each layer. Both protocols exhibit an asymptotic time scaling of $n/3 + O(1)$, which is asymptotically three times faster than the best possible SWAP-based time of $n - 1$ (bounded by the diameter of the graph) [132]. The odd-even sort algorithm provides a nearly tight time upper bound of n [153] and will be our main point of comparison.

Routing using reversals has been studied extensively due to its applications in comparative genomics (where it is known as *sorting by reversals*) [154, 155]. References [156–158] present routing algorithms where, much like in our case, reversals have length-weighted costs. However, these models assume reversals are performed sequentially, while we assume independent reversals can be performed in parallel, where the total cost is given by the evolution time, akin to circuit depth. To our knowledge, results from the sequential case are not easily adaptable to the parallel setting and require a different approach.

Routing on paths is a fundamental building block for routing on more general graphs. For example, a two-dimensional grid graph is the Cartesian product of two path graphs, and the best known routing routine applies a path routing subroutine 3 times [132]. A quantum protocol for routing on the path in time cn, for a constant $c > 0$, would imply a routing time of $3cn$ on the grid. More generally, routing algorithms for the *generalized hierarchical product* of graphs can take advantage of faster routing of the path base graph [15].

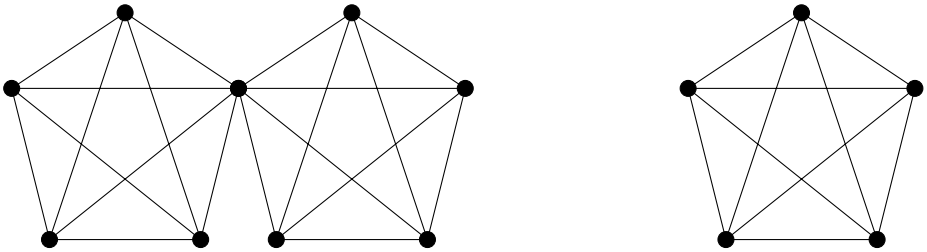
In the rest of this chapter, we present the following results on quantum routing

using fast reversals. In Sec. 4.2, we give basic examples of using fast reversals to perform routing on general graphs to indicate the extent of possible speedup over SWAP-based routing, namely a graph for which routing can be sped up by a factor of 3, and another for which no speedup is possible. Sec. 4.3 presents algorithms for routing sparse permutations, where few qubits are routed, both for paths and for more general graphs. Here, we obtain the full factor 3 speedup over SWAP-based routing. Then, in Sec. 4.4, we prove the main result that there is a quantum routing algorithm for the path with worst-case constant-factor advantage over any SWAP-based routing scheme. Finally, in Sec. 4.5, we show that our algorithm has average-case routing time $2n/3 + o(n)$ and any SWAP-based protocol has average-case routing time at least $n - o(n)$.

4.2 Simple bounds on routing using reversals

Given the ability to implement a fast reversal ρ with cost given by Eq. (4.2), the largest possible asymptotic speedup of reversal-based routing over SWAP-based routing is a factor of 3. This is because the reversal operation, which is a particular permutation, cannot be performed faster than $n/3 + o(n)$, and can be performed in time n classically using odd-even sort. As we now show, some graphs can saturate the factor of 3 speedup for general permutations, while other graphs do not admit any speedup over SWAPs.

Maximal speedup: For n odd, let K_n^* denote two complete graphs, each on $(n+1)/2$ vertices, joined at a single "junction" vertex for a total of n vertices (Fig. 4.1a). Consider a permutation on K_n^* in which every vertex is sent to the other complete subgraph, except that the junction vertex is sent to itself. To route with SWAPs, note that each vertex (other



(a) Joined graph K_9^*. **(b)** Complete graph K_5.

Figure 4.1: K_9^* admits the full factor of 3 speedup in the worst case when using reversals over SWAPs, whereas K_5 admits no speedup when using reversals over SWAPs.

than that at the junction) must be moved to the junction at least once, and only one vertex can be moved there at any time. Because there are $(n+1)/2-1$ non-junction vertices on each subgraph, implementing this permutation requires a SWAP-circuit depth of at least $n-1$.

On the other hand, any permutation on K_n^* can be implemented in time $n/3+O(1)$ using reversals. First, perform a reversal on a path that connects all vertices with opposite-side destinations. After this reversal, every vertex is on the side of its destination and the remainder can be routed in at most 2 steps [132]. The total time is at most $(n+1)/3+2$, exhibiting the maximal speedup by an asymptotic factor of 3.

No speedup: Now, consider the complete graph on n vertices, K_n (Fig. 4.1b). Every permutation on K_n can be routed in at most time 2 using SWAPs [132]. Consider implementing a 3-cycle on three vertices of K_n for $n \geq 3$ using reversals. Any reversal sequence that implements this permutation will take at least time 2. Therefore, no speedup is gained over SWAPs in the worst case.

We have shown that there exists a family of graphs that allows a factor of 3

speedup for any permutation when using fast reversals instead of SWAPs, and others where reversals do not grant any improvement. The question remains as to where the path graph lies on this spectrum. Faster routing on the path is especially desirable since this task is fundamental for routing in more complex graphs.

4.3 An algorithm for sparse permutations

We now consider routing sparse permutations, where only a small number k of qubits are to be moved. For the path, we show that the routing time is at most $n/3 + O(k^2)$. More generally, we show that for a graph of radius r, the routing time is at most $2r/3+O(k^2)$. (Recall that the radius of a graph $G = (V, E)$ is $\min_{u\in V} \max_{v\in V} \mathrm{dist}(u, v)$, where $\mathrm{dist}(u, v)$ is the distance between u and v in G.) Our approach to routing sparse permutations using reversals is based on the idea of bringing all k qubits to be permuted to the center of the graph, rearranging them, and then sending them to their respective destinations.

4.3.1 Paths

A description of the algorithm on the path, called `MiddleExchange`, appears in Algorithm 4.3.1. Fig. 4.2 presents an example of `MiddleExchange` for $k = 6$.

In Theorem 4.3.1, we prove that Algorithm 4.3.1 achieves a routing time of asymptotically $n/3$ when implementing a sparse permutation of $k = o(\sqrt{n})$ qubits on the path graph. First, let $\mathcal{S}_n$ denote the set of permutations on $\{1, \ldots, n\}$, so $|\mathcal{S}_n| = n!$. Then, for any permutation $\pi \in \mathcal{S}_n$ that acts on a set of labels $\{1, \ldots, n\}$, let π_i denote the

Input : π, a permutation

function `MiddleExchange(`π`)`**:**

identify the labels $x_1, \ldots, x_k \in [n]$ to be permuted, with $x_i < x_{i+1}$

let t be the largest index for which $x_t \leq \lfloor n/2 \rfloor$, i.e., the last label x_t left of the median

for $i = 1$ *to* $t - 1$ **:**

  perform $\rho(x_i - i + 1, x_{i+1} - 1)$

for $j = k$ *to* $t + 2$ **:**

  perform $\rho(x_j + k - j, x_{j-1} + 1)$

perform $\rho(x_t - t + 1, \lfloor n/2 \rfloor)$

perform $\rho(x_{t+1} + k - t - 1, \lfloor n/2 \rfloor + 1)$

$\bar{\rho} \leftarrow$ the sequence of all reversals so far

route the labels $x_1, \ldots, x_k$ such that after performing $\bar{\rho}$ in reverse order, each label is at its destination

perform $\bar{\rho}$ in reverse order

Algorithm 4.3.1: MiddleExchange algorithm to sort sparse permutations on the path graph. We let $\rho(i, j)$ denote a reversal on the segment starting at i and ending at j, inclusive.

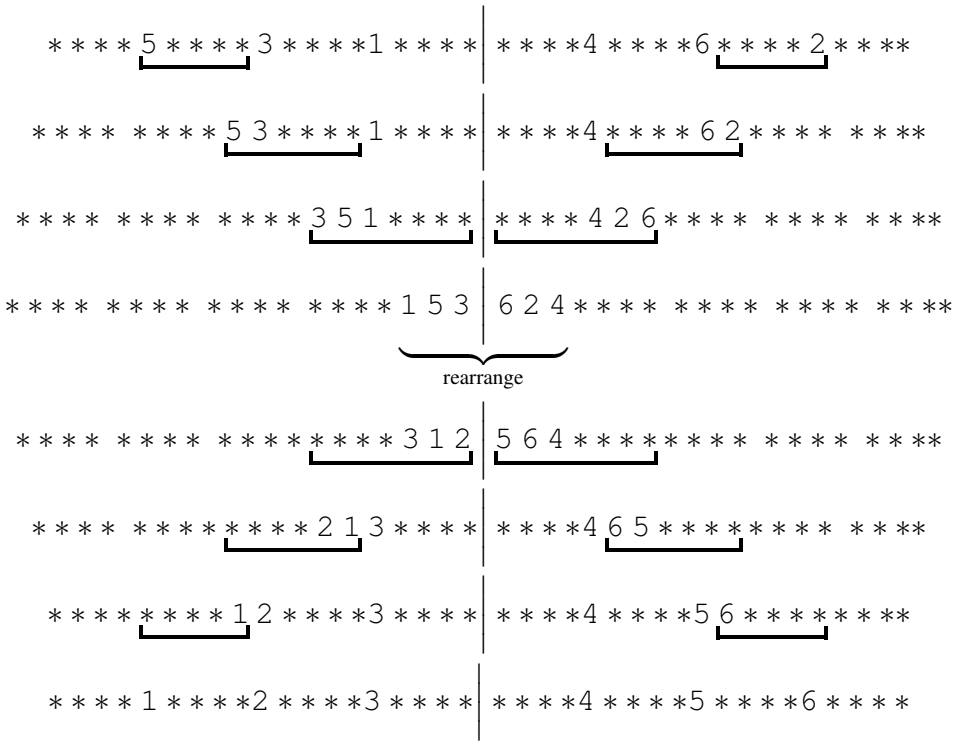


Figure 4.2: Example of `MiddleExchange` (Algorithm 4.3.1) on the path for $k = 6$.

destination of label i under π. We may then write $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$. Let $\bar{\rho}$ denote an ordered series of reversals $\rho_1, \ldots, \rho_m$, and let $\bar{\rho}_1 + \bar{\rho}_2$ be the concatenation of two reversal series. Finally, let $S \cdot \rho$ and $S \cdot \bar{\rho}$ denote the result of applying ρ and $\bar{\rho}$ to a sequence S, respectively, and let $|\rho|$ denote the length of the reversal ρ, i.e., the number of vertices it acts on.

Theorem 4.3.1. Let $\pi \in \mathcal{S}_n$ with $k = |\{x \in [n] \mid \pi_x \neq x\}|$ (i.e., k elements are to be permuted, and $n - k$ elements begin at their destination). Then Algorithm 4.3.1 routes π in time at most $n/3 + O(k^2)$.

Proof. Algorithm 4.3.1 consists of three steps: compression (Line 4–Line 9), inner permutation (Line 11), and dilation (Line 12). Notice that compression and dilation are inverses of each other.

Let us first show that Algorithm 4.3.1 routes π correctly. Just as in the algorithm, let $x_1, \ldots, x_k$ denote the labels $x \in [n]$ with $x_i < x_{i+1}$ such that $\pi_x \neq x$, that is, the elements that do not begin at their destination and need to be permuted. It is easy to see that these elements are permuted correctly: After compression, the inner permutation step routes x_i to the current location of the label π_{x_i} in the middle. Because dilation is the inverse of compression, it will then route every x_i to its correct destination. For the non-permuting labels, notice that they lie in the support of either no reversal or exactly two reversals, ρ_1 in the compression step and ρ_2 in the dilation step. Therefore ρ_1 reverses the segment containing the label and ρ_2 re-reverses it back into place (so $\rho_1 = \rho_2$). Therefore, the labels that are not to be permuted end up exactly where they started once the algorithm is complete.

Now we analyze the routing time. Let $d_i = x_{i+1} - x_i - 1$ for $i \in [k-1]$. As in the algorithm, let t be the largest index for which $x_t \leq \lfloor n/2 \rfloor$. Then, for $1 \leq i \leq t-1$, we have $|\rho_i| = d_i + i$, and, for $t+2 \leq j \leq k$, we have $|\rho_j| = d_{j-1} + k - j$. Moreover, we have $|\rho_t| = \lfloor n/2 \rfloor - x_t - 1 + t$ and $|\rho_{t+1}| = x_{t+1} - \lfloor n/2 \rfloor + k - t$. From all reversals in the first part of Algorithm 4.3.1, $\bar{\rho}$, consider those that are performed on the left side of the median (position $\lfloor n/2 \rfloor$ of the path). The routing time of these reversals is

$$\begin{aligned}
\frac{1}{3}\sum_{i=1}^{t} |\rho_i| + 1 &= \frac{1}{3}\left(\lfloor n/2 \rfloor - x_t - 1\right) + \frac{1}{3}\sum_{i=1}^{t} (d_i + i + 1) \\
&= \frac{t(t+1)}{6} + \frac{1}{3}\left(\lfloor n/2 \rfloor - x_t - 1\right) + \sum_{i=1}^{t} (x_{i+1} - x_i) \\
&= O(t^2) + \frac{1}{3}\left(\lfloor n/2 \rfloor - x_1\right) \\
&\leq \frac{n}{6} + O(k^2).
\end{aligned} \tag{4.3}$$

By a symmetric argument, the same bound holds for the compression step on the right half of the median. Because both sides can be performed in parallel, the total cost for the compression step is at most $n/6 + O(k^2)$. The inner permutation step can be done in time at most k using odd-even sort. The cost to perform the dilation step is also at most $n/6 + O(k^2)$ because dilation is the inverse of compression. Thus, the total routing time for Algorithm 4.3.1 is at most $2(n/6 + O(k^2)) + k = n/3 + O(k^2)$. □

It follows that sparse permutations on the path with $k = o(\sqrt{n})$ can be implemented using reversals with a full asymptotic factor of 3 speedup.

4.3.2 General graphs

We now present a more general result for implementing sparse permutations on an arbitrary graph.

Theorem 4.3.2. Let $G = (V, E)$ be a graph with radius r and π a permutation of vertices. Let $S = \{ v \in V : \pi_v \neq v \}$. Then π can be routed in time at most $2r/3 + O(|S|^2)$.

Proof. We route π using a procedure similar to Algorithm 4.3.1, consisting of the same three steps adapted to work on a spanning tree of G: compression, inner permutation, and dilation. Dilation is the inverse of compression and the inner permutation step can be performed on a subtree consisting of just $k = |S|$ nodes by using the ROUTING VIA MATCHINGS algorithm for trees in $3k/2 + O(\log k)$ time [152]. It remains to show that compression can be performed in $r/3 + O(k^2)$ time.

We construct a *token tree* $\mathcal{T}$ that reduces the compression step to routing on a tree. Let c be a vertex in the *center* of G, i.e., a vertex with distance at most r to all vertices. Construct a shortest-path tree $\mathcal{T}'$ of G rooted at c, say, using breadth-first search. We assign a token to each vertex in S. Now $\mathcal{T}$ is the subtree of $\mathcal{T}'$ formed by removing all vertices $v \in V(\mathcal{T}')$ for which the subtree rooted at v does not contain any tokens, as depicted in Fig. 4.3. In $\mathcal{T}$, call the first common vertex between paths to c from two distinct tokens an *intersection* vertex, and let $\mathcal{I}$ be the set of all intersection vertices. Note that if a token t_1 lies on the path from another token t_2 to c, then the vertex on which t_1 lies is also an intersection vertex. Since $\mathcal{T}$ has at most k leaves, $|\mathcal{I}| \leq k - 1$.

For any vertex v in $\mathcal{T}$, let the *descendants* of v be the vertices $u \neq v$ in $\mathcal{T}$ whose path on $\mathcal{T}$ to c includes v. Now let $\mathcal{T}_v$ be the subtree of $\mathcal{T}$ rooted at v, i.e., the tree

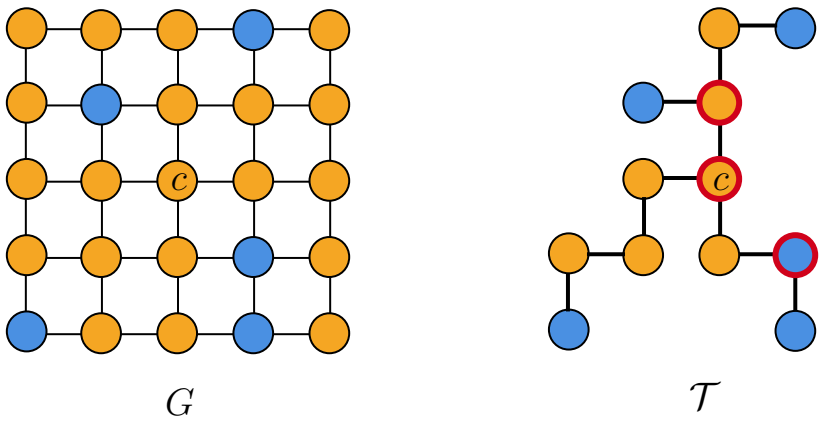


Figure 4.3: Illustration of the token tree $\mathcal{T}$ in Theorem 4.3.2 for a case where G is the 5×5 grid graph. Blue circles represent vertices in S and orange circles represent vertices not in S. Vertex c denotes the center of G. Red-outlined circles represent intersection vertices. In particular, note that one of the blue vertices is an intersection because it is the first common vertex on the path to c of two distinct blue vertices.

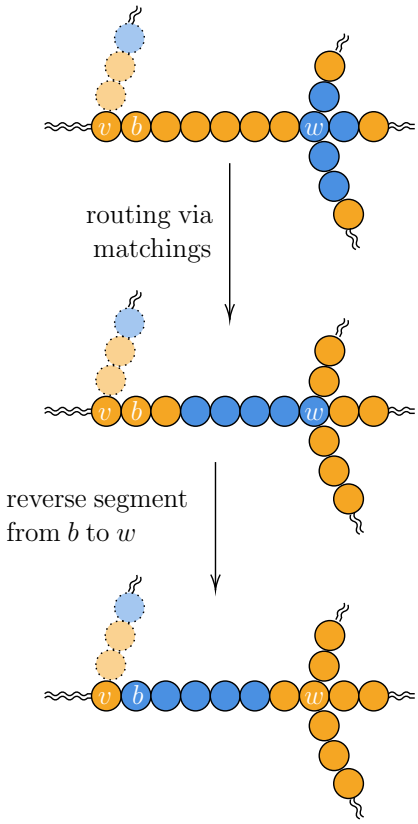
composed of v and all of the descendants of v. We say that all tokens have been *moved up to* a vertex v if for all vertices u in $\mathcal{T}_v$ without a token, T_u also does not contain a token. The compression step can then be described as moving tokens up to c.

We describe a recursive algorithm for doing so in Algorithm 4.3.2. The base case considers the trivial case of a subtree with only one token. Otherwise, we move all tokens on the subtrees of descendant b up to the closest intersection w using recursive calls as illustrated in Fig. 4.4. Afterwards, we need to consider whether the path p between v and w has enough room to store all tokens. If it does, we use a ROUTING VIA MATCHINGS algorithm for trees to route tokens from w onto p, followed by a reversal to move these tokens up to v. Otherwise, the path is short enough to move all tokens up to v by the same ROUTING VIA MATCHINGS algorithm.

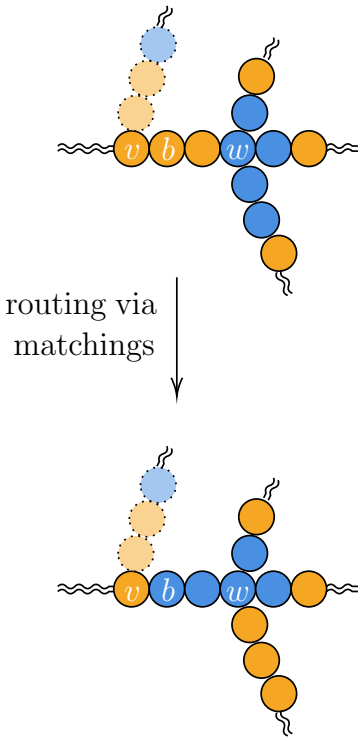
We now bound the routing time on $\mathcal{T}_{w_1}$ of `MoveUpTo(`w_1`)`, for any vertex $w_1 \in V(\mathcal{T})$. First note that all operations on subtrees $\mathcal{T}_b$ of $\mathcal{T}_{w_1}$ are independent and can be performed in parallel. Let $w_1, w_2, \ldots, w_t$ be the sequence of intersection

```
Input : A vertex v in token tree T
function MoveUpTo(v):
    if T_v contains no intersection vertices other than v then    // Base case
        for each leaf node u ∈ V(T_v) :
            if u is the first leaf node then
                Perform reversal from u to v.
            else
                Perform reversal from u to v, exclusive.
        return
    for each descendant b of v :
        w := the intersection vertex in T_b closest to b    // may include b
        MoveUpTo(w)
        m := the number of tokens from S in T_b
        l(p) := the length of the path p from w to b in T_v
        if l(p) ≥ m then    // Enough room on p, form a path of tokens at b
            Route the m tokens in T_b to the first m vertices of p using ROUTING
              VIA MATCHINGS.
            Perform a reversal on the segment starting at w and ending at b.
        else // Not enough room on p, form a tree of tokens rooted at b
            Route the m tokens in T_b as close as possible to b using ROUTING
              VIA MATCHINGS.
    if v has no token then                    // Put token on root v
        Perform a reversal on the segment starting from v and ending at a vertex u
          in T_v with a token such that no descendant of u has a token.
```

Algorithm 4.3.2: An algorithm that recursively moves all tokens from S that lie on $\mathcal{T}_v$ up to an intersection vertex v.



(a) When $l(p) \geq m$. In this case, $l(p) = 7 \geq 5 = m$.



(b) When $l(p) < m$. In this case, $l(p) = 3 < 5 = m$.

Figure 4.4: An example of moving the m tokens in $\mathcal{T}_w$ up to b (Line 14–Line 18 in Algorithm 4.3.2).

vertices that `MoveUpTo(·)` is recursively called on that dominates the routing time of `MoveUpTo(`w_1`)`. Let d_w, for $w \in V(\mathcal{T}_{w_1})$, be the distance of w to the furthest leaf node in $\mathcal{T}_w$. Assuming that the base case on Line 2 has not been reached, we have a routing time of

$$T(w_1) \leq T(w_2) + \frac{d_{w_1} - d_{w_2}}{3} + O(k), \tag{4.4}$$

where $O(k)$ bounds the time required to route $m \leq k$ tokens on a tree of size at most $2m$ following the recursive `MoveUpTo(`w_2`)` call [152]. We expand the time cost $T(w_i)$ of recursive calls until we reach the base case of w_t to obtain

$$T(v) \leq T(w_t) + \sum_{i=1}^{t-1} \left(\frac{d_{w_i} - d_{w_{i+1}}}{3} + O(k) \right) \tag{4.5}$$

$$= T(w_t) + \frac{d_{w_1} - d_{w_t}}{3} + t \cdot O(k) \leq \frac{d_{w_1}}{3} + (t+1)O(k). \tag{4.6}$$

Since $d_v \leq r$ and $t \leq k$, this shows that compression can be performed in $r/3 + O(k^2)$ time. □

In general, a graph with radius r and diameter d will have $d/2 \leq r \leq d$. Using Theorem 4.3.2, this implies that for a graph G and a sparse permutation with $k = o(\sqrt{r})$, the bound for the routing time will be between $d/3 + o(d)$ and $2d/3 + o(d)$. Thus, for such sparse permutations, using reversals will always asymptotically give us a constant-factor worst-case speedup over any SWAP-only protocol since $\mathrm{rt}(G) \geq d$. Furthermore, for graphs with $r = d/2$, we can asymptotically achieve the full factor of 3 speedup.

Input : π, a permutation of a contiguous subset of $[n]$.

function `GenericDivideConquer(BinarySorter, π)`**:**
  if $|\pi| = 1$ **then**
    return $\emptyset$
  $B := \texttt{BinaryLabeling}(\pi)$
  $\overline{\rho} := \texttt{BinarySorter}(B)$
  $\pi := \pi \cdot \overline{\rho}$
  $\overline{\rho} = \overline{\rho} + \texttt{GenericDivideConquer}(\textit{BinarySorter}, \pi[0, \lfloor \frac{n}{2} \rfloor])$
  $\overline{\rho} = \overline{\rho} + \texttt{GenericDivideConquer}(\textit{BinarySorter}, \pi[\lfloor \frac{n}{2} \rfloor + 1, |\pi|])$
  return $\overline{\rho}$

Algorithm 4.4.1: Divide-and-conquer algorithm for recursively sorting π. `BinaryLabeling`(π) is a subroutine that uses Eq. (4.7) to transform π into a bitstring, and `BinarySorter` is a subroutine that takes as input the resulting binary string and returns an ordered reversal sequence $\bar{\rho}$ that sorts it.

4.4 Algorithms for routing on the path

Our general approach to implementing permutations on the path relies on the divide-and-conquer strategy described in Algorithm 4.4.1. It uses a correspondence between implementing permutations and sorting binary strings, where the former can be performed at twice the cost of the latter. This approach is inspired by [157] and [156] who use the same method for routing by reversals in the sequential case.

First, we introduce a binary labeling using the indicator function

$$I(v) = \begin{cases} 0 & \text{if } v < n/2, \\ 1 & \text{otherwise.} \end{cases} \tag{4.7}$$

This function labels any permutation $\pi \in \mathcal{S}_n$ by the binary string $I(\pi) := (I(\pi_1), I(\pi_2), \ldots, I(\pi_n))$. Let π be the target permutation, and σ any permutation such that $I(\pi\sigma^{-1}) = (0^{\lfloor n/2 \rfloor} 1^{\lceil n/2 \rceil})$. Then it follows that σ divides π into permutations π_L, π_R

acting only on the left and right halves of the path, respectively, i.e., $\pi = \pi_L \cdot \pi_R \cdot \sigma$. We find and implement σ via a binary sorting subroutine, thereby reducing the problem into two subproblems of length at most $\lceil n/2 \rceil$ that can be solved in parallel on disjoint sections of the path. Proceeding by recursion until all subproblems are on sections of length at most 1, the only possible permutation is the identity and π has been implemented. Because disjoint permutations are implemented in parallel, the total routing time is $T(\pi) = T(\sigma) + \max(T(\pi_L), T(\pi_R))$.

We illustrate Algorithm 4.4.1 with an example, where the binary labels are indicated below the corresponding destination indices:

$$\begin{array}{c} \begin{array}{cccccccc} 7 & 6 & 0 & 2 & 5 & 1 & 3 & 4 \end{array} \xrightarrow{\text{label}} \begin{array}{cccccccc} 7 & 6 & 0 & 2 & 5 & 1 & 3 & 4 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \xrightarrow{\text{sort}} \begin{array}{cccccccc} 0 & 3 & 1 & 2 & 5 & 7 & 6 & 4 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \xrightarrow{\text{label}} \begin{array}{cccccccc} 0 & 3 & 1 & 2 & & 5 & 7 & 6 & 4 \\ 0 & 1 & 0 & 1 & & 0 & 1 & 1 & 0 \end{array} \\ \\ \downarrow \text{sort} \\ \\ \begin{array}{cccccccc} 0 & 1 & & 2 & 3 & & 4 & 5 & & 6 & 7 \\ 0 & 1 & & 0 & 1 & & 0 & 1 & & 0 & 1 \end{array} \xleftarrow{\text{sort}} \begin{array}{cccccccc} 0 & 1 & & 3 & 2 & & 5 & 4 & & 6 & 7 \\ 0 & 1 & & 1 & 0 & & 1 & 0 & & 0 & 1 \end{array} \xleftarrow{\text{label}} \begin{array}{cccccccc} 0 & 1 & 3 & 2 & & 5 & 4 & 6 & 7 \\ 0 & 0 & 1 & 1 & & 0 & 0 & 1 & 1 \end{array} \end{array} \tag{4.8}$$

Each labeling and sorting step corresponds to an application of Eq. (4.7) and `BinarySorter`, respectively, to each subproblem. Specifically, in Eq. (4.8), we use TBS (Algorithm 4.4.2) to sort binary strings.

We present two algorithms for `BinarySorter`, which perform the work in our sorting algorithm. The first of these binary sorting subroutines is Tripartite Binary Sort (TBS, Algorithm 4.4.2). TBS works by splitting the binary string into nearly equal (contiguous) thirds, recursively sorting these thirds, and merging the three sorted thirds into one sorted sequence. We sort the outer thirds forwards and the middle third backwards which allows us to merge the three segments using at most one reversal. For

```
   Input : B, a binary string
1  function TripartiteBinarySort(B):
2      if |B| = 1 then
3          return ∅
4      m_1 := ⌊|B|/3⌋
5      m_2 := ⌊2|B|/3⌋
6      ρ̄ := TripartiteBinarySort(B[0, m_1])
7      ρ̄ := ρ̄ + TripartiteBinarySort(B[m_1 + 1, m_2] ⊕ 11...1)
8      ρ̄ := ρ̄ + TripartiteBinarySort(B[m_2 + 1, |B|])
9      B ← apply reversals in ρ̄ to B
10     i := index of first 1 in B
11     j := index of last 0 in B
12     return ρ̄ + ρ(i, j)
```

Algorithm 4.4.2: Tripartite Binary Sort (TBS). We let $\rho(i, j)$ denote a reversal on the subsequence $S[i, j]$ (inclusive of i and j). In line 7, $\oplus$`11...1` indicates that we flip all the bits, so that we sort the middle third backwards.

example, we can sort a binary string as follows:

$$
\begin{array}{c}
\texttt{010011100011010011110111001} \\
\texttt{010011100}\quad\texttt{011010011}\quad\texttt{110111001} \\
\text{TBS} \downarrow \qquad \text{TBS} \downarrow \text{ backwards} \qquad \downarrow \text{TBS} \\
\texttt{000001111}\quad\texttt{111110000}\quad\texttt{000111111} \\
\texttt{00000}\underbrace{\texttt{1111111110000000}}\texttt{11111} \\
\texttt{00000000000011111111111111},
\end{array}
\tag{4.9}
$$

where the arrows with TBS indicate recursive calls to TBS and the bracket indicates the reversal to merge the segments. Let `GDC(TBS)` denote Algorithm 4.4.1 when using TBS to sort binary strings, where `GDC` stands for `GenericDivideConquer`.

The second algorithm is an adaptive version of TBS (Algorithm 4.4.3) that, instead

of using equal thirds, adaptively chooses the segments' length. Adaptive TBS considers every pair of partition points, $0 \leq i \leq j < n-1$, that would split the binary sequence into two or three sections: $B[0, i]$, $B[i+1, j]$, and $B[j+1, n-1]$ (where $i = j$ corresponds to no middle section). For each pair, it calculates the minimum cost to recursively sort the sequence using these partition points. Since each section can be sorted in parallel, the total *sorting time* depends on the maximum time needed to sort one of the three sections and the cost of the final merging reversal. Let `GDC(ATBS)` denote Algorithm 4.4.1 when using Adaptive TBS to sort binary strings.

Notice that the partition points selected by TBS are considered by the Adaptive TBS algorithm and are selected by Adaptive TBS only if no other pair of partition points yields a faster sorting time. Thus, for any permutation, the sequence of reversals found by Adaptive TBS costs no more than that found by TBS. However, TBS is simpler to implement and will be faster than Adaptive TBS in finding the sorting sequence of reversals.

4.4.1 Worst-case bounds

In this section, we prove that all permutations of sufficiently large length n can be sorted in time strictly less than n using reversals. Let $n_x(b)$ denote the number of times character $x \in \{0, 1\}$ appears in a binary string b, and let $T(b)$ (resp., $T(\pi)$) denote the best possible sorting time to sort b (resp., implement π) with reversals. Assume all logarithms are base 2 unless specified otherwise.

Lemma 4.4.1. Let $b \in \{0, 1\}^n$ such that $n_x(b) < cn + O(\log n)$, where $c \in [0, 1/3]$ and

```
   Input  : B, a binary string
 1 function AdaptiveTripartiteBinarySort(B):
 2 ρ̄ := ∅
 3 for i = 0 to n − 2 :
 4     for j = i to n − 2 :
 5         ρ̄_0 = AdaptiveTripartiteBinarySort(B[0, i])
 6         c_0 := cost(ρ̄_0)
 7         ρ̄_1 = AdaptiveTripartiteBinarySort(B[i + 1, j])
 8         c_1 := cost(ρ̄_1)
 9         ρ̄_2 = AdaptiveTripartiteBinarySort(B[j + 1, n − 1])
10         c_2 := cost(ρ̄_2)
11         r :=  cost of merging reversal using i and j as partition points
12         if ρ̄ = ∅ or max{c_0, c_1, c_2} + r < cost(ρ̄) then
13             ρ̄ := ρ̄_0 ∔ ρ̄_1 ∔ ρ̄_2
14 return ρ̄
```

Algorithm 4.4.3: Adaptive TBS. For the sake of clarity, we implement an exhaustive search over all possible ways to choose the partition points. However, we note that the optimal partition points can be found in polynomial time by using a dynamic programming method [156].

$x \in \{0, 1\}$. Then, $T(b) \leq (c/3 + 7/18)\, n + O(\log n)$.

Proof. To achieve this upper bound, we use TBS (Algorithm 4.4.2). There are $\lfloor \log_3 n \rfloor$ steps in the recursion, which we index by $j \in \{0, 1, \ldots, \lfloor \log_3 n \rfloor\}$, with step 0 corresponding to the final merging step. Let $|\rho_j|$ denote the size of the longest reversal in recursive step j that merges the three sorted subsequences of size $n/3^{j+1}$. The size of the final merging reversal ρ_0 can be bounded above by $(c + 2/3)n + O(\log n)$ because $|\rho_0|$ is maximized when every x is contained in the leftmost third if $x = 1$ or the rightmost third

if $x = 0$. So we have

$$T(b) \leq \left(\sum_{j=0}^{\log_3 n} \frac{|\rho_j|}{3} \right) + O(\log n) \leq \left(\frac{c}{3} + \frac{2}{9} \right) n + O(\log n) + \left(\sum_{j=1}^{\log_3 n} \frac{|\rho_j|}{3} \right) + O(\log n) \tag{4.10}$$

$$\leq \left(\frac{c}{3} + \frac{7}{18} \right) n + O(\log n), \tag{4.11}$$

where we used $|\rho_j| \leq n/3^j$ for $j \geq 1$. □

Now we can prove a bound on the cost of a sorting series found by Adaptive TBS for any binary string of length n.

Theorem 4.4.1. For all bit strings $b \in \{0,1\}^n$ of arbitrary length $n \in \mathbb{N}$, $T(b) \leq (1/2 - \varepsilon)\, n + O(\log n) \approx 0.483n + O(\log n)$, where $\varepsilon = 1/3 - 1/\sqrt{10}$.

Proof. Let $b \in \{0,1\}^n$ for some $n \in \mathbb{N}$. Partition b into three sections $b = b_1 b_2 b_3$ such that $|b_1| = |b_3| = \lfloor n/3 \rfloor$ and $|b_2| = n - 2\lfloor n/3 \rfloor$. Since $\lfloor n/3 \rfloor = n/3 - d$ where $d \in \{0, 1/3, 2/3\}$, we write $|b_1| = |b_2| = |b_3| = n/3 + O(1)$ for the purposes of this proof. Recall that if segments b_1 and b_3 are sorted forwards and segment b_2 is sorted backwards, the resulting segment can be sorted using a single reversal, ρ (see the example in Eq. (4.9)). Then we have

$$T(b) \leq \max(T(b_1), T'(b_2), T(b_3)) + \frac{|\rho| + 1}{3}, \tag{4.12}$$

where $T'(b_2)$ is the time to sort b_2 backwards using reversals.

We proceed by induction on n. For the base case, it suffices to note that every

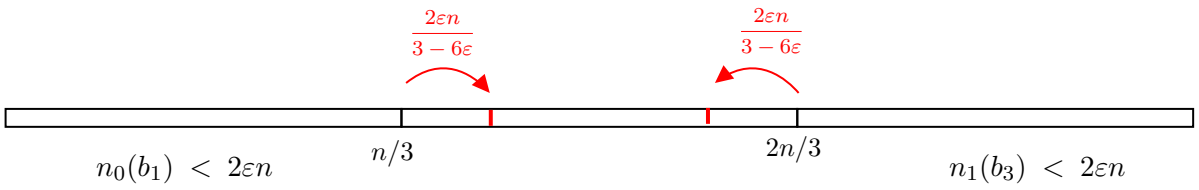


Figure 4.5: Case 2 of Theorem 4.4.1. If there are few zeros and ones in the leftmost and rightmost thirds, respectively, we can shorten the middle section so that it can be sorted quickly. Then, because each of the outer thirds contain far more zeros than ones (or vice versa), they can both can be sorted quickly as well.

binary string can be sorted using reversals and, for finitely many values of $n \in \mathbb{N}$, any time needed to sort a binary string of length n exceeding $(1/2 - \varepsilon)\, n$ can be absorbed into the $O(\log n)$ term. Now assume $T(b) \leq (1/2 - \varepsilon)\, k + O(\log k)$ for all $k < n$, $b \in \{0,1\}^k$.

Case 1: $n_0(b_1) \geq 2\varepsilon n$ **or** $n_1(b_3) \geq 2\varepsilon n$**.** In this case, $|\rho| \leq n - 2\varepsilon n$, so

$$T(b) \leq \frac{n - 2\varepsilon n + 1}{3} + \max(T(b_1), T'(b_2), T(b_3)) \leq \left(\frac{1}{2} - \varepsilon\right) n + O(\log n) \tag{4.13}$$

by the induction hypothesis.

Case 2: $n_0(b_1) < 2\varepsilon n$ **and** $n_1(b_3) < 2\varepsilon n$**.** In this case, adjust the partition such that $|b_1| = |b_3| = n/3 + 2\varepsilon n/(3 - 6\varepsilon) - O(1)$ and consequently $|b_2| = n/3 - 4\varepsilon n/(3 - 6\varepsilon) + O(1)$, as depicted in Fig. 4.5. In this adjustment, at most $2\varepsilon n/(3 - 6\varepsilon)$ zeros are added to the segment b_1 and likewise with ones to b_3. Thus, $n_1(b_3) \leq 2\varepsilon n + 2\varepsilon n/(3 - 6\varepsilon) = (1 + 1/(3 - 6\varepsilon))\, 2\varepsilon n$. Since $n = (3 - 6\varepsilon)|b_1| - O(1)$, we have

$$n_1(b_3) \leq \left(1 + \frac{1}{3 - 6\varepsilon}\right) 2\varepsilon((3 - 6\varepsilon)|b_1| - O(1)) = (2 - 3\varepsilon)4\varepsilon|b_1| - O(1). \tag{4.14}$$

Let $c = (2 - 3\varepsilon)4\varepsilon = 2/15$. Applying Lemma 4.4.1 with this value of c yields

$$T(b_3) \leq \left(\frac{2}{45} + \frac{7}{18}\right)|b_1| + O(\log(|b_1|)) = \left(\frac{1}{\sqrt{10}} - \frac{1}{6}\right)n + O(\log n). \tag{4.15}$$

Since $|b_1| = |b_3|$, we obtain the same bound $T(b_1) \leq (1/\sqrt{10} - 1/6)n + O(\log n)$ by applying Lemma 4.4.1 with the same value of c.

By the inductive hypothesis, $T'(b_2)$ can be bounded above by

$$T'(b_2) \leq \left(\frac{1}{2} - \varepsilon\right)\left(\frac{n}{3} - \frac{4\varepsilon}{3 - 6\varepsilon}n + O(1)\right) + O(\log n) = \left(\frac{1}{\sqrt{10}} - \frac{1}{6}\right)n + O(\log n). \tag{4.16}$$

Using Eq. (4.12) and the fact that $|\rho| \leq n$, we get the bound

$$T(b) \leq \left(\frac{1}{\sqrt{10}} - \frac{1}{6}\right)n + O(\log n) + \frac{n+1}{3} = \left(\frac{1}{2} - \varepsilon\right)n + O(\log n)$$

as claimed. □

This bound on the cost of a sorting series found by Adaptive TBS for binary sequences can easily be extended to a bound on the minimum sorting sequence for any permutation of length n.

Corollary 4.4.2. For a length-n permutation π, $T(\pi) \leq \left(1/3 + \sqrt{2/5}\right)n + O(\log^2 n) \approx 0.9658n + O(\log^2 n)$.

Proof. To sort π, we turn it into a binary string b using Eq. (4.7). Then let $\rho_1, \rho_2, \ldots, \rho_m$ be a sequence of reversals to sort b. If we apply the sequence to get $\pi' = \pi\rho_1\rho_2\cdots\rho_m$,

every element of π' will be on the same half as its destination. We can then recursively perform the same procedure on each half of π', continuing down until every pair of elements has been sorted.

This process requires $\lfloor \log n \rfloor$ steps, and at step i, there are 2^i binary strings of length $\frac{n}{2^i}$ being sorted in parallel. This gives us the following bound to implement π:

$$T(\pi) \leq \sum_{i=0}^{\log n} T(b_i), \tag{4.17}$$

where $b_i \in \{0,1\}^{n/2^i}$. Applying the bound from Theorem 4.4.1, we obtain

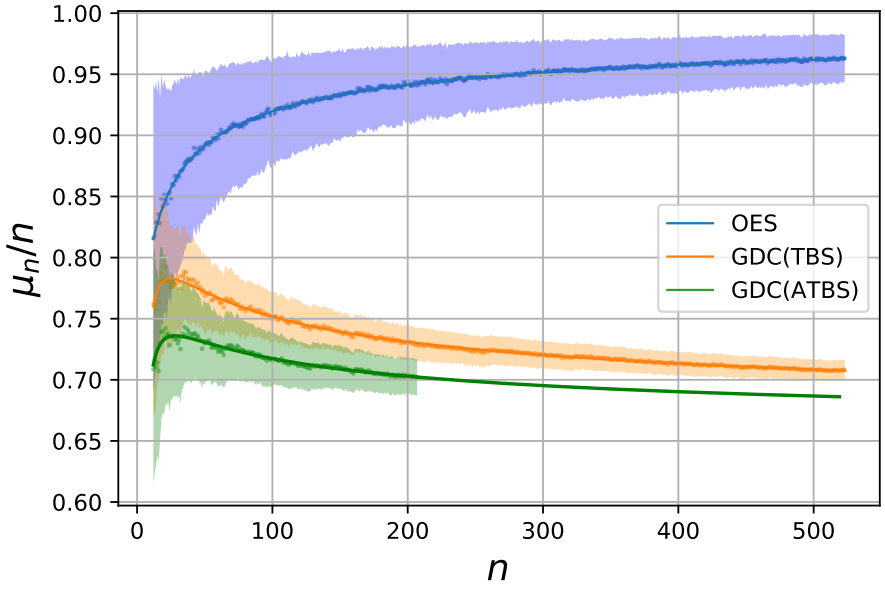
$$\begin{aligned} T(\pi) &\leq \sum_{i=0}^{\log n} T(b_i) \leq \sum_{i=0}^{\log n} \left(\left(\frac{1}{6} + \frac{1}{\sqrt{10}} \right) \frac{n}{2^i} + O(\log(n/2^i)) \right) \\ &= \left(\frac{1}{3} + \sqrt{\frac{2}{5}} \right) n + O(\log^2 n). \end{aligned}$$

□

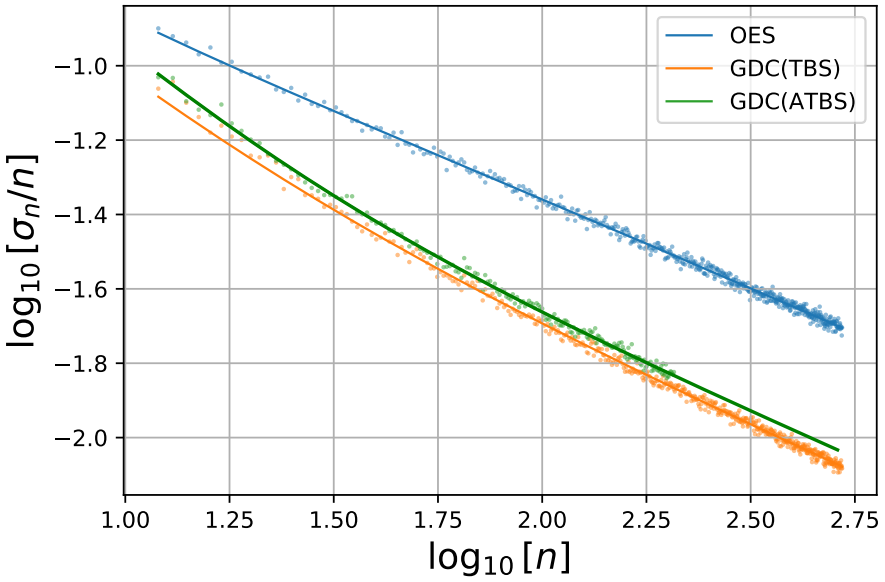
4.5 Average-case performance

So far we have presented worst-case bounds that provide a theoretical guarantee on the speedup of quantum routing over classical routing. However, the bounds are not known to be tight, and may not accurately capture the performance of the algorithm in practice.

In this section we show better performance for the *average-case* routing time, the expected routing time of the algorithm on a permutation chosen uniformly at random from $\mathcal{S}_n$. We present both theoretical and numerical results on the average routing time of swap-based routing (such as odd-even sort) and quantum routing using TBS and ATBS.



(a) Normalized mean routing time with standard deviation.



(b) Log normalized standard deviation of the routing time.

Figure 4.6: The mean routing time and fit of the mean routing time for odd-even sort (OES), and routing algorithms using Tripartite Binary Sort (`GDC(TBS)`) and Adaptive TBS (`GDC(ATBS)`). We exhaustively search for $n < 12$ and sample 1000 permutations uniformly at random otherwise. We show data for `GDC(ATBS)` only for $n \leq 207$ because it becomes too slow after that point. We find that the fit function $\mu_n = an + b\sqrt{n} + c$ fits the data with an $R^2 > 99.99\%$ (all three algorithms). For OES, the fit gives $a \approx 0.9999$; for `GDC(TBS)`, $a \approx 0.6599$; and for `GDC(ATBS)`, $a \approx 0.6513$. Similarly, for the standard deviation, we find that the fit function $\sigma_n^2 = an + b\sqrt{n} + c$ fits the data with $R^2 \approx 99\%$ (all three algorithms), suggesting that the normalized deviation of the performance about mean scales as $\sigma_n/n = \Theta(n^{-0.5})$ asymptotically.

We show that on average, `GDC(TBS)` (and `GDC(ATBS)`, whose sorting time on any instance is at least as fast) beats swap-based routing by a constant factor $2/3$. We have the following two theorems, whose proofs can be found in Secs. 4.7 and 4.8, respectively.

Theorem 4.5.1. The average routing time of any SWAP-based procedure is lower bounded by $n - o(n)$.

Theorem 4.5.2. The average routing time of `GDC(TBS)` is $2n/3 + O(n^\alpha)$ for a constant $\alpha \in \left(\frac{1}{2}, 1\right)$.

These theorems provide average-case guarantees, yet do not give information about the non-asymptotic behavior. Therefore, we test our algorithms on random permutations for instances of intermediate size.

Our numerics [159] show that Algorithm 4.4.1 has an average routing time that is well-approximated by $c \cdot n + o(n)$, where $2/3 \lesssim c < 1$, using TBS or Adaptive TBS as the binary sorting subroutine, for permutations generated uniformly at random. Similarly, the performance of odd-even sort (OES) is well-approximated by $n + o(n)$. Furthermore, the advantage of quantum routing is evident even for fairly short paths. We demonstrate this by sampling 1000 permutations uniformly from $\mathcal{S}_n$ for $n \in [12, 512]$, and running OES and `GDC(TBS)` on each permutation. Due to computational constraints, `GDC(ATBS)` was run on sample permutations for lengths $n \in [12, 206]$. On an Intel i7-6700HQ processor with a clock speed of 2.60 GHz, OES took about 0.04 seconds to implement each permutation of length 512; `GDC(TBS)` took about 0.3 seconds; and, for permutations of length 200, `GDC(ATBS)` took about 6 seconds.

The results of our experiments are summarized in Fig. 4.6. We find that the

mean normalized time costs for OES, `GDC(TBS)`, and `GDC(ATBS)` are similar for small n, but the latter two decrease steadily as the lengths of the permutations increase while the former steadily increases. Furthermore, the average costs for `GDC(TBS)` and `GDC(ATBS)` diverge from that of OES rather quickly, suggesting that `GDC(TBS)` and `GDC(ATBS)` perform better on average for somewhat small permutations ($n \approx 50$) as well as asymptotically.

The linear coefficient a of the fit of μ_n for OES is $a \approx 0.9999 \approx 1$, which is consistent with the asymptotic bound proven in Theorems 4.5.1 and 4.5.2. For the fit of the mean time costs for `GDC(TBS)` and `GDC(ATBS)`, we have $a \approx 0.6599$ and $a \approx 0.6513$ respectively. The numerics suggest that the algorithm routing times agree with our analytics, and are fast for instances of realistic size. For example, at $n = 100$, `GDC(TBS)` and `GDC(ATBS)` have routing times of $\sim 0.75n$ and $0.72n$, respectively. On the other hand, OES routes in average time $> 0.9n$. For larger instances, the speedup approaches the full factor of $2/3$ monotonically. Moreover, the fits of the standard deviations suggest $\sigma_n/n = \Theta(1/\sqrt{n})$ asymptotically, which implies that as permutation length increases, the distribution of routing times gets relatively tighter for all three algorithms. This suggests that the average-case routing time may indeed be representative of typical performance for our algorithms for permutations selected uniformly at random.

4.6 Conclusion

We have shown that our algorithm, `GDC(ATBS)` (i.e., Generic Divide-and-Conquer with Adaptive TBS to sort binary strings), uses the fast state reversal primitive to

outperform any SWAP-based protocol when routing on the path in the worst and average case. Recent work shows a lower bound on the time to perform a reversal on the path graph of n/α, where $\alpha \approx 4.5$ [18]. Thus we know that the routing time cannot be improved by more than a factor α over SWAPs, even with new techniques for implementing reversals. However, it remains to understand the fastest possible routing time on the path. Clearly, this is also lower bounded by n/α. Our work could be improved by addressing the following two open questions: (i) how fast can state reversal be implemented, and (ii) what is the fastest way of implementing a general permutation using state reversal?

We believe that the upper bound in Corollary 4.4.2 can likely be decreased. For example, in the proof of Lemma 4.4.1, we use a simple bound to show that the reversal sequence found by `GDC(TBS)` sorts binary strings with fewer than cn ones sufficiently fast for our purposes. It is possible that this bound can be decreased if we consider the reversal sequence found by `GDC(ATBS)` instead. Additionally, in the proof of Theorem 4.4.1, we only consider two pairs of partition points: one pair in each case of the proof. This suggests that the bound in Theorem 4.4.1 might be decreased if the full power of `GDC(ATBS)` could be analyzed.

Improving the algorithm itself is also a potential avenue to decrease the upper bound in Corollary 4.4.2. For example, the generic divide-and-conquer approach in Algorithm 4.4.1 focused on splitting the path exactly in half and recursing. An obvious improvement would be to create an adaptive version of Algorithm 4.4.1 in a manner similar to `GDC(ATBS)` where instead of splitting the path in half, the partition point would be placed in the optimal spot. It is also possible that by going beyond the divide-

and-conquer approach, we could find faster reversal sequences and reduce the upper bound even further.

Our algorithm uses reversals to show the first quantum speedup for unitary quantum routing. It would be interesting to find other ways of implementing fast quantum routing that are not necessarily based on reversals. Other primitives for rapidly routing quantum information might be combined with classical strategies to develop fast general-purpose routing algorithms, possibly with an asymptotic scaling advantage. Such primitives might also take advantage of other resources, such as long-range Hamiltonians or the assistance of entanglement and fast classical communication.

4.7 Average routing time using only SWAPs

In this section, we prove Theorem 4.5.1. First, define the infinity distance $d_\infty \colon \mathcal{S}_n \to \mathbb{N}$ to be $d_\infty(\pi) = \max_{1 \le i \le n} |\pi_i - i|$. Note that $0 \le d_\infty(\pi) \le n - 1$. Finally, define the set of permutations of length n with infinity distance at most k to be $B_{k,n} = \{\pi \in \mathcal{S}_n : d_\infty(\pi) \le k\}$.

The infinity distance is crucially tied to the performance of odd-even sort, and indeed, any SWAP-based routing algorithm. For any permutation π of length n, the routing time of any SWAP-based algorithm is bounded below by $d_\infty(\pi)$, since the element furthest from its destination must be swapped at least $d_\infty(\pi)$ times, and each of those SWAPs must occur sequentially. To show that the average routing time of any SWAP-based protocol is asymptotically at least n, we first show that $|B_{(1-\varepsilon)n,n}|/n! \to 0$ for all $0 < \varepsilon \le 1/2$.

Schwartz and Vontobel [160] present an upper bound on $|B_{k,n}|$ that was proved

in [161] and [162]:

Lemma 4.7.1. For all $0 < r < 1$, $|B_{rn,n}| \leq \Phi(rn, n)$, where

$$\Phi(k,n) = \begin{cases} ((2k+1)!)^{\frac{n-2k}{2k+1}} \prod_{i=k+1}^{2k} (i!)^{2/i} & \text{if } \ 0 < k/n \leq \frac{1}{2} \\ (n!)^{\frac{2k+2-n}{n}} \prod_{i=k+1}^{n-1} (i!)^{2/i} & \text{if } \ \frac{1}{2} \leq k/n < 1. \end{cases} \tag{4.18}$$

Proof. Note that $r = k/n$. For the case of $0 < r \leq 1/2$, refer to [161] for a proof. For the case of $1/2 \leq r < 1$, refer to [162] for a proof. □

Lemma 4.7.2.

$$n! = \Theta\left(\sqrt{n}\left(\frac{n}{e}\right)^n\right) \tag{4.19}$$

Proof. This follows from well-known precise bounds for Stirling's formula:

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n+1}} \leq n! \leq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}} \tag{4.20}$$

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \leq n! \leq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e \tag{4.21}$$

(see for example [163]). □

With Lemmas 4.7.1 and 4.7.2 in hand, we proceed with the following theorem:

Theorem 4.7.1. For all $0 < \varepsilon \leq 1/2$, $\lim_{n\to\infty} |B_{(1-\varepsilon)n,n}|/n! = 0$. In other words, the proportion of permutations of length n with infinity distance less than $(1-\epsilon)n$ vanishes asymptotically.

Proof. Lemma 4.7.1 implies that $|B_{(1-\varepsilon)n,n}|/n! \leq \Phi((1-\varepsilon)n, n)/n!$. The constraint

$0 < \varepsilon \leq 1/2$ stipulates that we are in the regime where $1/2 \leq r < 1$, since $r = 1 - \varepsilon$. Then we use Lemma 4.7.2 to simplify any factorials that appear. Substituting Eq. (4.18) and simplifying, we have

$$\frac{\Phi\left((1-\varepsilon)n, n\right)}{n!} = \frac{\prod_{i=(1-\varepsilon)n+1}^{n-1} (i!)^{2/i}}{(n!)^{2\varepsilon - 2/n}} = O\left(\frac{e^{2\varepsilon n-2}}{n^{2\varepsilon n-2}} \prod_{i=(1-\varepsilon)n+1}^{n-1} \frac{i^{2+1/i}}{e^2}\right). \tag{4.22}$$

We note that $i^{1/i}$ terms can be bounded by

$$\prod_{i=(1-\varepsilon)n+1}^{n-1} i^{\frac{1}{i}} \leq \prod_{i=(1-\varepsilon)n+1}^{n-1} n^{\frac{1}{(1-\varepsilon)n}} \leq n^{\frac{\varepsilon}{1-\varepsilon}} \leq n \tag{4.23}$$

since $\varepsilon \leq 1/2$. Now we have

$$\begin{aligned}
O\left(\frac{e^{2\varepsilon n-2}}{n^{2\varepsilon n-2}} \prod_{i=(1-\varepsilon)n+1}^{n-1} \frac{i^{2+1/i}}{e^2}\right) &= O\left(\frac{n}{n^{2\varepsilon n-2}} \prod_{i=(1-\varepsilon)n+1}^{n-1} i^2\right) && (4.24)\\
&= O\left(\frac{n}{n^{2\varepsilon n-2}} \left(\frac{(n-1)!}{((1-\varepsilon)n+1)!}\right)^2\right) && (4.25)\\
&= O\left(\frac{n}{n^{2\varepsilon n-2} e^{2\varepsilon n}} \frac{(n-1)^{2n-1}}{((1-\varepsilon)n+1)^{2(1-\varepsilon)n+2}}\right) && (4.26)\\
&= O\left(\frac{n}{n^{2\varepsilon n-2} e^{2\varepsilon n}} \frac{n^{2n}}{((1-\varepsilon)n)^{2(1-\varepsilon)n}}\right) && (4.27)\\
&= O\left(\frac{n^3}{\exp\left((\ln(1-\varepsilon)(1-\varepsilon)+\varepsilon)2n\right)}\right). && (4.28)
\end{aligned}$$

Since $\ln(1-\varepsilon)(1-\varepsilon) + \varepsilon > 0$ for $\varepsilon > 0$, this vanishes in the limit of large n. □

Now we prove the theorem.

Proof of Theorem 4.5.1. Let $\bar{T}$ denote the average routing time of any SWAP-based protocol. Consider a random permutation π drawn uniformly from $\mathcal{S}_n$. Due to

Theorem 4.7.1, π will belong in $B_{(1-\varepsilon)n,n}$ with vanishing probability, for all $0 < \varepsilon \leq 1/2$. Therefore, for any fixed $0 < \varepsilon \leq 1/2$ as $n \to \infty$, $(1-\varepsilon)n < \mathbb{E}\left[d_\infty(\pi)\right]$. This translates to an average routing time of at least $n - o(n)$ because we have, asymptotically, $(1-\varepsilon)n \leq \bar{T}$ for all such ε. □

4.8 Average routing time using TBS

In this section, we prove Theorem 4.5.2, which characterizes the average-case performance of TBS (Algorithm 4.4.2). This approach consists of two steps: a recursive call on three equal partitions of the path (of length $n/3$ each), and a merge step involving a single reversal.

We denote the uniform distribution over a set S as $\mathcal{U}(S)$. The set of all n-bit strings is denoted $\mathbb{B}^n$, where $\mathbb{B} = \{0, 1\}$. Similarly, the set of all n-bit strings with Hamming weight k is denoted $\mathbb{B}^n_k$. For simplicity, assume that n is even. We denote the runtime of TBS on $b \in \mathbb{B}^n$ by $T(b)$.

When running `GDC(TBS)` on a given permutation π, the input bit string for TBS is $b = I(\pi)$, where the indicator function I is defined in Eq. (4.7). We wish to show that, in expectation over all permutations π, the corresponding bit strings are quick to sort. First, we show that it suffices to consider uniformly random sequences from $\mathbb{B}^n_{n/2}$.

Lemma 4.8.1. If $\pi \sim \mathcal{U}(\mathcal{S}_n)$, then $I(\pi) \sim \mathcal{U}(\mathbb{B}^n_{n/2})$.

Proof. We use a counting argument. The number of permutations π such that $I(\pi) \in \mathbb{B}^n_{n/2}$ is $(n/2)!(n/2)!$, since we can freely assign index labels from $\{1, 2, \ldots, n/2\}$ to the 0 bits of $I(\pi)$, and from $\{n/2 + 1, \ldots, n\}$ to the 1 bits of $I(\pi)$. Therefore, for a uniformly

random π and arbitrary $b \in \mathbb{B}^n_{n/2}$,

$$\Pr(I(\pi) = b) = \frac{(n/2)!(n/2)!}{n!} = \frac{1}{\binom{n}{n/2}} = \frac{1}{|\mathbb{B}^n_{n/2}|}. \tag{4.29}$$

Therefore, $I(\pi) \sim \mathcal{U}(\mathbb{B}^n_{n/2})$. □

While $\mathbb{B}^n_{n/2}$ is easier to work with than $\mathcal{S}_n$, the constraint on the Hamming weight still poses an issue when we try to analyze the runtime recursively. To address this, Lemma 4.8.2 below shows that relaxing from $\mathcal{U}(\mathbb{B}^n_{n/2})$ to $\mathcal{U}(\mathbb{B}^n)$ does not affect expectation values significantly.

We give a recursive form for the runtime of TBS. We use the following convention for the substrings of an arbitrary n-bit string a: if a is divided into 3 segments, we label the segments $a_{0.0}, a_{0.1}, a_{0.2}$ from left to right. Subsequent thirds are labeled analogously by ternary fractions. For example, the leftmost third of the middle third is denoted $a_{0.10}$, and so on. Then, the runtime of TBS on string a can be bounded by

$$T(a) \leq \max_{i \in \{0,1,2\}} T(a_{0.i}) + \frac{n_1(a_{0.0}) + n_1(\overline{a_{0.2}}) + n/3 + 1}{3}, \tag{4.30}$$

where $\overline{a}$ is the bitwise complement of bit string a and $n_1(a)$ denotes the Hamming weight of a. Logically, the first term on the right-hand side is a recursive call to sort the thirds, while the second term is the time taken to merge the sorted subsequences on the thirds using a reversal. Each term $T(a_{0.i})$ can be broken down recursively until all subsequences

are of length 1. This yields the general formula

$$T(b) \leq \frac{1}{3}\left(\sum_{r=1}^{\lceil \log_3(n) \rceil} \max_{i \in \{0,1,2\}^{r-1}} \{n_1(a_{0.i0}) + n_1(\overline{a_{0.i2}})\} + n/3^r + 1\right), \tag{4.31}$$

where $i \in \emptyset$ indicates the empty string.

Lemma 4.8.2. Let $a \sim \mathcal{U}(\mathbb{B}^n)$ and $b \sim \mathcal{U}(\mathbb{B}^n_{n/2})$. Then

$$\mathbb{E}[T(b)] \leq \mathbb{E}[T(a)] + \widetilde{O}(n^\alpha) \tag{4.32}$$

where $\alpha \in (\frac{1}{2}, 1)$ is a constant.

The intuition behind this lemma is that by the law of large numbers, the deviation of the Hamming weight from $n/2$ is subleading in n, and the TBS runtime does not change significantly if the input string is altered in a subleading number of places.

Proof. Consider an arbitrary bit string a, and apply the following transformation. If $n_1(a) = k \geq n/2$, then flip $k - n/2$ ones chosen uniformly randomly to zero. If $k < n/2$, flip $n/2 - k$ zeros to ones. Call this stochastic function $f(a)$. Then, for all a, $f(a) \in \mathbb{B}^n_{n/2}$, and for a random string $a \sim \mathcal{U}(\mathbb{B}^n)$, we claim that $f(a) \sim \mathcal{U}(\mathbb{B}^n_{n/2})$. In other words, f maps the uniform distribution on $\mathbb{B}^n$ to the uniform distribution on $\mathbb{B}^n_{n/2}$.

We show this by calculating the probability $\Pr(f(a) = b)$, for arbitrary $b \in \mathbb{B}^n_{n/2}$. A string a can map to b under f only if a and b disagree in the same direction: if, WLOG, $n_1(a) \geq n_1(b)$, then a must take value 1 wherever a, b disagree (and 0 if $n_1(a) \leq n_1(b)$). We denote this property by $a \succeq b$. The probability of picking a uniformly random a such that $a \succeq b$ with x disagreements between them is $\binom{n/2}{x}$, since $n_0(b) = n/2$. Next, the

probability that f maps a to b is $\binom{n/2+x}{x}$. Combining these, we have

$$\Pr(f(a)=b) = \sum_{x=-n/2}^{n/2} \Pr\Big(a \succeq b,\, n_1(a) = \frac{n}{2}+x\Big) \cdot \Pr\Big(f(a)=b \mid a \succeq b,\, n_1(a) = \frac{n}{2}+x\Big), \tag{4.33}$$

$$= \sum_{x=-n/2}^{n/2} \frac{\binom{n/2}{|x|}}{2^n} \cdot \frac{1}{\binom{n/2+|x|}{|x|}}, \tag{4.34}$$

$$= \frac{1}{\binom{n}{n/2}} \sum_{x=-n/2}^{n/2} \frac{\binom{n}{n/2-x}}{2^n}, \tag{4.35}$$

$$= \frac{1}{\binom{n}{n/2}} = \frac{1}{|\mathbb{B}^n_{n/2}|}. \tag{4.36}$$

Therefore, $f(a) \sim \mathcal{U}(\mathbb{B}^n_{n/2})$. Thus, f allows us to simulate the uniform distribution on $\mathbb{B}^n_{n/2}$ starting from the uniform distribution on $\mathbb{B}^n$.

Now we bound the runtime of TBS on $f(a)$ in terms of the runtime on a fixed a. Fix some $\alpha \in (\frac{1}{2}, 1)$. We know that $n_1(f(a)) = n/2$, and suppose $|n_1(a) - n/2| \leq n^\alpha$. Since $f(a)$ differs from a in at most n^α places, then at level r of the TBS recursion (see Eq. (4.31)), the runtimes of a and $f(a)$ differ by at most $1/3 \cdot \min\{2n/3^r, n^\alpha\}$. This is because the runtimes can differ by at most two times the length of the subsequence.

Therefore, the total runtime difference is bounded by

$$\Delta T \le \frac{1}{3}\sum_{r=1}^{\lceil \log_3(n)\rceil} \min\Big\{\frac{2n}{3^r}, n^\alpha\Big\}, \tag{4.37}$$

$$= \frac{1}{3}\left(\sum_{r=1}^{\lceil \log_3(2n^{1-\alpha})\rceil} n^\alpha + 2\sum_{r=\lceil \log_3(2n^{1-\alpha})\rceil+1}^{\lceil \log_3(n)\rceil} \frac{n}{3^r}\right), \tag{4.38}$$

$$= \frac{1}{3}\left(n^\alpha \log(2n^\alpha/3) + 2\sum_{s=0}^{\lfloor \log_3(n^\alpha/2)\rfloor - 1} 3^s\right) \tag{4.39}$$

$$= \frac{1}{3}\left(n^\alpha \log(2n^\alpha/3) + n^\alpha/2 - 1\right) = \widetilde{O}(n^\alpha). \tag{4.40}$$

On the other hand, if $|n_1(a) - n/2| \ge n^\alpha/2$, we simply bound the runtime by that of OES, which is at most n.

Now consider $a \sim \mathcal{U}(\mathbb{B}^n)$ and $b = f(a) \sim \mathcal{U}(\mathbb{B}^n_{n/2})$. Since $n_1(a)$ has the binomial distribution $\mathcal{B}(n, 1/2)$, where $\mathcal{B}(k, p)$ is the sum of k Bernoulli random variables with success probability p, the Chernoff bound shows that deviation from the mean is exponentially suppressed, i.e.,

$$\Pr(|n_1(a) - n/2| \ge n^\alpha) = \exp(-O(n^{2\alpha-1})). \tag{4.41}$$

Therefore, the deviation in the expectation values is bounded by

$$|\mathbb{E}[T(f(a))] - \mathbb{E}[T(a)]| \le n\exp(-O(n^{2\alpha-1})) + c(1 - \exp(-O(n^{2\alpha-1})))n^\alpha \log(n) \tag{4.42}$$

$$= \widetilde{O}(n^\alpha), \tag{4.43}$$

where c is a constant. Finally, we conclude that

$$\mathbb{E}[T(b)] \leq \mathbb{E}[T(a)] + \widetilde{O}(n^{\alpha}) \tag{4.44}$$

as claimed. □

Next, we prove the main result of this section, namely, that the runtime of `GDC(TBS)` is $2n/3$ up to additive subleading terms.

Proof of Theorem 4.5.2. We first prove properties for sorting a random n-bit string $a \sim \mathcal{U}(\mathbb{B}^n)$ and then apply this to the problem of sorting $b \sim \mathcal{U}(\mathbb{B}^n_{n/2})$ using Lemmas 4.8.1 and 4.8.2.

The expected runtime for TBS can be calculated using the recursive formula in Eq. (4.31):

$$\mathbb{E}[T(a)] \leq \frac{1}{3}\left(\sum_{r=1}^{\log_3(n)} \mathbb{E}\left[\max_{i\in\{0,1,2\}^{r-1}}\{n_1(a_{0.i0}) + n_1(\overline{a_{0.i2}})\}\right] + n/3^r + 1\right). \tag{4.45}$$

The summand contains an expectation of a maximum over Hamming weights of i.i.d. uniformly random substrings of length $n/3^r$, which is equivalent to a binomial distribution $\mathcal{B}(n/3^r, 1/2)$ where we have $n/3^r$ Bernoulli trials with success probability $1/2$. Because of independence, if we sample $X_1, X_2 \sim \mathcal{B}(n/3^r, 1/2)$, then $X_1 + X_2 \sim \mathcal{B}(2n/3^r, 1/2)$. Using Lemma 4.8.3 with $m = 3^{r-1}$, the expected maximum can be bounded by

$$\frac{n}{3^r} + O\left(\sqrt{(n/3^r)\log(3^{r-1}n/3^r)}\right) = \frac{n}{3^r} + \widetilde{O}\left(n^{1/2}\right) \tag{4.46}$$

since the second term is largest when $r = O(1)$. Therefore,

$$\mathbb{E}[T(a)] \leq \frac{1}{3}\left(\sum_{r=1}^{\log_3(n)} \frac{2n}{3^r}\right) + \widetilde{O}\big(n^{1/2}\big) = \frac{n}{3} + \widetilde{O}\big(n^{1/2}\big). \tag{4.47}$$

Lemma 4.8.2 then gives $\mathbb{E}[T(b)] \leq \frac{n}{3} + \widetilde{O}(n^{\alpha})$.

The routing algorithm `GDC(TBS)` proceeds by calling TBS on the full path, and then in parallel on the two disjoint sub-paths of length $n/2$. We show that the distributions of the left and right halves are uniform if the input permutation is sampled uniformly as $\pi \sim \mathcal{U}(\mathcal{S}_n)$. There exists a bijective mapping g such that $g(\pi) = (b, \pi_L, \pi_R) \in \mathbb{B}^n_{n/2} \times \mathcal{S}_{n/2} \times \mathcal{S}_{n/2}$ for any $\pi \in \mathcal{S}_n$ since

$$|*|\,\mathcal{S}_n = n! = \binom{n}{n/2}\left(\frac{n}{2}\right)!\left(\frac{n}{2}\right)! = |*|\,\mathbb{B}^n_{n/2} \times \mathcal{S}_{n/2} \times \mathcal{S}_{n/2}. \tag{4.48}$$

In particular, g can be defined so that b specifies which entries are taken to the first $n/2$ positions—say, without changing the relative ordering of the entries mapped to the first $n/2$ positions or the entries mapped to the last $n/2$ positions—and π_L and π_R specify the residual permutations on the first and last $n/2$ positions, respectively. Given $g(\pi) = (b, \pi_L, \pi_R)$, TBS only has access to b. After sorting, TBS can only perform deterministic permutations $\mu_L(b), \mu_R(b) \in \mathcal{S}_{n/2}$ on the left and right halves, respectively, that depend only on b. Thus TBS performs the mappings $\pi_L \mapsto \pi_L \circ (\mu_L(b))$ and $\pi_R \mapsto \pi_R \circ (\mu_R(b))$ on the output. Now it is easy to see that when $\pi_L, \pi_R \sim \mathcal{U}(\mathcal{S}_{n/2})$, the output is also uniform because the TBS mapping is independent of the relative permutations on the left and right halves.

More generally, we see that a uniform distribution over permutations $\mathcal{U}(\mathcal{S}_n)$ is mapped to two uniform permutations on the left and right half, respectively. Symbolically, for, $\pi \sim \mathcal{U}(\mathcal{S}_n)$, we have that

$$g(\pi) = (b, \pi_L, \pi_R) \sim \mathcal{U}(\mathbb{B}^n_{n/2} \times \mathcal{S}_{n/2} \times \mathcal{S}_{n/2}) = \mathcal{U}(\mathbb{B}^n_{n/2}) \times \mathcal{U}(\mathcal{S}_{n/2}) \times \mathcal{U}(\mathcal{S}_{n/2}). \quad (4.49)$$

As shown earlier, given uniform distributions over left and right permutations, the output is also uniform. By induction, all permutations in the recursive steps are uniform.

We therefore get a sum of expected TBS runtime on bit strings of lengths $n/3^r$, i.e.,

$$\sum_{r=1}^{\log_2 n} \mathbb{E}[T(b_r)] \leq \sum_{r=1}^{\log_2 n} \mathbb{E}[T(a_r)] + \widetilde{O}\Big(\Big(\frac{n}{2^{r-1}}\Big)^{\alpha}\Big) \leq \frac{2n}{3} + \widetilde{O}(n^{\alpha}) \quad (4.50)$$

where, by Lemma 4.8.1 and the uniformity of permutations in recursive calls, we need only consider $b_r \sim \mathcal{U}(\mathbb{B}^{n/2^r}_{n/2^{r-1}})$ and we bound the expected runtime using Lemma 4.8.2 with $a_r \sim \mathcal{U}(\mathbb{B}^{n/2^{r-1}})$. □

We end with a lemma about the order statistics of binomial random variables used in the proof of the main theorem.

Lemma 4.8.3. Given m i.i.d. samples from the binomial distribution $X_i \sim B(n, p)$ with $i \in [m]$, and $p \in [0, 1]$, the maximum $Y = \max_i X_i$ satisfies

$$\mathbb{E}[Y] < pn + O\left(\sqrt{n \log(mn)}\right). \quad (4.51)$$

Proof. We use Hoeffding's inequality for the Bernoulli random variable $X \sim \mathcal{B}(n, p)$,

which states that

$$\Pr(X \geq (p+\epsilon)n) \leq \exp(-2n\epsilon^2) \quad \forall \varepsilon \geq 0. \tag{4.52}$$

Pick $\epsilon = \sqrt{\frac{c}{2n}\log(mn)}$, where $c > 0$ is a constant. For this choice, we have

$$\Pr(X_i \geq (p+\epsilon)n) \leq \left(\frac{1}{mn}\right)^c \tag{4.53}$$

for every $i = 1, \ldots, m$. Then the probability that $Y < (p+\epsilon)n$ is identical to the probability that $X_i < (p+\epsilon)n$ for every i, which for i.i.d X_i is given by

$$\Pr(Y < (p+\epsilon)n) = \Pr(X < (p+\epsilon)n)^m > \left(1 - \frac{1}{(mn)^c}\right)^m. \tag{4.54}$$

Using Bernoulli's inequality ($(1+x)^r \geq 1 + rx$ for $x \geq -1$), we can simplify the above bound to

$$\Pr(Y < (p+\epsilon)n)^m > 1 - m^{1-c}n^{-c}. \tag{4.55}$$

Finally, we bound the expected value of Y by an explicit weighted sum over its range:

$$\mathbb{E}[Y] = \sum_{k=0}^{n} \Pr(Y = k) \cdot k \tag{4.56}$$

$$= \sum_{k=0}^{\lfloor (p+\epsilon)n \rfloor} \Pr(Y = k) \cdot k + \sum_{k=\lfloor (p+\epsilon)n \rfloor + 1}^{n} \Pr(Y = k) \cdot k \tag{4.57}$$

$$\leq \sum_{k=0}^{\lfloor (p+\epsilon)n \rfloor} \Pr(Y = k)) \cdot k + n \cdot \sum_{k=\lfloor (p+\epsilon)n \rfloor + 1}^{n} \Pr(Y = k) \tag{4.58}$$

$$\leq \sum_{k=0}^{\lfloor (p+\epsilon)n \rfloor} \Pr(Y = k) \cdot k + (mn)^{1-c} \tag{4.59}$$

$$\leq (p + \epsilon)n + (mn)^{1-c}. \tag{4.60}$$

Since $(mn)^{1-c} < 1$ for $c > 1$,

$$\mathbb{E}[Y] < \left\lceil pn + 1 + \sqrt{\frac{cn}{2} \log(mn)} \right\rceil = pn + O(\sqrt{n \log(mn)}) \tag{4.61}$$

as claimed. □

Chapter 5: Bang-bang control as a design principle for classical and quantum optimization algorithms

As quantum computing enters the so-called NISQ era [4], some focus has started shifting to noisy, shallow digital computations, and a need to re-examine existing quantum heuristic algorithms has emerged. The quantum adiabatic optimization algorithm (QAO), introduced in the previous decade [26], provides a paradigm for quantum speedups in optimization problems, where one performs a quasistatic Schrödinger evolution from an initial quantum state into the ground state of computational or physical interest. Runtime bounds for QAO typically depend, via adiabatic theorems, on the minimum spectral gap between the ground state and first excited state.

The Quantum Approximate Optimization Algorithm (QAOA) provides an alternative framework to designing quantum optimization algorithms, which is based on parameterized families of quantum circuits with adjustable parameters [28, 164]. Such *variational* circuits are parameterized by a depth, an initial quantum state, and a set of Hamiltonian operators under which the state can evolve. An instance of a variational circuit is further specified by a series of (labeled) evolution times that determine which operator is applied and for how long. Along with QAOA, several other recent models of heuristic computation fit into the variational circuit paradigm [165–169].

A primary distinguishing feature between the quasistatic paradigm of QAO and simulated annealing (SA) and the variational circuit paradigm is in the design of their evolution schedules, from quasistatic to a rapidly switching, or *bang-bang*, schedule. Recently, it was observed [24, 25] that the Pontryagin Minimum Principle [23] implies that variational methods that employ a bang-bang evolution schedule are sufficient for optimality of the optimization protocol. Furthermore, the paper that introduces QAOA [28] also gives evidence pointing to an exponential speedup between QAOA and QAO. This raises two questions: Firstly, can a design shift from quasistatic to bang-bang yield provable superpolynomial improvements in the runtime, or are the two frameworks polynomially equivalent? Secondly, can the same control theoretic reasoning be applied to the design of classical optimization algorithms? In this work, we answer these questions by studying the performance of bang-bang controlled algorithms on certain well-studied instances, and make comparisons to the quasistatic, annealing-type algorithms. We prove that, on these instances, going from quasistatic scheduling to bang-bang can bring about an exponential speedup for both classical and quantum optimization. We also discuss the applicability and potential limitations of the optimal control framework to the problem of designing heuristic optimization algorithms.

5.1 Summary of results

The main results of this chapter may be found in Sec. 5.7, where we study the performance of four candidate algorithms given in Table 5.2 on two benchmarking instances, and find that the bang-bang control algorithms exponentially outperform both

classical and quantum annealing-based algorithms. These results are also summarized in Table 5.1.

Instance	Annealing-based		Bang-bang	
	QAO	SA	QAOA	BBSA
`Bush`, $\lambda \geq 1$	poly(n) [170]	exp(n) [170]	$O(1)$	$\tilde{O}\left(n^{3.5\ldots}\right)$
`Bush`, $\lambda < 1$	exp(n) [170]	exp(n) [170]	$O(1)$	$\tilde{O}\left(n^{3.5\ldots}\right)$
`Spike`, $2a + b \leq 1$	poly(n) [171]	exp(n) [170]	$O(1)$	$O(n)$
`Spike`, $2a + b > 1$	exp(n) [171]	exp(n) [170]	$O(1)$	$O(n)$

Table 5.1: Performance of the four algorithms, summarized. This work gives new results for QAOA (Sec. 5.7.3) and BBSA (Sec. 5.7.2). For the two instances studied, we distinguish different parameter regimes. For the `Bush` instance, the performance of QAO depends on the choice of mixer B_λ (see Eq. 5.23). For `Spike`, the QAO performance depends on spike parameters a and b. We see that bang-bang control algorithms outperform their (quantum and classical) annealing-based counterparts for these instances. Sources for existing results are cited, and the new contributions are referenced by the relevant sections.

In addition, we study the performance of single-round QAOA (or QAOA1) on a more general class of symmetric cost functions, and give sufficient conditions under which QAOA1 can successfully find minima for these functions. These results are stated in Lemma 5.7.1 and Theorem 5.7.1. In Sec. 5.5, we elaborate on the theoretical motivation behind choosing a bang-bang schedule and the caveats therein.

5.2 Preliminaries

First, we present some notation that will be used throughout the chapter. Any problem instance of size n will be given as a constraint satisfaction problem on Boolean strings of length n. An n-bit string will be expressed as a boldfaced variable, e.g. $\mathbf{z} \in \{0,1\}^n$, in analogy with vector quantities. Variables denoting bits of a string will be expressed in normal font (e.g. the i-th bit of $\mathbf{z}$ is z_i). Similarly, the Hamming weight

of a string, which is defined as the (integer) 1-norm of the bit string, or the number of 1's in a bit string,

$$|\mathbf{z}| := \sum_{i=1}^{n} z_i \tag{5.1}$$

will also be represented by non-bold letters such as w, v to indicate that it is a scalar quantity like the value of a bit.

We will be interested in expressing states by labels such as a string variable $\mathbf{z}$, or scalar variables w, z, etc. In either case, the convention will be to use l_2-normalized kets $|\cdot\rangle$, or l_1-normalized vectors, for which we will use the notation $|\cdot)$. In particular, a state labeled by Hamming weight w will denote the equal superposition over all bit strings with that Hamming weight,

$$|w\rangle := \frac{1}{\sqrt{\binom{n}{w}}} \sum_{|\mathbf{z}|=w} |\mathbf{z}\rangle\,, \quad |w) := \frac{1}{\binom{n}{w}} \sum_{|\mathbf{z}|=w} |\mathbf{z}) \tag{5.2}$$

Problem instances are given as a cost function on bit strings,

$$c : \{0,1\}^n \to \mathbb{Z} \tag{5.3}$$

$$c(\mathbf{z}) = \text{cost of bit string } \mathbf{z}. \tag{5.4}$$

There is a natural Hamiltonian operator C (and corresponding unitary $\mathcal{C}$) associated with this function that is diagonal in the computational basis, with eigenvalue $c(\mathbf{z})$ for every

corresponding eigenvector $\mathbf{z} \in \{0,1\}^n$. Explicitly,

$$C := \sum_{\mathbf{z}\in\{0,1\}^n} c(\mathbf{z}) \left|\mathbf{z}\right\rangle \left\langle \mathbf{z}\right|, \quad \mathcal{C}(\gamma) := e^{-i\gamma C} \tag{5.5}$$

Classical n-bit strings are naturally representable as vertices of an n-dimensional hypercube graph. This is often the representation of choice, as walks on the hypercube are generated by sequences of bit flips on the string, which correspond to the 1-local quantum operator

$$B := -\sum_{i=1}^{n} X_i, \quad \mathcal{B}(\beta) := e^{-i\beta B} \tag{5.6}$$

where $X_i \equiv \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{i-1} \otimes X \otimes \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{n-i}$, and $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ is the Pauli-X operator. Unitary evolutions of a quantum state under B, C achieve amplitude mixing and coherent, cost-dependent phase rotations, respectively. Canonically, both QAO and QAOA (see Table 5.2 for full names) use Hamiltonians of the form B and C. However, as discussed towards the end of Sec. 5.7.1, other choices can affect the performance on a given instance.

Abbreviation	Name of Algorithm	Reference
QAO	Quantum adiabatic optimization (algorithm)	[26]
SA	Simulated annealing	[170]
QAOA	Quantum approximate optimization algorithm	[28]
BBSA	Bang-bang simulated annealing	§ 5.4.1

Table 5.2: Table of abbreviations for the algorithms studied in this chapter. The last algorithm, BBSA, is introduced in this chapter.

Now, we will describe the candidate algorithms listed in Table 5.2. It will become

evident that these algorithms can all be expressed in the control framework given in Appendix 5.8. This connection is important, as it allows us to borrow existing results from optimal control theory to the setting of heuristic optimization.

5.3 Annealing-based algorithms

5.3.1 Simulated annealing

Simulated Annealing (SA) is a family of classical heuristic optimization algorithms that seek to minimize a potential via the evolution of a classical probability distribution under a simulated cooling process. The dynamics of the distribution are governed by two competing influences:

- Descent with respect to the cost function $c(\mathbf{z})$.
- Thermal fluctuations that kick the walker in a random uphill direction with Boltzmann

  probability, defined according to a controlled temperature parameter τ.

In practice, the above dynamics may be achieved via the following random walk:

1. Initialize the walker at location $\mathbf{r}_1$.
2. Run a p-round annealing schedule, where the i-th round is given by $t_i \in \mathbb{Z}_{\geq 0}$ time steps and a temperature parameter $\tau_i \in \mathbb{R}_{\geq 0}$. For index $i \in [p]$, run:
   (a) For t_i iterations, repeat:

i. Pick direction $\mathbf{e}$ uniformly at random from available local unit displacement vectors.

ii. Let $\delta_{\mathbf{e}} := c(\mathbf{r}_i \oplus \mathbf{e}) - c(\mathbf{r}_i)$ be the cost increase in moving walker from current position $\mathbf{r}_i$ to new position $\mathbf{r}_{i+1} = \mathbf{r}_i \oplus \mathbf{e}$.

iii. If $\delta_{\mathbf{e}} \leq 0$, move to new location with certainty. Otherwise, move with Boltzmann probability $e^{-\delta_{\mathbf{e}}/\tau_i}$, where τ_i is the current temperature in the schedule. In other words,

$$\Pr(\mathbf{r}_i \to \mathbf{r}_i \oplus \mathbf{e}) = \min\left\{1, e^{-\delta_{\mathbf{e}}/\tau_i}\right\}. \tag{5.7}$$

3. Repeat steps 1-2 several times, and report the minimal sampled configuration $\mathbf{z}^*$ and the corresponding cost $c(\mathbf{z}^*)$.

The temperature schedule $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_p)$ must be optimized in order to achieve a final distribution that is well-supported on low-energy states (including the global minima, ideally). In practice, one applies a finite "cooling" schedule in which the elements of $\boldsymbol{\tau}$ descend from ∞ to 0. At each temperature τ_i, the time steps t_i may be seen as relaxation time steps, where the walker distribution equilibrates under thermal exchange with the simulated bath at temperature τ_i. In the limit of infinitely slow, monotonically decreasing temperature schedules that satisfy certain additional conditions arising from deep local minima in the problem instance, simulated annealing always converges to the lowest-cost configuration [172–175]. However, finite-time schedules and a finite relaxation time per temperature step can undo the theoretical guarantee.

The position update of the walkers in the above scheme is implemented via

the Metropolis-Hastings rule, where uphill motions are suppressed with Boltzmann probability. This implies that steeper climbs quickly become exponentially unlikely, resulting in an effective trapping of walkers in basins of depth $\sim \tau_i$. Within these basins, sufficiently high relaxation times allow the walkers to find deep minima. Intuitively, the walkers are allowed to climb barriers "just high enough" so as to settle into progressively deeper minima, as τ_i decreases through the course of the algorithm.

The above process may be seen as a discretization of an approximately equivalent continuous-time Markov process. In the parlance of control introduced in Appendix 5.8, the dynamics of the walker distribution is generated by a stochastic operator $H\left(\tau(t)\right)$ that is singly controlled by the time-dependent temperature parameter $\tau(t)$. For two neighboring positions $\mathbf{z}, \mathbf{z'}$ in the space with a mutual displacement unit vector $\mathbf{e} = \mathbf{z'} - \mathbf{z}$, the corresponding matrix element may be written as

$$H(\tau)_{\langle \mathbf{z}\mathbf{z'} \rangle} = \begin{cases} \mathcal{W}(\tau, \mathbf{z}), & \text{if } \mathbf{z} = \mathbf{z'} \\ 1, & \text{if } \delta_{\mathbf{e}} \leq 0 \\ e^{-\delta_{\mathbf{e}} c/\tau}, & \text{if } \delta_{\mathbf{e}} > 0 \end{cases} \tag{5.8}$$

where the diagonal term term $\mathcal{W}(\tau, \mathbf{z})$ is the negative column sum of the $\mathbf{z}$ column of $H(\tau)$, a condition which ensures stochasticity of the Markov process. Then, the continuous-time dynamics of the probability vector are given by the differential equation

$$\dot{P}(t) = -H(\tau(t))P(t) \tag{5.9}$$

Under a discretization of the above into small time slices Δt_i (such that $||H(\tau_i)||\Delta t_i < 1$), and approximating the temperature schedule as a piecewise constant function, we may rewrite the continuous process as a Markov chain where the dynamics at the i-th slice are given by the stochastic matrix $\mathbb{1} - H(\tau_i)\Delta t_i$. This corresponds to the i-th step of the discrete random walk.

The infinite-temperature and zero-temperature limits of H are important special cases. At $\tau = \infty$, walkers choose random directions and walk with certainty, independent of the potential. This corresponds to the case of *diffusion*. On the other hand, at $\tau = 0$, walkers walk in a randomly chosen direction if and only if the resulting cost is no greater than the current cost. This is what we may call *randomized gradient descent*. We will denote these operators by D, G respectively and give their form below:

$$D_{\langle \mathbf{z}\mathbf{z}' \rangle} := H(0)_{\langle \mathbf{z}\mathbf{z}' \rangle} = \begin{cases} -n(\mathbf{z}), & \text{if } \mathbf{z} = \mathbf{z}' \\ 1, & \text{if } \delta_{\mathbf{e}} \leq 0 \\ 1, & \text{if } \delta_{\mathbf{e}} > 0 \end{cases} \tag{5.10}$$

$$G_{\langle \mathbf{z}\mathbf{z}' \rangle} := H(\infty)_{\langle \mathbf{z}\mathbf{z}' \rangle} = \begin{cases} -n_<(\mathbf{z}), & \text{if } \mathbf{z} = \mathbf{z}' \\ 1, & \text{if } \delta_{\mathbf{e}} \leq 0 \\ 0, & \text{if } \delta_{\mathbf{e}} > 0 \end{cases} \tag{5.11}$$

where $n(\mathbf{z})$ is the number of neighbors, and $n_<(\mathbf{z})$ the number of "downhill" neighbors, of $\mathbf{z}$. (Note: For all bit strings $\mathbf{z}$, $n(\mathbf{z}) = n$ on the usual n-dimensional hypercube.)

5.3.2 SA with linear update

Under Metropolis-Hastings Monte Carlo, we see that the dynamics evolve under $H(\tau)$, which is an operator controlled by the temperature schedule $\boldsymbol{\tau}$. The obvious bang-bang analogue to this is to alternate between periods of zero- and infinite-temperature Metropolis moves, which is the algorithm introduced in Sec. 5.4.1. However, to argue that bang-bang control is optimal using the optimal control framework (as in Sec. 5.5), we must first ensure that the dynamics are linear in the controls. In this section, we present a linearized variant of SA, so that within algorithms of this class, it will be the case that bang-bang control is optimal as a consequence of the Pontryagin Minimum Principle.

Suppose that instead of Metropolis-Hastings probability $\min\left\{1, e^{-\delta_{\mathbf{e}}/\tau}\right\}$, we use a probability $u\Theta(\delta_{\mathbf{e}})$, where $\Theta(\cdot)$ is the Heaviside step function, and $u \in [0,1]$ is a control parameter. That is,

$$\Pr(\mathbf{z} \to \mathbf{z} \oplus \mathbf{e}) = \begin{cases} 1, & \text{if } \delta_{\mathbf{e}} \leq 0 \\ u, & \text{if } \delta_{\mathbf{e}} > 0 \end{cases} \tag{5.12}$$

This rule is qualitatively different from Metropolis-Hastings, since it attaches importance not to the exact energy difference between neighboring states, but only to its sign. Furthermore, the update rule is not guaranteed to satisfy physical prerequisites such as detailed balance that guarantee the convergence of the limiting distribution. However, it is a valid update rule, and we will call SA equipped with these dynamics linear update SA.

Importantly, linear update SA is expressible in the linear control framework. It is possible to write the Markov matrix $H(u)$ corresponding to the continuous version of

Eq. 5.12 as a sum of the diffusion matrix D and the randomized gradient descent operator G,

$$H(u) = uD + (1-u)G \tag{5.13}$$

Finally, we see that, $H(u=0) = D$ and $H(u=0) = G$, thus reproducing the operators appearing in standard SA in the limit of infinite and zero temperature (i.e. $u = 0, 1$), which are the relevant parameter values under bang-bang control.

5.3.3 QAO

The adiabatic algorithm, proposed in 2000 by Farhi, Goldstone and Gutmann [26], is a (quantum) heuristic combinatorial optimization algorithm based upon the adiabatic theorem from quantum mechanics. The adiabatic theorem, loosely stated, says that a system evolving under a time-varying Hamiltonian, when initialized in a ground state, stays in the instantaneous ground state as the Hamiltonian is varied *slowly* in time. The recipe to turn this statement into an algorithm for finding global minima is as follows:

1. Initialize the system in an easily preparable ground state of a Hamiltonian B.

2. Read the problem instance (cost function $c(\mathbf{z})$), and map it to an equivalent Hamiltonian C, as in Eq. 5.5.

3. Implement Schrödinger evolution of the state over the time interval $[0, T]$ under a controlled Hamiltonian $H(s) = u_1(s)B + u_2(s)C$, where $s = t/T$ is the scaled time parameter, and u_1, u_2 are functions of s that describe the *annealing schedule*. The schedule satisfies $u_1(0) = 1 - u_2(0) = 1$, and $u_1(1) = 1 - u_2(1) = 0$.

4. Measure the resulting state in the computational basis.

Under adiabaticity (i.e. when the schedule varies slowly in s), the above algorithm evolves the initial state from the ground state of B to that of C, which is a state that encodes the solution to optimization problem. In particular, the algorithm succeeds if the rate is slower than inverse polynomial in the first spectral gap $\lambda(s)$ (i.e. the energy difference between the ground state and the first excited state) at all times. Typically, this yields a condition on the true runtime T [176, 177]:

$$T \gtrsim O\left(\frac{1}{\lambda^2}\right) \tag{5.14}$$

where $\lambda = \min_s \lambda_s$. Therefore, the guarantee of success of an adiabatic protocol lies in knowing that the minimum gap λ does not scale super-polynomially with n. However, it should be noted that this does not rule out good empirical performance. In fact, by cleverly varying speed as a function of the instantaneous spectral gap, important speedups such as the Grover speedup [178], and the exponential speedup for glued trees, [179] (where, strictly speaking, the protocol is non-adiabatic) can be recovered.

Like Linear Update SA, the QAO Hamiltonian

$$H(u_1, u_2) = u_1 B + u_2 C \tag{5.15}$$

fits into the linear control framework [24]. In fact, we may simplify the above Hamiltonian to a singly-controlled Hamiltonian as follows. In practical applications of QAO, there is a maximum magnitude threshold (say J) for the controls, given by hardware constraints.

We assume that this cutoff does not scale with the input size of the instance. Assume also that the lower cutoff for both u_1 and u_2 is 0. In other words, $u_1, u_2 \in [0, J]$. These design constraints give us a restricted version of QAO where the controls are non-negative and bounded. This restriction is applied simply to state our algorithms within a uniform, linear control framework. Adiabatic algorithms for the instances studied in later sections fit within this framework.

Then, observe that when $u_1 + u_2 > 0$, we can rescale the controls by factor $u_1(s) + u_2(s)$, giving us the following mapping of the time variable and the controls:

$$\frac{\mathrm{d}s}{\mathrm{d}t} \mapsto \frac{\mathrm{d}s}{\mathrm{d}t} \cdot (u_1(s) + u_2(s)) \tag{5.16}$$

$$(u_1(s), u_2(s)) \mapsto \left(u := \frac{u_1(s)}{u_1(s) + u_2(s)}, 1 - u \right) \tag{5.17}$$

Under this mapping, the time parameter is rescaled by a factor of at most $2J$ (corresponding to a constant slowdown), while the parametric Hamiltonian now looks like

$$H(u_1, u_2) \mapsto H(u) = uB + (1 - u)C \tag{5.18}$$

When $u_1(s) = u_2(s) = 0$, which is the only case not covered by the above mapping, we see that the dynamics "switch off" completely. This feature is useful only when the total time T is greater than the time necessary to complete the algorithm. However, if we study protocols as a function of the time horizon T, this feature becomes unnecessary, and we may safely ignore it.

Therefore, we have successfully mapped QAO to a linear, single control framework

with only a constant overhead in the run time. From now on, we assume that QAO possesses the form given in Eq. 5.18.

5.4 Bang-bang algorithms

In parallel with the developments in annealing-based methods, extensive studies have been conducted into the problem of optimal control of quantum dynamics (see [180]), particularly in the context of many-body ground state preparation, e.g [181]. It is often found to be that case that, contrary to a quasistatic schedule, a rapidly switching, bang-bang schedule could be engineered to prepare states quickly.

In combinatorial optimization, an alternative framework based on circuits with variable parameters has been investigated, and has recently gained interest with the introduction of the Quantum Approximate Optimization Algorithm (QAOA), [28]. This is in fact an example of bang-bang control, as observed in [24]. The related problem of ground state preparation has also been approached using Variational Quantum Eigensolver (VQE) ansätze [29] that bear close resemblance to QAOA in their setup. The recent work by Hadfield et al. [165] has proposed a relabeling of the acronym QAOA to the 'Quantum Alternating Operator Ansatz' to capture this generality. In this manner, a new path that explores classical design strategies of quantum algorithms, also known as a hybrid approach, has been paved.

In the coming sections, we will formally introduce QAOA, as well as a new, classical bang-bang version of SA which we call bang-bang SA, or BBSA. Then in Sec. 5.5, we will elaborate on the theoretical motivation behind choosing the bang-bang

approach.

5.4.1 Bang-bang simulated annealing (BBSA)

BBSA is the restriction of linear update simulated annealing (see Sec. 5.3.2) to bang-bang schedules. In other words, this is an algorithm that alternately applies diffusion and randomized gradient descent to the state. An instance of this algorithm may then be specified by the number of rounds p (where in each round we apply the two operators in succession), and the corresponding evolution times for each round.

Observe that Metropolis-Hastings SA, when restricted to $\tau = 0, \infty$, reduces to bang-bang SA.

5.4.2 QAOA

The Quantum Approximate Optimization Algorithm (QAOA) was introduced by Farhi *et al.* in 2014, [28], as an alternative ansatz to the QAO. We note (as is done in [24]) that, like QAO, QAOA is a restriction of the linearly controlled Hamiltonian to the case of bang-bang control, i.e., where we only allow $u = 0, 1$ at any given time. Therefore, QAOA may also be thought of as a parameterized circuit where the parameters are evolution times under H_0 or H_1, or, equivalently, the lengths of the bangs in the control schedule.

Restricted in this way, a QAOA protocol effectively implements a series of alternating Hamiltonian evolutions under the mixing operator B, and the cost operator C. Therefore, for a total of p rounds of alternating evolution with evolution angles

$\boldsymbol{\beta} := (\beta_1, \ldots, \beta_p)$, $\boldsymbol{\gamma} := (\gamma_1, \ldots, \gamma_p)$ for B and C respectively, the final state prepared by QAOA may be expressed as

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = \left[\prod_{i=1}^{p} \mathcal{B}(\beta_i)\mathcal{C}(\gamma_i)\right] |\psi_0\rangle \tag{5.19}$$

where we used the parameterized operators from Eq. 5.5, 5.6, and, as in the case of QAO, the initial state $|\psi_0\rangle$ is an easily preparable state such as the equal superposition of bitstrings, $|+^{\otimes n}\rangle$.

QAOA with a fixed number of rounds p, also written as QAOAp, is a scheme for preparing one of a family of trial states of the form $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$. With the angles as search parameters, a figure of merit such as the energy expectation of the cost operator $E(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \langle \boldsymbol{\beta}, \boldsymbol{\gamma}| C |\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ is approximately minimized with the aid of classical outer loop optimization.

5.5 Conditions for optimality of bang-bang control

Now, we will elaborate on the theoretical motivation for choosing a bang-bang approach to optimization algorithms, expanding on the observations made in [24, 25]. The Pontryagin Minimum Principle (PMP) from optimal control theory [23] provides key insight into the nature of optimal schedules for heuristic optimization algorithms expressible in the control framework. As discussed in Appendix 5.8, PMP gives necessary conditions on the control in the form of a minimization of the control Hamiltonian, which is a classical functional of the state amplitudes and corresponding conjugate "momenta", and depends on the control parameters as well.

When the control Hamiltonian $\mathcal{H}$ is linear in the control vector $\mathbf{u}$, the minimization condition Eq. 5.54 implies that the optimal control is *extremal*, in the sense that the control only takes values on the boundary of the feasible control set at any given time. When the control parameters are individually constrained to lie in a certain interval, $u_i \in [a_i, b_i]$, then we say that the optimal protocol is bang-bang, i.e. $u_i(t) = a_i$ or b_i. Thus, the individual controls switch between their extremal values through the course of the protocol. While the heuristic algorithms QAO, QAOA and SA with linear update satisfy the condition of linear control, one should exercise caution when stating the optimality of bang-bang control within these frameworks. We note a few important caveats here:

1. PMP simply gives a necessary condition for optimality, it does not provide the optimal protocol. A different control theory tool, the Hamilton-Jacobi-Bellman equation, does provide a way to find the optimal protocol via dynamic programming.

2. There may be an arbitrary number of switches in the optimal bang-bang protocol. In fact, some problems exhibit the so-called *Fuller phenomenon*, in which the optimal control sequence has an infinite number of bangs, and is therefore rendered infeasible.

3. The control Hamiltonian may become *singular* at any point during the protocol. A singular interval is one in which the first derivative of $\mathcal{H}$ with respect to u vanishes. In these intervals, the optimal control is not necessarily bang-bang. The presence of generic singular intervals has already been observed before in the dynamics of spin systems (see, e.g. [182, 183]). Therefore, in order to guarantee that the optimal

control is bang-bang at all times, one must first show that there are no singular intervals during the protocol.

4. The original PMP is stated and proved for dynamics over Euclidean vector spaces over $\mathbb{R}$. However, in quantum optimization the amplitudes take values in $\mathbb{C}$, and the Hilbert space is a complex projective vector space with a non-Euclidean geometry. The generalization must be made with caution.

Despite these caveats, PMP does provide theoretical motivation for using bang-bang control as a design principle for heuristic optimization algorithms. In the following sections, we exhibit examples where bang-bang control exponentially outperforms conventional SA and QAO.

5.6 The problem instances

Now, we describe the problem instances that will be used as benchmarks for our algorithms. The two following instances have appeared in the context of comparisons between quantum and classical heuristic optimization algorithms, usually to show the inability of the classical algorithm to escape a local minimum and find the true, global minimum [170, 171, 184]. This is often interpreted as evidence of a quantum advantage, such as the ability to tunnel through barriers. In keeping with this tradition, we will select these as our benchmarking instances, and look for general features in the performance of our candidate algorithms.

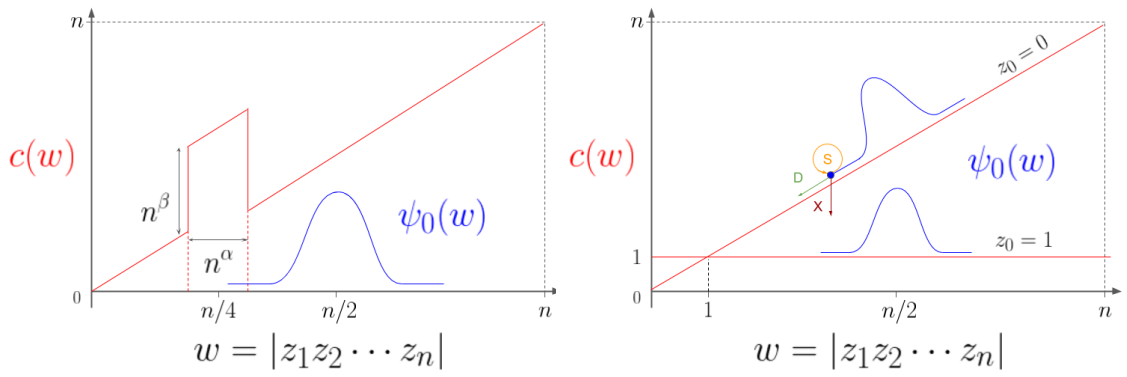


Figure 5.1: Schematic energy landscapes of the two instances, `Spike` (left) and `Bush` (right). In each diagram, the blue curve indicates the distribution of the initial state, the equal superposition over all bit strings.

5.6.1 Bush of implications

The bush of implications or `Bush` is an instance first crafted in [170] in order to demonstrate the failure of SA where QAO succeeds, with an exponential separation between the two. In `Bush`, the potential is not fully symmetric under permutation of bits. Instead, the first bit (the "central" bit, indexed by 0) determines the potential acting on the Hamming weight of the remaining n "peripheral" bits. Specifically,

$$c(\mathbf{z} = z_0 z_1 \dots z_n) = z_0 + \sum_{i=1}^{n} z_i \, (1 - z_0) \; = z_0 + w \, (1 - z_0) \tag{5.20}$$

where $w = |z_1 \dots z_n|$. So, the potential is constant and equal to 1 when $z_0 = 1$, and a Hamming ramp, $r(w) = w$ when $z_0 = 0$, as shown in Fig. 5.1. Note that we adopted a bit-flipped definition of c as compared to the original in [170]. The reason is simply notational convenience. The energy landscape of the bush of implications can be viewed as the number of clauses violated in a constraint satisfaction problem, where each clause

takes the form $\neg z_0 \implies \neg z_j$ for $j > 0$, which lends the instance its name.

5.6.2 Hamming ramp with spike

Next, we present a second family of Hamming-symmetric potentials studied first in [170, 185], the Hamming ramp with a spike. In the general form more recently studied in [171, 184, 186], this potential is given by a ramp $r(w) = w$, plus a rectangular "spike" function $s(w)$ centered at $w = n/4$ with width $O(n^a)$ and height $O(n^b)$, for two exponents $a, b \in [0, 1]$.

$$\text{Ramp: } r(w) = w, \text{ Spike: } s(w) = \begin{cases} n^b, \text{ if } w \in [\frac{n}{4} - \frac{n^a}{2}, \frac{n}{4} + \frac{n^a}{2}] \\ 0, \text{ otherwise.} \end{cases} \tag{5.21}$$

$$\text{Full Potential: } c(w) = r(w) + s(w) \tag{5.22}$$

We will use this form for the `Spike` family of instances.

5.7 Performance

Now, we will state the performance of the algorithms from Sec. 5.3, 5.4 on the instances defined in Sec. 5.6.1, deriving or using existing results as appropriate. We will find that in both the classical (SA vs. BBSA) and the quantum (QAO vs. QAOA) settings, there exist parameter regimes in which the bang-bang algorithms are exponentially faster than their quasistatic analogues.

5.7.1 SA and QAO

For both the `Bush` and `Spike` examples, Farhi et al. argue in [170] that simulated annealing gets stuck in local minima, and is exponentially unlikely to reach the global minimum in polynomial time, in the input size $n \to \infty$. Additionally, they argue for the success of QAO on these instances in certain parameter regimes.

For the `Spike` example, [185] and [186] show that when the width and height parameters satisfy $a + b \leq 1/2$, quantum annealing solves `Spike` efficiently. If, on the other hand, $2a + b > 1$, it was shown by [171] that the minimum spectral gap has an exponential scaling in n, implying the failure of quantum annealing in this problem regime. For the `Bush` example, it was shown in [170] that the gap scaling is polynomial in n, thus allowing for an efficient adiabatic algorithm to solve this instance. We note that the performance depends on the choice of the initial mixing Hamiltonian B. In particular, out of the following family of mixers

$$B_\lambda = -\lambda(n+1)X_0 - \sum_{i=1}^{n} X_i, \tag{5.23}$$

QAO is successful when $\lambda \geq 1$. On the other hand, when $\lambda = 1/(n+1) < 1$, we recover the canonical mixing operator B from Eq. 5.6, and QAO is expected to take exponential time to solve `Bush`.

Despite the caveats, `Bush` and `Spike` are examples of instances where we have an exponential separation between a quantum (QAO) and classical (SA) algorithm. However, in the next section we show that a different, purely classical, bang-bang strategy

matches the performance of QAO on the `Bush` and `Spike` instances by solving them in polynomial time.

5.7.2 Bang-bang simulated annealing

Now, we will show that the bang-bang version of simulated annealing is able to find the ground state of both `Bush` and `Spike` in time polynomial in n, and therefore exponentially outperforms SA (and QAO for certain parameter regimes, see Table 5.1), on both instances.

5.7.2.1 `Bush`

We will now show that BBSA efficiently finds the minimum of `Bush` via BBSA. In fact, the protocol simply involves performing randomized gradient descent (G) without any switches to diffusion. First, we characterize the G matrix for this instance. The natural basis for this problem is a conditional Hamming basis $\{|z_0, w) : z_0 \in [1], w \in [n]\}$ parameterized by the value of the central bit z_0, and the weight of the peripheral string $w = |z_1 \cdots z_n|$. The allowed transitions under G are as given below:

$$|0, w) \to |1, w), \text{ for all } w > 0. \tag{5.24}$$

$$|z_0, w) \to |z_0, w-1), \text{ for all } z_0 \in [1], w > 0. \tag{5.25}$$

$$|1, 0) \to |0, 0). \tag{5.26}$$

In particular, this implies that a walker at the global minimum $|0, 0)$ cannot leave under G. Consider a discrete, Markov chain Monte Carlo implementation of G, in which we break

up the Markov evolution into $N = 1/\delta t$ steps of size δt. The stepsize δt is an empirical parameter which will be set later, while at the moment we only assume that $\delta t \ll 1$. Then, we may write the Markov evolution as

$$|P_N) = \left[\prod_{i=1}^{N} e^{-G\delta t}\right] |P_0) \simeq \left[\prod_{i=1}^{N} (\mathbb{1} - G\delta t)\right] |P_0) \tag{5.27}$$

Each step $\mathbb{1} - G\delta t$ above is a stochastic evolution if δt is sufficiently small, i.e., if all entries of the matrix represent valid probabilities. The requirement that the column sum be 1 is automatically satisfied since G is column-sum-zero. Then, we start with a walker sampled from the initial state $|P_0)$, and, for every step 1 to N, we update the walker's position based on the transition probabilities given by $\mathbb{1} - G\delta t$. This is given in more detail below. We will show that the above procedure transports a fraction of at least $n^{-2.503}$ of walkers to the global minimum, in number of steps $N = O\left(\frac{1}{\delta t} \log n\right)$. Finally, arguing that it suffices to choose $\delta t = \Theta(n^{-1})$ gives a polynomial runtime of $\Theta(n^{3.503} \log n)$ to have a constant success probability.

In our analysis, we only keep track of the walker in the $z_0 = 0$ subspace, which contains the global minimum. Any walker that starts in or enters the $z_0 = 1$ subspace during the algorithm will be presumed dead, and we terminate its walk. This simplification is allowed, since it may only worsen the success probability obtained through this analysis. Initially, exactly half of the walkers are alive, i.e. in the subspace $z_0 = 0$, and concentrated in a band of width $\sim \sqrt{n}$ around $w = n/2$. For a walker at Hamming weight $w > 0$, there are three possible moves (illustrated in Fig. 5.1):

1. (D) Descend to weight $w - 1$, with probability $w\delta t$.

2. (S) Stay at the same location with probability $1 - (w + 1)\delta t$.

3. (X) "Die", i.e., escape to the $z_0 = 1$ subspace, with probability δt.

When $w = 0$, the D and X moves are forbidden, and the walker can only stay in place. Additionally, we denote the event of survival (i.e. D or S) by $\bar{X}$. Now, we track the random walk under the stated moves. Let $\hat{m}$ be a random variable representing the total number of moves the walker takes to reach the global minimum, $|0, 0)$. If the walker dies, we say that $\hat{m} = \infty$. Otherwise, $\hat{m}$ is finite and equal to the sum of number of moves spent at each weight $w = 1, 2, \ldots, n$. Defining a corresponding random variable $\hat{m}_w$ for the number of moves spent at each weight, we may write

$$\hat{m} = \sum_{w=1}^{n} \hat{m}_w \tag{5.28}$$

The expected value of $\hat{m}$ tells us how many moves any given walker needs to reach the global minimum under G. However, since we are only interested in living walkers, we will condition the expectation on the walker staying alive ($\bar{X}$). Then,

$$\mathbb{E}\left(\hat{m} \mid \bar{X}\right) = \sum_{w=1}^{n} \mathbb{E}\left(\hat{m}_w \mid \bar{X}\right) \tag{5.29}$$

At each weight w, the condition of survival limits the allowed moves to the regular expression S^*D. In other words, the walker stays in place for some number of moves before descending. Note that the probability of not dying in m moves is $(1 - \delta t)^m$. Therefore, the probability of spending m total moves, conditioned on survival, is given

by

$$\Pr\left(\hat{m}_w = m \mid \bar{X}\right) = \frac{\Pr\left(S^{m-1}D\right)}{\Pr\left(\bar{X}^m\right)} = \frac{\left(1-(w+1)\,\delta t\right)^{m-1} w\delta t}{(1-\delta t)^m} \tag{5.30}$$

$$= \left(\frac{1-(w+1)\,\delta t}{1-\delta t}\right)^{m-1} \cdot \frac{w\delta t}{1-\delta t} \tag{5.31}$$

$$\lesssim e^{-w\delta t(m-1)} \frac{w\delta t}{1-\delta t} \tag{5.32}$$

where the last inequality follows from a Taylor series comparison of $(1-(w+1)\delta t)/1-\delta t$ under the assumption that $\delta t < 1$. So, the expectation value of $\hat{m}_w$ is

$$\mathbb{E}\left(\hat{m}_w \mid \bar{X}\right) = \sum_{m=1}^{\infty} m \cdot \Pr\left(m \mid \bar{X}\right) \lesssim \frac{w\delta t \cdot e^{w\delta t}}{1-\delta t} \sum_{m=1}^{\infty} m \cdot e^{-mw\delta t} \tag{5.33}$$

$$= \frac{w\delta t}{(1-\delta t)\left(1-e^{-w\delta t}\right)^2} \tag{5.34}$$

Finally, the full expectation value is given by

$$\mathbb{E}\left(\hat{m} \mid \bar{X}\right) \lesssim \frac{1}{1-\delta t} \sum_{w=1}^{n} \frac{w\delta t}{\left(1-e^{-w\delta t}\right)^2} \tag{5.35}$$

Next, using the variable substitution $x = w\delta t, dx = \delta t$, we may turn the above sum into an approximate integral. In fact, the integrand $x/(1-e^{-x})^2$ is monotonically decreasing,

so the sum is upper bounded by

$$\mathbb{E}\left(\hat{m} \mid \bar{X}\right) \lesssim \frac{\delta t}{(1-\delta t)\left(1-e^{-\delta t}\right)^2} + \frac{1}{\delta t(1-\delta t)} \int_{\delta t}^{n\delta t} \frac{x}{\left(1-e^{-x}\right)^2} dx \tag{5.36}$$

$$\lesssim \frac{4}{(1-\delta t)\,\delta t} + \frac{1}{\delta t(1-\delta t)} \int_{\delta t}^{n\delta t} \frac{4}{x} dx = \frac{4}{(1-\delta t)\,\delta t} + \frac{4}{\delta t(1-\delta t)} \log(n) \tag{5.37}$$

where we used the trick that since $x/2$ and $1-e^{-x}$ are both monotonically increasing, and $x/2 < 1-e^{-x}$ for $x = 0, 1$, then it follows that $x/2 < 1-e^{-x}$ for all $x \in [0,1]$. In fact, a tighter bound may be obtained by replacing 2 by $e/(e-1) \approx 1.58$, which yields a scaling of $\mathbb{E}\left(\hat{m} \mid \bar{X}\right) \lesssim \frac{2.503}{\delta t} \log n$. Finally, the expected survival probability is $\Pr\left(\bar{X}\right) \gtrsim e^{-\delta t \cdot \frac{2.503}{\delta t} \log n} = \frac{1}{n^{2.503}}$, which is polynomial in n. Therefore, applying this algorithm for $\frac{1}{\delta t} \log n$ with $\delta t = \Theta(n^{-1})$, yields a polynomial probability of success. Repeating for at most $n^{2.503}$ trials amplifies the success probability to a constant. So, the total time complexity is $On^{3.503} \log n$, which is efficient in the input size n.

In Fig. 5.2 below, numerics of the continuous-time process (see Eq. 5.27) confirm that the total time indeed scales as $\log n$.

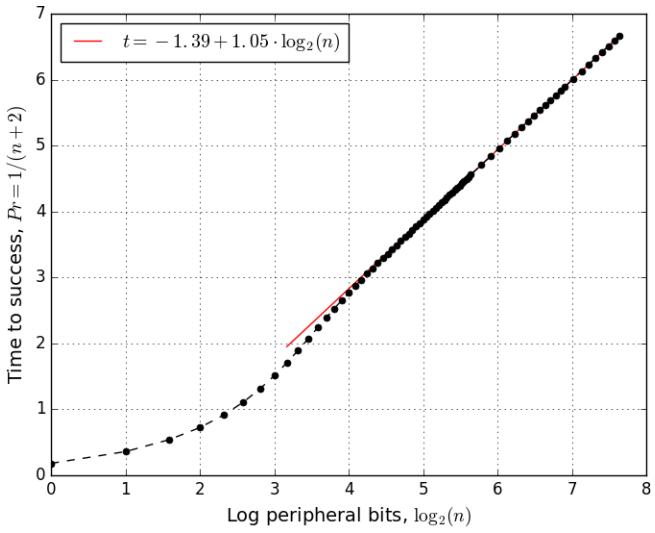


Figure 5.2: Plot of the input size n vs. total time for success (determined by the time taken for a polynomial fraction of walkers to reach the global minimum). Note that the continuous-time process does not contain the polynomial factors; those arise from discretization into small timesteps δt of order $\lesssim 1/n$.

5.7.2.2 `Spike`

In the previous section, we showed that `Bush` is a problem instance where classical bang-bang algorithm (BBSA) can outperform a classical quasistatic algorithm (SA) exponentially. While this suffices to show the polynomial inequivalence of SA and BBSA, it is nonetheless interesting to explore further examples where this is the case. The `Spike` problem, as presented in 5.21, is the second instance where BBSA can exponentially outperform SA and QAO. Since the separation is sensitive to details such as the shape of the spike, we refer the reader to Appendix 5.9 for further discussion.

5.7.3 QAOA

Lastly, we will show that one round of QAOA (or QAOA1) efficiently finds the minimum of the instances `Bush` and `Spike`. In fact, as discussed later in this section, QAOA1 solves a more general class of symmetric instances that includes the `Spike` (and with some more analysis, the `Bush`) example. This is one of the main results of the chapter, given in Theorem 5.7.1.

5.7.3.1 `Spike`

One of the key features of this instance is that the spike has exponentially small overlap with the initial state $|+\rangle^{\otimes n}$. Intuitively, this implies that the state does not "see" the spike, and should therefore behave as if evolving under a pure Hamming ramp. We state this as the following lemma:

Lemma 5.7.1. Let c(w) be a Hamming-symmetric cost function on bitstrings of size n, and let $p(n) \in [0, 1]$ be a problem size-dependent probability. Suppose $c(w) = r(w) + s(w)$, where r, s are two functions satisfying the following:

1. $\min_w c(w) = \min_w r(w)$.
2. There exist angles β, γ such that QAOA1 with schedule (β, γ) minimizes $r(w)$ with probability at least $p(n)$.
3. If the initial state is $|\psi_0\rangle = \sum_w A_w |w\rangle$, then $s(w)$ overlaps weakly with $|\psi_0\rangle$ in the sense that

$$\sum_{w=1}^{n} 4|A_w|^2 \sin^2\left(\frac{\gamma s(w)}{2}\right) \leq o(p(n))$$

Then, QAOA1 with schedule (β, γ) minimizes $c(w)$ with probability at least $p(n) - o(p(n))$.

For the `Spike` instance, we decompose the cost into a ramp term and a spike, $c(w) = r(w) + s(w)$. First, we compute the success probability of QAOA1 on only the ramp term $r(w)$. This potential may be written as

$$R = \sum_{w=0}^{n} w \left|w\right\rangle \left\langle w\right| = \sum_{i=1}^{n} \frac{\mathbb{1} - Z_i}{2} = \frac{n}{2}\mathbb{1} - \frac{1}{2}\sum_{i=1}^{n} Z_i \tag{5.38}$$

which is a 1-local operator on qubits, just like B. It can be seen that the protocol simply applies a rotation from the $|+\rangle$ state to the $|0\rangle$ state on each qubit via a $Z/2$ rotation followed by an X rotation, and succeeds with probability 1. The angles can be read off from the Bloch sphere: $\gamma = 2 \cdot \pi/4 = \pi/2$, and $\beta = \pi/4$.

Then, it follows from Lemma 5.7.1 that the effect of the spike $s(w)$ under QAOA1 is negligible if $\sum_w 4\sin^2(\gamma s(w)/2)|A_w|^2$ is small, where A_w are amplitudes of the initial state in the symmetric basis. But this sum may be bounded as

$$\sum_{w=0}^{n} 4\sin^2(\gamma s(w)/2)|A_w|^2 = \frac{1}{2^{n-2}} \sum_{w=n/4-n^a/2}^{n/4+n^a/2} \sin^2(\gamma n^b/2)\binom{n}{w} \tag{5.39}$$

$$\leq \frac{1}{2^{n-2}} \sum_{w=n/4-n^a/2}^{n/4+n^a/2} \binom{n}{w} = 4 \sum_{w=n/4-n^a/2}^{n/4+n^a/2} B(w; n, 1/2) \tag{5.40}$$

where $B(w; n, 1/2)$ is a binomial term corresponding to the probability of n tosses of a fair coin returning exactly w heads. Now, we may use known bounds on tail distributions

such as Hoeffding's inequality, and we finally have

$$\sum_{w=0}^{n} 4\sin^2(\gamma s(w)/2)|A_w|^2 = 4\sum_{w=n/4-n^a/2}^{n/4+n^a/2} B(w;n,1/2) = o(1) \text{ when } a<1 \tag{5.41}$$

Then, applying Lemma 5.7.1, we conclude that, for a spike with $a \in [0,1)$ and arbitrary b, QAOA1 with angles $(\pi/4, \pi/2)$ finds the global minimum with probability polynomially close to 1.

This QAOA1 protocol is asymptotically successful for any (a,b) chosen from the set $[0,1) \times \mathbb{R}$. In practice, finite n instances will show effects of the finite overlap of the initial state with the spike at a close to 1. But even in this regime, the barrier height is essentially irrelevant, since it appears in the argument of a sinusoid and may only affect the bounds in Eq. 5.39 by a constant.

5.7.3.2 `Bush`

The `Bush` instance is a quasi-symmetric potential, since it depends on the value of the central bit. In the $z_0 = 1$ sector, the potential is a constant, while in the $z_0 = 0$ sector, it is a ramp. So, in analogy with Eq. 5.38

$$C = |1\rangle\langle 1| \otimes \mathbb{1} + |0\rangle\langle 0| \otimes \left(\frac{n}{2}\mathbb{1} - \frac{1}{2}\sum_{i=1}^{n} Z_i\right) \tag{5.42}$$

For ease of analysis, separate the mixing operator into the mutually commuting peripheral terms and central term:

$$\mathcal{B}(\beta) = e^{-i\beta B} = (\cos\beta\mathbb{1}_0 - i\sin\beta X_0)\prod_{i=1}^{n}(\cos\beta\mathbb{1}_i - i\sin\beta X_i) \equiv \mathcal{B}_0\mathcal{B}_i$$

As before, the QAOA protocol implements one Z rotation (operator $\mathcal{C}(\gamma) = e^{-i\gamma C}$) followed by an X rotation (operator $\mathcal{B}(\beta)$). Since the `Bush` potential contains a ramp in the relevant sector, we will try the protocol used for the `Spike` instance, $\beta = \pi/4, \gamma = \pi/2$.

The Z-rotation transforms the initial state (on the peripheral bits) into the $+Y$ eigenstate, $|+\rangle^{\otimes n} \to \frac{1}{\sqrt{2^n}}(|0\rangle + i|1\rangle)^{\otimes n}$. So, the full state transforms as

$$\frac{1}{\sqrt{2}}|1\rangle\otimes|+\rangle^{\otimes n} + \frac{1}{\sqrt{2}}|0\rangle\otimes|+\rangle^{\otimes n} \xrightarrow[\mathcal{C}(\pi/2)]{} \frac{-i}{\sqrt{2}}|1\rangle\otimes|+\rangle^{\otimes n} + |0\rangle\otimes\frac{1}{\sqrt{2^{n+1}}}(|0\rangle + i|1\rangle)^{\otimes n}$$

Next, $\mathcal{B}_i$ transforms the state to

$$\frac{-i}{\sqrt{2}}|1\rangle\otimes|+\rangle^{\otimes n} + |0\rangle\otimes\frac{1}{\sqrt{2^{n+1}}}(|0\rangle + i|1\rangle)^{\otimes n} \xrightarrow[\mathcal{B}_i(\pi/4)]{} \frac{-ie^{-in\pi/4}}{\sqrt{2}}|1\rangle\otimes|+\rangle^{\otimes n} + \frac{1}{\sqrt{2}}|0\rangle\otimes|0\rangle^{\otimes n}$$

and finally, the central mixing term $\mathcal{B}_0$ gives (with $\omega := e^{-in\pi/4}$)

$$\frac{-i\omega}{\sqrt{2}}|1\rangle\otimes|+\rangle^{\otimes n} + \frac{1}{\sqrt{2}}|0\rangle\otimes|0\rangle^{\otimes n} \xrightarrow[\mathcal{B}_0(\pi/4)]{} \frac{-i}{2}|1\rangle\otimes\big(\omega|+\rangle^{\otimes n} - |0\rangle^{\otimes n}\big) + \frac{1}{2}|0\rangle\otimes\big(\omega|+\rangle^{\otimes n} + |0\rangle^{\otimes n}\big)$$

which is the final state $|\psi_f\rangle$. The success probability is then

$$\text{Pr(success)} = |\langle \mathbf{0} \rangle \psi_f|^2 = \frac{1}{4} |1 - \omega \langle 0 \rangle +^n|^2 = 1/4 + O(1/2^n) \tag{5.43}$$

which is a finite constant and may be boosted polynomially close to 1 with a logarithmic number of repetitions.

5.7.3.3 Other symmetric instances

The success of QAOA1 on the two chosen instances is in part due to the fact that only the potential on the support of the initial state affects the state dynamics. This feature is absent from the other algorithms studied here. Notably, for the adiabatic algorithm on `Spike`, while it is true that the spectral gap is minimized at the same point u^* as for the ramp without the spike (see [184]), the size of the gap itself depends on the spike parameters, so that in particular, when the spike is sufficiently broad or tall, the gap becomes exponentially small in n. In stark contrast, the performance of QAOA1 is *independent* of the gap parameters, since the state has vanishing support on the spike.

Now, we will use this feature to give conditions under which a symmetric cost function may be successfully minimized by QAOA1. When the cost can be decomposed into a linear ramp and a super-linear part that has small support on the initial state, one may ignore the super-linear terms and treat the problem as a linear ramp. Suppose we have a Hamming-symmetric cost function $c(\tilde{w}) = c_0 + c_1\tilde{w} + c_2\tilde{w}^2 + \cdots$, written as a Taylor series in $\tilde{w}$, the shifted Hamming weight $\tilde{w} = w - n/2$ (which we henceforth replace with w). Separate the function into a linear part and a super-linear part, $c(w) = r(w) + q(w)$,

where

$$r(w) = c_0 + c_1 w \tag{5.44}$$

$$s(w) = c_2 w^2 + \cdots \tag{5.45}$$

Under Lemma 5.7.1, if it is the case that $s(w)$ overlaps weakly with the initial state (which is roughly supported on weights $n/2 \pm O\left(\sqrt{n}\right)$), and if the addition of $s(w)$ does not change the global minimum of $r(w)$, then such a cost function $c(w)$ can be optimized using a "ramp protocol" for $r(w)$, as was done for the `Spike` problem in Sec. 5.7.3.1 (provided the slope of the ramp $|c_1| \geq O(1/\text{poly}(n))$).

However, in this case we can do better (Theorem 5.7.1 below): even if the global minimum of $c(w)$ does not coincide with that of $r(w)$, the ramp protocol may be suitably modified to ensure the successful minimization of $c(w)$. Suppose $\min_w c(w) = w^*$. For the ramp $r(w) = c_0 + c_1 w$, the first step of QAOA is evolution under $\mathcal{C}(\pi/(2|c_1|))$. For $c(w)$, we modify γ to γ^* (to be determined), and keep $\beta = \pi/4$ unchanged. Then, the final state may be written as

$$|\psi_f\rangle = \bigotimes_{i=1}^{n} \left(\sin\left(\gamma^*/2\right)|0\rangle + \cos\left(\gamma^*/2\right)|1\rangle\right) \tag{5.46}$$

$$= \sum_{w=0}^{n} \left(\sin\left(\gamma^*/2\right)\right)^{n-w} \left(\cos\left(\gamma^*/2\right)\right)^{w} \binom{n}{w}^{1/2} |w\rangle \tag{5.47}$$

Then, by inspection, γ^* must maximize the success probability, or equivalently, the

function $(\sin(\gamma^*/2))^{2(n-w^*)}(\cos(\gamma^*/2))^{2w^*}$. An elementary calculation yields that

$$\gamma^* = \arccos\sqrt{\frac{w^*}{n}} \tag{5.48}$$

Finally, the success probability is

$$\text{Pr(success)} = \frac{(w^*)^{w^*}(n-w^*)^{w^*}}{n^n}\binom{n}{w^*} = O(1) \tag{5.49}$$

by Stirling's approximation. So, QAOA1 with $\beta = \pi/4, \gamma = \gamma^*$ successfully optimizes the cost function $c(w)$. Finally, we note that if the minimum w^* of c is unknown, the above QAOA1 protocol may be carried out for all $n+1$ possible values of w^* until success, which is at most a factor $O(n)$ overhead. Therefore, we have just proven the following result:

Theorem 5.7.1. When $c(w) = r(w) + s(w)$ and r is linear in w with slope $\Omega(1/\text{poly}(n))$, and $s(w)$ satisfies the weak overlap condition 3 in Lemma 5.7.1, $c(w)$ can be successfully minimized via QAOA1 with at most a polynomial number of classical repetitions.

There is an intuitive picture for the feature of QAOA discovered in Theorem 5.7.1 above. As has been observed before [171, 187], the low energy spectrum of the mixing operator B can be mapped to a suitable harmonic oscillator that treats the Hamming weight w as the position variable. Under this mapping, the initial state $|+\rangle^{\otimes n}$ acts as the vacuum state wavepacket, and a linear ramp with slope a, $C = a\sum_w w\,|w\rangle\langle w|$ is the analogous position operator. We may then qualitatively work out the action of QAOA on the initial wavepacket. The first round, evolution under C, displaces the vacuum to

a state with finite momentum $p = a\gamma$. Then, evolution under the harmonic oscillator Hamiltonian B for time $\beta = \pi/2$ rotates the coherent state so that the final state is one that is displaced in w. So, in a single round of QAOA, the wavepacket gains momentum and propagates to a new location in hamming weight space. (This feature has been recently noted in [169].) While the above method recovers the QAOA1 protocol qualitatively, it gets the angle γ wrong by a factor $2/\pi$. This is due to the curvature of the phase space. In fact, the wavepacket is more accurately described by a *spin-coherent* state, in which the conjugate operators are the total spin operators S_x and S_z. It remains to be seen how this (spin-)coherent state picture may be employed to understand the behavior of QAOA on other (especially non-Hamming symmetric) instances. The simplicity of this description suggests a classical algorithm which simulates the momentum transfer and jump operations of the wavepacket via local gradient measurements of the cost function. This could give rise to a new, quantum-inspired classical search heuristic that escapes local minima more efficiently than existing classical methods.

5.8 The control framework

Given a dynamical equation depending on additional parameters (which we call the *controls*), what properties does a control protocol which optimizes a given cost function satisfy? The relevance of this question extends across many fields where optimal control (with respect to a cost function) is desired. In fact, it has been observed [24, 25] that the optimal control problem also applies to heuristic optimization algorithms, where the controlled dynamics are described precisely by Schrödinger evolution under the annealing

Hamiltonian, and the cost function is given by the energy of the final state.

Consider a first-order differential equation describing the dynamics of an n-dimensional real vector $x \in \mathbb{R}^n$, and controlled by m control parameters which we denote by the vector $u \in \mathbb{R}^m$:

$$\dot{x}(t) = f(x(t), u(t)) \tag{5.50}$$

The functional form f may be very general; we only assume that f is "Markovian" (i.e., depends only on the current state $(x(t), u(t))$), and that there is no explicit time-dependence. Typically, it is further assumed that the control u inhabits a fixed, *compact* subset, $u \in \mathcal{U} \subset \mathbb{R}^m$. The domain $\mathcal{U}$ represents a feasible set of controls.

In order to talk about optimal control, we must first specify a notion of cost. In a real problem such as optimizing the trajectory of a spacecraft, the cost might be expressed in terms of time, amount of fuel used (i.e. a trajectory-dependent cost), and the distance of the final position from the target location (i.e., a final state cost). Thus, the cost function may generally be expressed as a (weighted) sum of three costs:

1. the total time for the process, $T = \int_0^T 1 \mathrm{d}t$
2. the *running* cost, which is given as an integral over the running time, $\int_0^T L\left(x(t), u(t), t\right) \mathrm{d}t$
3. the *terminal* cost, which is a final state-dependent function $K(x(T))$.

The full cost function may be expressed in the general form

$$J = K(x_{final}) + \int_0^\infty L\left(x(t), u(t), t\right) \mathrm{d}t \tag{5.51}$$

where J is a functional of the control schedule $u(t)$ and the dynamical path $x(t)$. The objective is to find the control function $u(t)$, over all piecewise continuous functions $u : \mathbb{R}_{\geq 0} \to \mathcal{U}$, that minimize the overall cost, i.e. $\arg\min_{u(t)} J(u)$. This is the so-called infinite time horizon formulation of the problem. Alternatively, one can fix the total time for the protocol T to be finite. Then, we are asked to minimize over all piecewise continuous functions $u : [0, T] \to \mathcal{U}$ the cost

$$J = K(x(T)) + \int_0^T L\left(x(t), u(t), t\right) \mathrm{d}t \tag{5.52}$$

A wealth of literature in classical control theory discusses the question of optimal control, and we emphasize its potential applicability in the setting of designing efficient heuristic optimizers, both classical and quantum. Here, we will focus on one result, the *Pontryagin Minimum Principle* (PMP), which imposes necessary conditions for a control protocol to be optimal using the so-called control Hamiltonian description.

The control Hamiltonian $\mathcal{H}$ is a classical functional describing auxiliary Hamiltonian dynamics on a set of variables given by x and corresponding *co-state* (or conjugate momentum) variables p. The conjugate momenta depend on the cost function J in Eq. 5.51, and are introduced as Lagrange multipliers that impose the equations of motion for each coordinate of x. The full cost function (at time t), which includes the cost terms in J and the constraints, is given by the control Hamiltonian $\mathcal{H}$.

$$\mathcal{H} := L(x, u) - p \cdot f(x, u) \tag{5.53}$$

Then, PMP states that the optimal control is one which minimizes the control Hamiltonian at all times. That is,

$$\mathcal{H}\left(x(t), p(t), u^*\right) \leq \min_{u \in \mathcal{U}} \mathcal{H}\left(x(t), p(t), u\right) \tag{5.54}$$

In the special case when $\mathcal{H}$ is linear in the control u, the above minimality condition is satisfied only if the control lies on the boundary of the feasible set $\mathcal{U}$. This implies that optimal trajectories are bang-bang, i.e., the controls only take their extremal values. The optimal point(s) on the boundary are determined by the intersection of the constant-$\mathcal{H}$ hyperplanes in control space with the set boundary. However, an important exception arises when the derivative of $\mathcal{H}$ with respect to u vanishes over a finite interval. In this case, the control becomes singular, i.e., its optimal value no long lies solely on the boundary of $\mathcal{U}$.

The control framework described here covers many heuristic optimization algorithms, and we will fix some notation to suit this setting. The dynamical vector of interest will a state $|\psi\rangle$ (quantum) or $|\psi)$ (classical), and the generator of dynamics will be a *controlled* linear operator

$$H(\mathbf{u}) = \sum_{i=0}^{m} u_i H_i \equiv \mathbf{u} \cdot \mathbf{H} \tag{5.55}$$

where $\mathbf{u}$ and $\mathbf{H}$ are vectors with components u_i and H_i respectively. We assume that individual Hamiltonians H_i are time-independent, and only their overall strength, controlled by the coefficient $u_i(t)$, is time-dependent. We will fix the range of all u_i to $[0, 1]$.

5.9 Bang-bang simulated annealing on the `Spike`

For `Spike`, the strategy used for `Bush`, namely, run randomized gradient descent (zero-temperature SA) from start to finish, fails due to the presence of a barrier. So, if we run gradient descent for time $O(n)$ per walker, then we are left with a distribution sharply peaked at the false minimum. We may now attempt to diffuse across the barrier. For a sufficiently wide barrier, this strategy will again fail, since the diffusion rate across the spike is exponentially small in n^a. However, we instead turn on diffusion for a short time, so that a constant fraction of the walkers "hop on" the barrier, while the rest diffuse away from the barrier. Then, we turn on randomized gradient descent again until the finish. The fraction of walkers on the barrier are now guaranteed to walk to the global minimum in time $O(n)$, as the slope is positive.

So, it can be seen that for the spike problem, an algorithm with the same structure as SA but a schedule that is designed without the adiabaticity constraint, successfully finds the global minimum, and thus exponentially outperforms SA (and QAO for certain parameter regimes, see Table 5.1) on the same instance. It should be noted that the success of BBSA depends sensitively on the shape of the spike. In particular, we expect success (i.e. at least $1/\text{poly}(n)$ walkers reach the global minimum) when the part of the spike with positive slope (i.e. the "uphill" portion) has width $O(\log n)$.

5.10 Proof of Lemma 5.7.1

Let $\mathcal{C} = e^{-i\gamma \sum_w c(w)|w\rangle\langle w|}$ and let $\mathcal{R}, \mathcal{S}$ be defined analogously with the cost terms $r(w)$ and $s(w)$, where $c(w) = r(w) + s(w)$. $\mathcal{R}$ and $\mathcal{S}$ are mutually commuting, so $\mathcal{C} = \mathcal{R}\mathcal{S}$, and the first step of the QAOA1 protocol may be written as

$$\mathcal{C}\left|\psi_0\right\rangle = \mathcal{R}\mathcal{S}\left|\psi_0\right\rangle = \mathcal{R}\sum_{w=0}^{n} e^{-i\gamma s(w)} A_w \left|w\right\rangle = \mathcal{R}\left|\psi_0\right\rangle + \mathcal{R}\sum_{w=0}^{n}\left(e^{-i\gamma s(w)} - 1\right) A_w \left|w\right\rangle \tag{5.56}$$

After the mixing operator $\mathcal{B} = e^{-i\beta B}$ is applied, the final state is

$$\left|\psi_f\right\rangle = \mathcal{B}\mathcal{R}\left|\psi_0\right\rangle + \mathcal{B}\mathcal{R}\sum_{w=0}^{n}\left(e^{-i\gamma s(w)} - 1\right) A_w \left|w\right\rangle \tag{5.57}$$

The overlap with the global minimum $\left|\psi^*\right\rangle$ is

$$\left\langle\psi^*\right\rangle \psi_f = \left\langle\psi^*\right| \mathcal{B}\mathcal{R}\left|\psi_0\right\rangle + \left\langle\psi^*\right| \mathcal{B}\mathcal{R}\sum_{w=0}^{n}\left(e^{-i\gamma s(w)} - 1\right) A_w \left|w\right\rangle \tag{5.58}$$

$$\implies \left|\left\langle\psi^*\right\rangle \psi_f - \left\langle\psi^*\right| \mathcal{B}\mathcal{R}\left|\psi_0\right\rangle\right| = \left|\left\langle\psi^*\right| \mathcal{B}\mathcal{R}\sum_{w=0}^{n} 2e^{i\gamma s(w)/2 - i\pi/2} \sin\left(\frac{\gamma s(w)}{2}\right) A_w \left|w\right\rangle\right| \tag{5.59}$$

Now, $p = \left|\left\langle\psi^*\right| \mathcal{B}\mathcal{R}\left|\psi_0\right\rangle\right|^2$, and let $p^* = \left|\left\langle\psi^*\right\rangle \psi_f\right|^2$, the success probabilities of QAOA1 on $r(w)$ and the full cost function $c(w)$, respectively. We wish to show that p^* is at least $p - o(p)$. Using the triangle inequality $|x| - |y| \leq |x - y|$ on the left side of Eq. 5.59, and Cauchy-Schwarz inequality $\left|\left\langle u\right\rangle v\right| \leq \left|\left\langle u\right\rangle u\right|^{1/2} \left|\left\langle v\right\rangle v\right|^{1/2}$ on the right side, we get

the following:

$$\sqrt{p} - \sqrt{p^*} \leq \sum_{w=1}^{n} 4|A_w|^2 \sin^2\left(\frac{\gamma s(w)}{2}\right)^{1/2} = \sqrt{q} \tag{5.60}$$

$$\implies p^* \geq p\left(1 - \sqrt{q/p}\right)^2 = p\left(1 - o(1)\right)^2 = p - o(p) \tag{5.61}$$

which proves the lemma.

Chapter 6: Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator

A promising near-term application of quantum devices is the production of highly entangled states with metrological advantage or with properties of interest for many-body physics and quantum information processing. One possible approach to produce useful quantum states is to use quantum devices to perform adiabatic quantum computing [26, 27], which in some cases may provide an advantage over classical approaches [188]. However, adiabatic quantum computing has stringent adiabaticity requirements that hinder its applicability on existing quantum platforms that have finite coherence times [189].

Alternatively, hybrid quantum-classical variational algorithms may approximately solve hard problems in realms such as quantum magnetism, quantum chemistry [24], and high-energy physics [190]. This is because the key resource of quantum computers and simulators is quantum entanglement, which is exactly what makes these many-body quantum problems hard. In a hybrid variational algorithm, entangled states are functions of variational parameters that are iteratively optimized by a classical algorithm. One example is the Quantum Approximate Optimization Algorithm [28], which consists of a "bang-bang" protocol that can provide approximate answers in a time-efficient way, using

devices with finite coherence times and without the use of error-correction [33, 191–194].

Similarly to adiabatic quantum computing, the QAOA protocol encodes the objective function of the optimization problem in a target spin Hamiltonian. The optimization steps of the QAOA are based on unitary evolution under the target Hamiltonian and a non commuting "mixing" operator. In general, the QAOA relies on a classical outer loop to optimize the quantum circuit, aided by physical intuition [30, 169, 195, 196] or observed structure of the variational parameters [194, 197, 198], producing fast, low-depth circuits for approximate solutions. The QAOA has also been proposed as an efficient way to produce entangled quantum states, such as the ground states of critical Hamiltonians, which gives access to their corresponding energies [199, 200].

In the work covered in this chapter, we employ a collection of interacting trapped-ion qubits to experimentally implement a specific instance of the QAOA, which is native to our quantum hardware. We focus on both the energy minimization of the quantum Hamiltonian and the combinatorial optimization of the corresponding classical problem. Both problems are encoded in the transverse field anti-ferromagnetic Ising Hamiltonian with long-range interactions:

$$H = \underbrace{\sum_{i<j} J_{ij}\sigma_i^x\sigma_j^x}_{H_A} + \underbrace{B\sum_i \sigma_i^y}_{H_B}. \tag{6.1}$$

Here we set the reduced Planck's constant $\hbar = 1$, σ_i^γ ($\gamma = x, y, z$) is the Pauli matrix acting on the i^{th} spin along the γ direction of the Bloch sphere, $J_{ij} > 0$ is the Ising coupling between spins i and j, which, in our case, falls off as a power law in the

distance between the spins, and B denotes the transverse magnetic field. It is well-known [201] that the Hamiltonian (6.1) exhibits a quantum phase transition for anti-ferromagnetic interactions with power law decay. One of the goals of this work is to find an approximation of the ground state energy both at the critical point $(B/J_0)_c$, where J_0 is the average nearest-neighbour coupling, and in the case of $B = 0$, optimizing the QAOA output for the classical Hamiltonian H_A. The realization of the QAOA entails a series of unitary quantum evolutions (see Fig. 1) under the non-commuting Hamiltonians H_A and H_B (defined under Eq. (6.1)) that are applied to a known initial state $|\psi_0\rangle$. The state obtained after p layers of the QAOA is:

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = \prod_{k=1}^{p} e^{-i\beta_k (H_B/J_0)} e^{-i\gamma_k (H_A/J_0)} |\psi_0\rangle \,, \tag{6.2}$$

where the evolution times (or, henceforth, "angles") β_k and γ_k are variational parameters used in the k-th QAOA layer to minimize the final energy $E(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \langle \boldsymbol{\beta}, \boldsymbol{\gamma}| H |\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$.

In order to implement the quantum optimization algorithm, each spin in the chain is encoded in the $^2\mathrm{S}_{1/2}$ $|F = 0, m_F = 0\rangle \equiv |\downarrow\rangle_z$ and $|F = 1, m_F = 0\rangle \equiv |\uparrow\rangle_z$ hyperfine "clock" states of a $^{171}\mathrm{Yb}^+$ ion (see Appendix A). In this work, depending on the number of qubits and measurements required, we employ two different quantum simulation apparatus to run the QAOA, which will herein be referred to as system 1 [202] and system 2 [32] (see Appendix A). Both systems are based on a linear RF Paul trap where we store chains of up to $N = 40$ ions and initialize the qubits in the ground state of H_B, namely the product state $|\uparrow\uparrow \cdots \uparrow\rangle_y \equiv |+\rangle^{\otimes N} = |\psi_0\rangle$, where $|\uparrow\rangle_y \equiv (|\uparrow\rangle_z + i\,|\downarrow\rangle_z)/\sqrt{2}$ and B is assumed to be negative. The unitary evolution under H_A is realized by generating

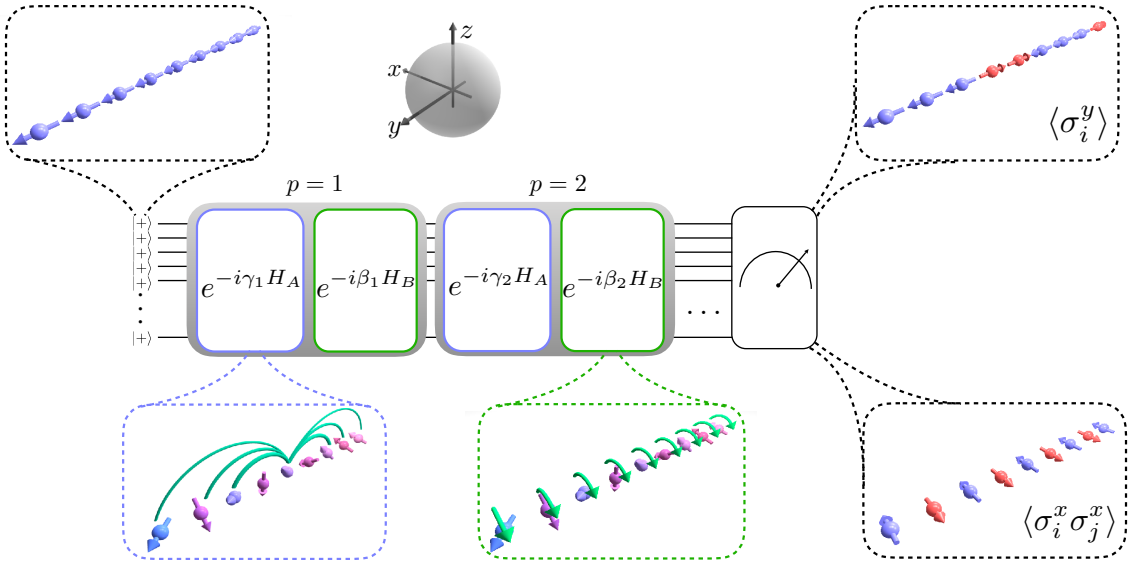


Figure 6.1: QAOA protocol. The system is initialized along the y direction in the Bloch sphere in the $|+\rangle^{\otimes N}$ state. The unitary evolution under $H_{A(B)}$ is implemented for angles $\gamma_i(\beta_i)$ for p times. At the end of the algorithm global measurements in the x and the y basis are performed to compute the average energy $\langle H\rangle = E(\boldsymbol{\beta},\boldsymbol{\gamma})$, which is compared to the theoretical ground state energy E_{gs}.

spin-spin interactions through spin-dependent optical dipole forces implemented by an applied laser field. This gives rise to effective long-range Ising couplings that fall off approximately as $J_{ij} \approx J_0/|i-j|^\alpha$ [203]. The power-law exponent $\alpha \sim 1$ and the interaction strengths vary in the range $J_0/2\pi$ =(0.3-0.57) kHz, depending on the system size and the experimental realization (see Appendix A for details). The unitary evolution under H_B is generated by applying a global rotation around the y-axis of the Bloch sphere.

After each run of the algorithm, we perform a projective measurement of each spin in the $x\,(y)$ basis to measure $\langle H_A\rangle$ ($\langle H_B\rangle$) (see Fig. 6.1). Measurements in the x and y bases are carried out by performing a $\pi/2$ rotation about the $y(x)$-axis of the Bloch sphere, illuminating the ions with resonant laser light, and collecting the σ_i^z-dependent fluorescence on a camera with site-resolved imaging. The energy is calculated by combining the measurements of the two-body correlators $\langle\sigma_i^x\sigma_j^x\rangle$ and the

total magnetization along the y axis $\sum_i \langle \sigma_i^y \rangle$, where the indices i, j range from 1 to N. We benchmark the experimental outcome $E(\boldsymbol{\beta}, \boldsymbol{\gamma})$ with the ground state E_{gs} of the target Hamiltonian (see Eq. 6.1) calculated numerically with exact diagonalization or Density Matrix Renormalization Group (DMRG) [204]. In order to quantify the performance of the QAOA, we use the dimensionless quantity

$$\eta \equiv \frac{E(\boldsymbol{\beta}, \boldsymbol{\gamma}) - E_{max}}{E_{gs} - E_{max}}, \tag{6.3}$$

where E_{max} is the energy of the highest excited state. This choice maps the entire many-body spectrum to the $[0, 1]$ interval. In the following we show that the best experimental performance η^* is close to the theoretical performance η_{th}, which itself is less than unity for a finite number p of QAOA layers.

6.1 Quantum Hamiltonian optimization

We first focus on the $p = 1$ optimization of the full quantum problem, where two variational parameters (γ and β) are used to minimize the energy of the Hamiltonian (1). In this case, the time-evolved one- and two-point correlation functions can be efficiently computed [205,206]. This leads to a general formula for the energy expectation under a state produced by the $p = 1$ QAOA that is used to compute the theoretical performance of the algorithm (see Appendix A). In Fig. 6.2a we show an experimental exhaustive search over the parameter space $\{\gamma, \beta\}$ and compare it to the theoretical performance of the algorithm, showing good agreement for $N = 20$ qubits. We also compare the performance of our algorithm as a function of B/J_0 with the expected QAOA

performance η_{th} (see Fig. 6.2b).

As shown in Ref. [201], for transverse fields greater than the critical value, the ground state is a low entanglement paramagnet, whereas below the critical point the ground state is an entangled superposition of anti-ferromagnetic states. We locate this critical point at $|B/J_0| = 0.31$ for 20 qubits by computing the half-chain entanglement entropy $S_{L/2} = -\mathrm{Tr}(\rho_{L/2} \log \rho_{L/2})$ of the ground state numerically, where $\rho_{L/2}$ is the half-chain reduced density matrix. As shown in Fig. 6.2b, while the experimental performance is $\eta > 94\%$ when $|B/J_0|$ is above the critical point, the gain relative to the initial state $|\psi_0\rangle$ is modest. On the other hand, below the critical point, the target state is more entangled, which allows for a larger experimental performance gain, at the expense of a reduced absolute performance. In order to quantitatively assess the gain over the finite initial state performance, we introduce a performance natural scale based on the quantity $\sigma_\eta(J_0, B, N)$, namely the standard deviation around the mean performance achieved implementing a QAOA protocol with random angles (see Appendix A for details). For $N = 20$ and $B/J_0 \sim -0.3$, $\sigma_\eta \sim 2 \times 10^{-3}$, our experimental performance at the critical point η^* is more than $20\sigma_\eta$ away from the initial state. On the other hand, the discrepancy between the ideal and experimental performance can be explained by taking into account our noise sources in the numerics (see Fig. 2c and the Combinatorial Optimization section below).

We investigate the performance of the $p = 1$ QAOA algorithm as a function of the number of qubits. For each system size, we ensure that the spin-spin couplings J_{ij} have the same dependence on the qubit distance $|i - j|$ by varying the trap parameters (see Appendix A). As shown in the inset of Fig. 6.2d, the half-chain entanglement entropy

as a function of system size N exhibits a peak located at $B/J_0 \sim -0.33$, displaying the onset of the phase transition as N tends to infinity. For all system sizes, we optimize the algorithm by performing a scan of the interaction angle γ and applying discrete variations of the mixing angle β around the optimal value predicted by the theory. In Fig. 6.2d, we compare the optimal experimental and theoretical performances η for different system sizes from 20 up to 40 qubits for fixed $B/J_0 \sim -0.3$. We observed experimentally that the QAOA yields a similar performance as a function of number of qubits even if the algorithm runtime stays approximately constant as the number of qubits increases. Numerically, we found that the performance η scales polynomially with N and with the number of layers p [207] (see Appendix A). Assuming extrapolation to higher numbers of qubits holds, this scaling, combined with a polynomial-time search heuristic, suggests that for any desired energy threshold ϵ, our approach allows us to approximate the energy to a degree $\eta > 1 - \epsilon$ in time and number of layers that scale as poly$(N, 1/\epsilon)$.

We experimentally perform a search for the optimal $p = 2$ QAOA performance using 20 qubits. Unlike the $p = 1$ case, there is no known analytic formula to efficiently compute the energy. However, exploiting relationships between optimal angles as a function of increasing p, we use a bootstrapping heuristic (see Appendix A for details) that allows the experiment to identify a set of optimal angles faster than a global parameter search. The bootstrapping heuristic computes a guess for optimal angles at p given optimal angles at lower p. A local optimizer, such as the greedy gradient descent described below, is then needed to take this guess to the true optimum. Our new heuristic method allows us to find variational parameters in time that scales polynomially with the number of layers and sublinearly in the number of qubits (when used in conjunction with the

quantum device).

We start from the optimal guess and perform a fine scan of γ_2, while varying γ_1, β_1 and β_2 in larger steps. The result is shown in Fig. 6.2d, where we plot the performances η as a function of γ_2 for every set of parameters used in the experiment. Fig. 6.2d shows also a color plot of all the optimal energies found as a function of the other three parameters γ_1, β_1 and β_2. The $p = 2$ QAOA performance with 20 qubits $\eta^* = (93.9 \pm 0.3)\%$ is in agreement with the $p = 1$ performance in system 2, taken with the same parameters (see Fig. 6.2c). This indicates that decoherence and bit-flip errors (see Appendix A) accumulated during longer evolution times are already balancing out the 2% expected performance gain of one additional optimization layer.

As a brute force approach is inefficient, we implement a closed-loop QAOA by interfacing the analog trapped-ion quantum simulator with a greedy gradient-descent algorithm to optimize the measured energy. In the $p = 1$ QAOA, we can visualize the optimization trajectory on the theoretical performance surface as shown in Fig. 6.3. Starting from a guess $(\beta^{(0)}, \gamma^{(0)})$, we measure the approximate local gradient by performing the energy measurements in two orthogonal directions $\beta^{(0)} + \delta\beta$ and $\gamma^{(0)} + \delta\gamma$ to compute the new guess $(\beta^{(1)}, \gamma^{(1)})$, where we measure the new energy on the quantum simulator. As shown in Fig. 6.3, the algorithm converges after about 10 iterations. Compared to an exhaustive search, the gradient descent uses fewer queries to the quantum simulator and is therefore more robust to slow drifts in the experimental system. For this reason, we are able to achieve a better performance compared to the exhaustive search method.

6.2 Combinatorial optimization

We further explore the performance of the trapped-ion system by investigating the combinatorial optimization of the classical Hamiltonian H_A (see Eq. (1) with $B = 0$) approximately sampling the output of the $p = 1$ QAOA, using high-fidelity, single-shot measurement of all the qubits. It has been proven, under reasonable complexity-theoretic assumptions, that no classical algorithm can efficiently sample exactly from a sufficiently general class of $p = 1$ QAOA circuits [191]. Recent results [208, 209] suggest that this could also hold in the case of approximate sampling (see Appendix A). In this case, by measuring in the x basis, it is possible to sample the probability distribution of all the 2^N eigenstates $|x_i\rangle$ of the Hamiltonian H_A. We performed the experiment with 12 qubits so that we can both compute the expected QAOA theoretical output and also experimentally over-sample the Hilbert space of all the possible $2^{12} = 4096$ possible outcomes. In Fig. 6.4a we show on a log scale the QAOA eigenstates probability distribution using the optimal variational parameters β^*, γ^* and compare the experimental eigenstate histogram with the exact diagonalization prediction of the QAOA output state, sorting the eigenstates according to their energies.

However, sampling from the full QAOA output distribution is a daunting task, since the experimental outcome is extremely sensitive to fluctuations in the Hamiltonian parameters and to experimental errors caused by detection and phonon-assisted bit-flip events and unwanted effective magnetic fields along the z direction of the Bloch sphere caused by uncompensated light shift (see also Appendix A). Given our measured experimental parameters, we can calculate the effect of these errors on the quantum

evolution, resulting in a good agreement with the experimental outcome, as shown in Fig. 4a.

Another useful way to compare numerics and experimental data is to implement the coarse-graining procedure of the Hilbert space proposed in Ref. [210]. After sorting in decreasing order the observed states according to their experimental probability, we iteratively group the states into "bubbles" of Hamming distance L around the most probable state, producing a coarse-grained dataset. We then apply the same coarse-graining to the theoretical probability distribution and plot the comparison in Fig. 6.4b. In this procedure the Hamming distance radius is varied to ensure that each bubble contains a comparable number of experimental shots, leading to bubbles of average Hamming distance $\bar{L} = 2.5$. In order to quantitatively compare the coarse-grained experiment and the theory, we use two different metrics, namely the total variation distance (TVD) and the Kullback-Leibler divergence ($\mathrm{D_{K-L}}$), defined as:

$$\mathrm{TVD} = \frac{1}{2}\sum_i |p_i - q_i|, \tag{6.4}$$

$$\mathrm{D_{K-L}} = -\sum_i p_i \log\left(\frac{q_i}{p_i}\right), \tag{6.5}$$

where $p_i(q_i)$ is the experimental (theoretical) probability of observing the i-th outcome. As shown in Fig. 6.4c, when the system is in the initial state, it is closer to a uniform probability distribution since $|\psi_0\rangle$ is an equal superposition of all the eigenstates of H_A. Indeed, at $\gamma = 0$, the TVD between the data $\{p_i\}$ and the uniform distribution is smaller than the one comparing the data and the ideal QAOA distribution $\{q_i\}_{\beta^*,\gamma^*}$. On the other hand, as the γ parameter is scanned, we observe a net decrease of both TVD and $\mathrm{D_{K-L}}$

between the experiment $\{p_i\}$ and the QAOA ideal distribution $\{q_i\}_{\beta^*,\gamma^*}$, in agreement with the decrease in energy, computed by measuring one and two-body correlators.

The variational quantum algorithm reported here, with up to 40 trapped-ion qubits, is the largest ever realized on a quantum device. We approximate the ground state energy of a non-trivial quantum Hamiltonian showing almost constant time scaling with the system size. Single-shot high-efficiency qubit measurements in different bases give access to the full distribution of bit-strings that is difficult or potentially impossible to model classically. With the addition of individual control over the interactions between qubits as well as improvements to fidelity and system size, the variational quantum-classical hybrid approach can be employed in this experimental platform to give insight into quantum chemistry [211–213] and hard optimization problems [214], such as Max-SAT or exact cover [215], or be used for the production of highly entangled states of metrological interest [216].

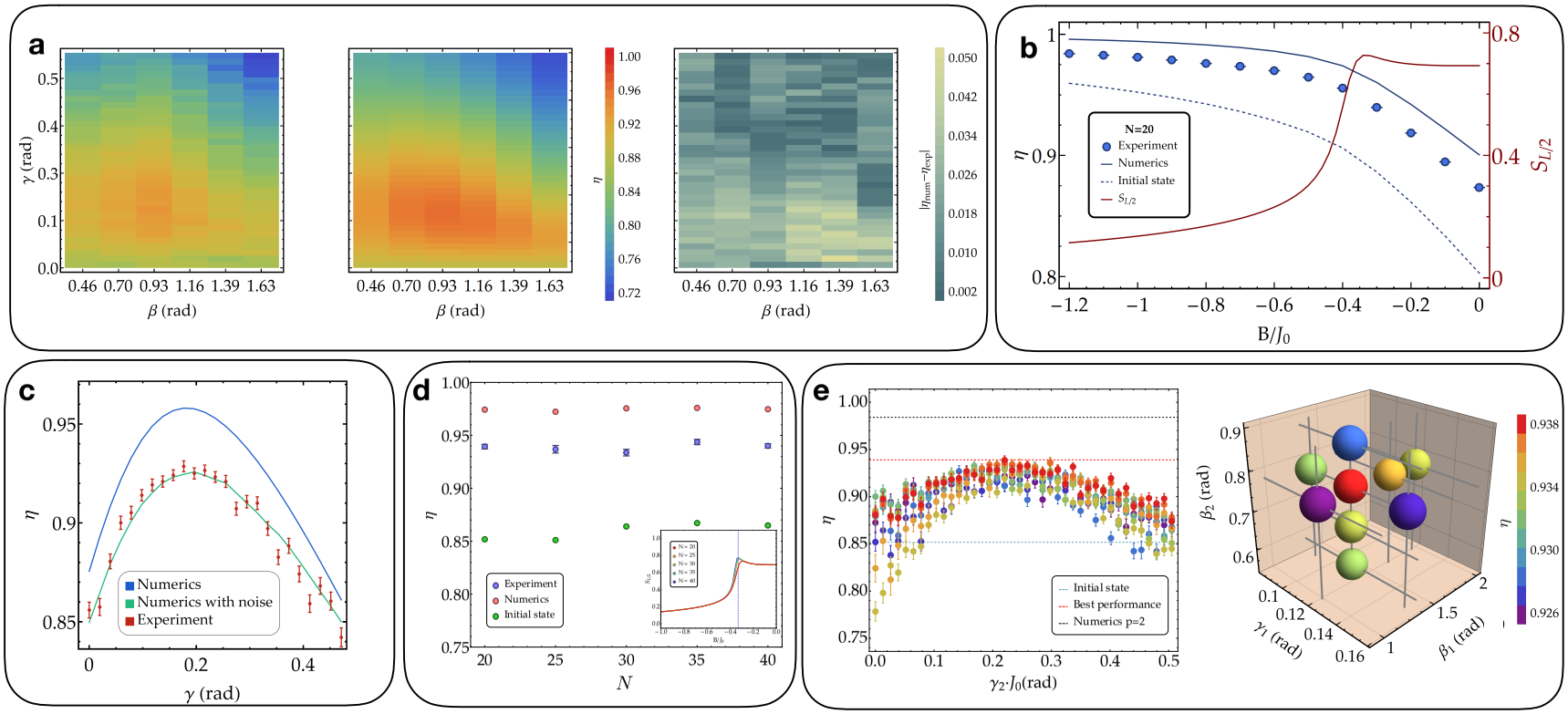


Figure 6.2: Exhaustive search for optimal performance. **(a)** Experimental (left) and theoretical (center) performance landscape and their absolute difference (right) as a function of the variational parameters β and γ for $N = 20$ qubits ($J_0/2\pi = 0.57$ kHz, $B/J_0 \sim -0.3$) , displaying an average absolute difference of 1.9% over 210 different $\{\beta, \gamma\}$ pairs. The optimal performance is $\eta^* = (93.8 \pm 0.4)\%$, whereas the theoretical performance is $\eta_{th} = 96.1\%$. Each data point is the result of 1100 (800) experimental repetitions to measure in the $x\,(y)$ basis (data taken on system 1). **(b)** Exhaustive search optimization as a function of B/J_0 (see Eq. (1)) (data taken on system 1). The dark red solid line is the half-chain entanglement entropy $S_{L/2}$ computed numerically with DMRG. The dashed blue line represents the performance of the initial product state $|\psi_0\rangle$. **(c)** Comparison between experimental performances and numerics for $B/J_0 = -0.25$ and $N = 12$ as a function of γ and $\beta^* = 1.12$. Taking into account bit-flip errors and slow drifts in the experimental parameters explains well the discrepancy between experimental and ideal performance (see Appendix A for details). **(d)** The $p = 1$ QAOA performance as a function of system size N up to 40 qubits (data taken on system 2). Comparison between QAOA experimental and theoretical performance for $B/J_0 \sim -0.3$. Green points show the baseline performance of the initial state $|\psi_0\rangle$. Inset: Convergence of the entanglement entropy peak as a function of number of qubits (see Appendix A). **(e)** $p = 2$ exhaustive search for $N = 20$ and $B/J_0 \sim -0.3$. Left: every color corresponds to a fine scan of γ_2 with a different set of variational parameters β_1, β_2 and γ_1 (data taken on system 2). Right: 3D color plot of the performance η, optimized over γ_2, as a function of the parameters β_1, β_2 and γ_1. The best outcome is $\eta^* = (93.9 \pm 0.3)\%$ (colored red), whereas the theoretical performance is $\eta_{th} = 98.4\%$ (see main text for details). In **(b),(c),(d)** and **(e)** the error bars are calculated by using the standard deviation from the mean of the measured performance.

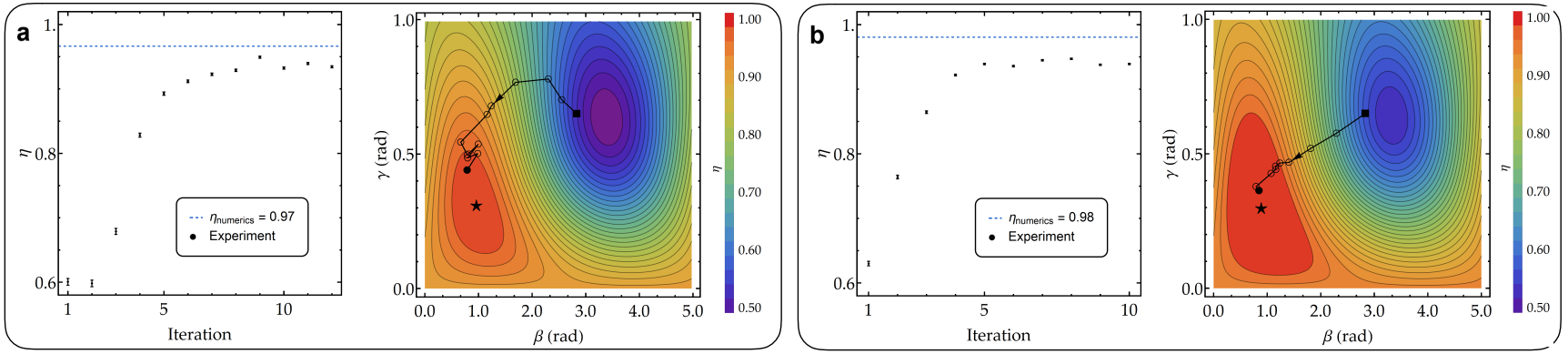


Figure 6.3: Gradient descent search for p=1 QAOA. (a) $N = 12$ and (b) $N = 20$. Left: performance η convergence as a function of iterations of the classical-quantum hybrid algorithm with (a) $N = 12$ ($J_0/2\pi = 0.57$ kHz, $B/J_0 = -0.3$) with a measured $\eta^* = (94.9 \pm 0.2)\%$ and (b) $N = 20$ qubits ($J_0/2\pi = 0.55$ kHz, $B/J_0 = -0.3$) with a measured $\eta^* = (94.7 \pm 0.1)\%$. Right: the algorithm trajectory on the theoretical performance landscape plotted as a function of γ and β. Each energy evaluation takes 4000 (6000) shots for 12 (20) qubits. The error bars are standard deviation from the mean of the measured performance (data taken on system 1).

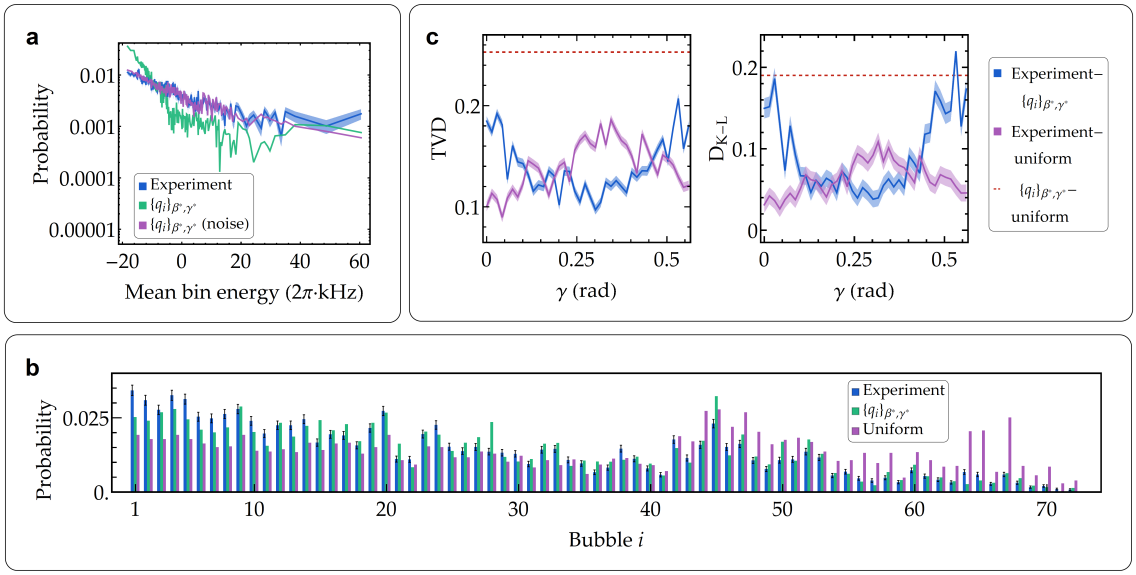


Figure 6.4: Sampling from $p = 1$ QAOA. **(a)** Eigenstate probability histogram for 12 qubits with $B = 0$. The numerical histogram is computed by decomposing the ideal QAOA output state on the $\{|x_i\rangle\}$ basis. We performed 10800 measurements to oversample the Hilbert space of dimension $2^N = 4096$ at the optimal parameters $\beta^* = 0.25$ and $\gamma^* = 0.31$. The 4096 eigenstates are grouped in bins of 20 for clarity purposes. The uncertainty bands follow the multinomial distribution standard deviation. Here $J_0/2\pi = 0.33$ kHz (see noise sources section in Appendix A for details). **(b)** Histogram of coarse-grained distributions (see main text for details) comparing data, theory and the uniform distribution. The error bars here also represent the standard deviation of the multinomial distribution. **(c)** Total Variation Distance and Kullback-Leibler divergence as a function of γ, keeping β fixed at the optimal value (1350 shots per time step). The non-zero TVD value of the violet curve at $\gamma = 0$ is due to state preparation and detection errors, as well as undersampling (see Appendix A). The distance from the uniform distribution increases as the γ parameter reaches the optimal point γ^*. Dashed lines are the comparison between the ideal QAOA distribution $\{q_i\}_{\beta^*,\gamma^*}$ and the uniform distribution. The uncertainty bands are based on the aforementioned error in the probability of each state bubble for the experimental distribution, propagated to the TVD and the $\mathrm{D}_{\mathrm{K-L}}$ according to Eq. (6.4) (data taken on system 2).

Chapter 7: Approximate optimization of the MaxCut problem with a local spin algorithm

Binary unconstrained optimization, i.e., the maximization of an objective function on the configuration space of binary variables, is an important NP-hard optimization problem whose restrictions include several problems from Karp's list of 21 NP-complete problems [217]. Due to the hardness of the problems, many solution approaches rely on finding approximately optima in the shortest possible time, or constructing algorithms that have an optimality guarantee but without guarantees on runtime. Algorithms in the latter category are often referred to as exact solvers, and include approaches that use linear, quadratic, or semi-definite programming (LP/QP/SDP) relaxations of the problem instance with techniques to obtain optimality bounds such as cutting planes, branch-and-bound, or Langrangian dual-based techniques. While the design of exact algorithms may be well-suited to theoretical analysis, the runtime scaling in instance size is often poor.

In some cases, it is possible to design polynomial-time algorithms with guarantees on the *approximation ratio*, i.e., the ratio of the optimum obtained to the global maximum. These are, however, ultimately limited by hardness results on achieving approximation ratios above a certain threshold value [218].

In the absence of runtime or optimality guarantees, problem-specific heuristics can

nevertheless perform better than expected, exhibiting superior performance in runtime, optimality or both. An important class of heuristics takes inspiration from physical processes seen in nature, and those in this category that mimic the evolution of quantum systems are known as *quantum-inspired optimization* methods.

Quantum-inspired (or "dequantized") algorithms have arisen in recent years of out a rich interplay between physics and algorithms research in the context of quantum computing. Thus, as notions of complexity now find analogues in many-body systems, so do quantum dynamics inform the design of quantum and classical algorithms. In the area of classical optimization, two quantum algorithms have generated considerable interest: quantum annealing and the quantum approximate optimization algorithm (QAOA). Promising developments in quantum annealing have inspired classical heuristic algorithms such as simulated quantum annealing [186] and sub-stochastic Monte Carlo [219], both of which mimic the evolution of the quantum state under an adiabatically evolving Hamiltonian.

Recent results on the performance of shallow-depth QAOA on the problem of MAX-E3-LIN2 led to improved approximations of corresponding classical algorithms for the same problem [28, 220]. More recently, a new classical heuristic known as Local Tensor (LT) was introduced in [33], taking inspiration from QAOA and closely related to previously known classical heuristics for distributed computing [221]. It was shown in [33] that LT has average-case performance better than single-layer QAOA for triangle-free MAXCUT and MAX-K-LIN2 by tuning only one global hyperparameter, in contrast to the two-parameter tuning required for QAOA. Currently, it is unknown whether LT may be useful as a heuristic more broadly, and if so, how the hyperparameters should be set in

practice. In this chapter, we address this question by implementing a version of LT and benchmarking it on the problem of MAXCUT. We find that on the instances studied, the performance of LT can be considerably enhanced by hyperparameter tuning, and that it is possible to provide good initial guesses on the hyperparameters as function of the instance description. Under such settings, the performance of LT is comparable to the performance of the commercially available solver, Gurobi.

Our setup is described in Secs. 7.1 and 7.2, followed by a discussion of hyperparameter tuning and the underlying physics of the algorithm in Secs. 7.4 to 7.6, respectively. Then, we provide a brief description of the problem instances studied (Sec. 7.3), and compare the performance of tuned LT with those of Gurobi (Sec. 7.7) and gradient descent (Sec. 7.8).

7.1 Spin problems

We refer to binary unconstrained optimization problems on spin degrees of freedom $s_i \in \{-1, 1\}$ as spin problems. Most generally, one can express the objective function as a polynomial in the variables. Furthermore, since higher powers of the binary variables are trivial, the polynomial is guaranteed to be degree at most one in each variable. The objective function to be maximized can therefore be viewed (up to a negative sign) as a Hamiltonian of a spin system, and the optimization problem maps to sampling from the ground state of the Hamiltonian. The cost Hamiltonian for a system of n spins with

indices $\{1, 2, \ldots, n\}$ can be written as

$$H = \sum_{\alpha} w_\alpha \prod_{i \in \alpha} s_i. \tag{7.1}$$

Therefore, H is a sum of monomials, or *clauses*, where a clause α is supported on the subset of spins $\alpha \subseteq \{1, \ldots, n\}$. The sum is weighted by clause weights w_α. The problem can be fully specified as a weighted hypergraph $G = (V, E, W)$, on vertices $V = \{1, 2, \ldots, n\}$, hyperedges $E = \{\alpha, \cdots\}$, and clause weights $W = \{w_\alpha, \ldots\}$.

A wide range of optimization problems be cast as binary unconstrained maximization problems [222], making this problem description very versatile. A simple (and commonly studied) case is one where the polynomial (7.1) is quadratic, i.e. $|\alpha| \leq 2$ for every α. This case captures several interesting physical systems such as Ising spin glasses, as well as a wide range of graph optimization problems. In this work, we focus on a particular quadratic spin problem, MAXCUT, defined in the following manner. Given a weighted graph $G = (V, E, W)$, we define a *cut* to be a partition of the vertices of the graph into two sets. The weight of the cut (or simply the cut) is then defined as the sum of weights of edges going across the cut. Therefore, for any $A \subset V$, the cut is

$$F(A) := \sum_{i \in A, j \in \bar{A}} w_{ij}. \tag{7.2}$$

Then, given a graph G, MAXCUT asks for the largest cut of the graph. To show that MAXCUT can be written as a quadratic spin problem, we consider the following encoding: Assign a spin s_i to vertex i. Then, there is a one-to-one mapping between bipartitions of

V, $(A, \bar{A})$, and spin configuration, $\mathbf{s} = (s_1, s_2, \ldots, s_n)$, namely, by setting $s_i = +1$ if $i \in A$ and -1 otherwise. Edges ij that lie wholly in either A or $\bar{A}$ do not count towards the cut, while edges between A and $\bar{A}$ do. In terms of the spins, edge ij will count towards the cut iff the spins s_i, s_j have opposite sign. Therefore, we may express the MAXCUT Hamiltonian in the following manner:

$$H_{\text{MAXCUT}} = \frac{1}{4} \sum_{i,j} w_{ij} \cdot (s_i s_j - 1). \tag{7.3}$$

$$\equiv \frac{1}{2} \mathbf{s}^T \cdot J \cdot \mathbf{s} \tag{7.4}$$

where $J_{ij} := w_{ij}/2$ with zero diagonal terms, $J_{ii} = 0$, and the last equivalence is an equality up to a constant offset $-\frac{1}{4} \sum_{i,j} w_{ij}$. The cut size for any configuration is the negation of the energy under H_{MAXCUT}. Notice that the ground state of H_{MAXCUT} corresponds to the largest cut in G. Since we are ultimately interested in the maximization problem, we will denote the negation of the energy by E.

Despite its simple statement (and apparent similarity to the polynomial-time solvable problem of MINCUT), MAXCUT is known to be NP-hard [217]. In fact, assuming the unique games conjecture holds, approximating MAXCUT to within a fraction 0.878.. is NP-hard [223]. This is also the best known performance guarantee, achieved by the exact classical algorithm due to Goemans and Williamson (GW) on MAXCUT with non-negative weights. Custom solvers for MAXCUT that that improve on practical performance while sometimes preserving optimality are known [224–226].

MAXCUT is a well-studied problem and often used as a benchmark for new classical,

quantum, and quantum-inspired solvers. Benchmarking of certain quantum-inspired optimization methods such as the coherent Ising machine [227] and the unified framework for optimization or UFO (see, e.g., [228]) has yielded promising results. In the following section, we discuss the LT heuristic framework and set up our implementation of the algorithm.

7.2 Local Tensor framework

Before describing our implementation, we review the local tensor (LT) algorithm framework laid out in [33]. The LT framework provides a general prescription for a class of *local* algorithms for the optimization of a Hamiltonian on spin variables. In a local algorithm, the state (e.g., a spin configuration) is encoded into the nodes of a graph, and the update rule at every node is local in the graph structure, depending only on nodes that are at most a bounded distance away. Local state updates therefore require information transfer among small neighborhoods and not the entire graph. If the graph has bounded degree, this can provide polynomial savings in the running cost of the algorithm. Additional speedup can be obtained in a true distributed model of computing where each node is an individual processor, and communication among nodes is slow compared to the internal operations of each processor.

LT is a local algorithm framework for optimization problems on spin degrees of freedom (such as MAXCUT). In LT, we first relax the domain of every spin variable from the binary set $\{-1, 1\}$ to a continuous superset such as the real interval $[-1, 1]$. By convention, we denote *soft* spins (i.e. those in the continuous domain) by letters u, v, etc.

and hard spins by letters r, s, etc. Then, LT simulates dynamics of a soft spin vector $\mathbf{v}$ in discrete time steps, and, at the end of a total number of steps p, retrieves a hard spin configuration $\mathbf{s}$ from the final state via a rounding procedure applied to the soft spins. There is considerable flexibility in this setup, and for ease of study, we construct a specific instance of LT here.

Suppose we are given a MAXCUT instance whose corresponding Hamiltonian (as in (7.4)) is H. Denote the state of spin i at time t by $v_{i,t}$, and the full state vector by $\mathbf{v}_t = (v_{1,t}, v_{2,t}, \ldots, v_{N,t})$. Then, we perform the following steps in order, simultaneously for all spins $i = 1, \ldots, N$.

1. Initialize all spins uniformly at random, $v_{i,0} \in [-1, 1]$.

2. For $t = 0, 1, \ldots, p-1$, update $v_{i,t} \mapsto v_{i,t+1}$ as follows:

   (a) $v_{i,t.5} = v_{i,t} + cF_{i,t}$ where $F_{i,t} := -\partial H / \partial v_{i,t}$ and c is a real constant.

   (b) $v_{i,t+1} = \tanh(\beta v_{i,t.5})$, where β is a positive constant.

3. After p rounds, round each spin to its sign, $v_{i,p} \mapsto s_i^* = \operatorname{sgn}(v_{i,p}) \in \{-1, 1\}$. Return s_i^*.

The final configuration $\mathbf{s}^*$ is a feasible solution candidate. As there the initial configuration $\mathbf{v}_0$ is sampled at random, an outer loop carries out several independent runs of the algorithm and selects the best solution.

For an instance of size n, the domain of feasible solutions corresponds to the vertices of an n-dimensional hypercube. The relaxation in LT extends the domain to the full hypercube, which allows for small, incremental updates and a well-behaved cost

function, at the cost of making the search space infinite. However, the rounding step at the end of the algorithm offsets this drawback in the form of a lenient rounding rule: Return the nearest vertex of the hypercube. Therefore, the final state of the graph is only required to lie in the correct quadrant (or 2^n-ant, to be precise) in order to produce the optimal solution.

The spin update sequence is carried out for a total of p rounds, each consisting of two steps. The *force* $F_i = \partial H / \partial v_i$, calculated for each spin, displaces the spin by an amount proportional to it. We refer to the constant of proportionality c as the *response*. Next, we apply the nonlinear function $\tanh(\beta v)$ to the spin, with a rescaling factor β. This choice of notation is motivated by analogy to the inverse temperature β in classical thermodynamics. As discussed in the next section, the soft spin v can be inferred as the expectation of an ensemble of spin configurations. When β is small, the expected value v is close to zero (i.e., random), while for large β the spins are "frozen" (expected value close to ± 1). The number of rounds p, response c, and β form the hyperparameters of the algorithm, which must be fixed (ideally by optimization) before the algorithm is run on an instance. In theory, the factors c, β can also be made to vary by round under a predetermined or adaptive schedule, in a manner similar to simulated annealing. Here, however, we will consider them to be constant in time.

7.3 Spin model instances

In order to test the dynamics and the performance of LT, we choose one of several online repositories of MAXCUT instances, the "Biq Mac" library [229]. Each

instance therein is a random graph with edge weights drawn from a particular probability distribution. In addition to the weight distribution, the instances is parameterized by the number of variables n and edge density d (i.e. the expected number of non-zero weight edges). The instance data is tabulated in Table 7.1.

Instance type	**Tag**	**Weight distribution**	**Instance size (n)**	**Clause density (d)**
`g05_`n	`g05`	$w_{ij} \in \{0, 1\}$	$60, 80, 100$	0.5
`pm1s_`n	`pm1s`	$w_{ij} \in \{-1, 0, 1\}$	$80, 100$	0.1
`pm1d_`n	`pm1d`	$w_{ij} \in \{-1, 0, 1\}$	$80, 100$	0.5
`w`d`_`n	`w`	$w_{ij} \in [-10, 10]$	100	$0.1, 0.5, 0.9$
`pw`d`_`n	`pw`	$w_{ij} \in [0, 10]$	100	$0.1, 0.5, 0.9$
`ising2.5-`n	`ising2.5`	$w_{ij} \propto \frac{\epsilon_{ij}}{\lvert j-i\rvert^{2.5}}$, $\epsilon_{ij} \sim \mathcal{N}(0, 1)$	$100, 150, 200, 250, 300$	–
`ising3.0-`n	`ising2.5`	$w_{ij} \propto \frac{\epsilon_{ij}}{\lvert j-i\rvert^{3.5}}$, $\epsilon_{ij} \sim \mathcal{N}(0, 1)$	$100, 150, 200, 250, 300$	–
`t2g`L	`torus`	2-dim. toroidal grid, $w_{\langle ij\rangle} \in \{-1, 1\}$	$L^2, L = 5, 6, 7$	$\frac{4}{n-1}$
`t3g`L		3-dim. toroidal grid, $w_{\langle ij\rangle} \in \{-1, 1\}$	$L^3, L = 5, 6, 7$	$\frac{6}{n-1}$

Table 7.1: The benchmarking instances. Each instance is a random graph on n vertices whose edge weights are chosen from the given distribution. The first column specifies the formatting of instance names, while the second column provides a shorter tag for all instances of a given type. In the case of the `w` instances, we sometimes group the instance type `w` by clause density d, in which case the instances are tagged as `w`d, where $d = 0.1, 0.5, 0.9$. The instance types `g05`, `pm1s`, `pm1d`, `w`, and `pw` are random graphs on n vertices with clause density d, where edge weights are drawn from the distrubtion in column 3. The `ising` instances are a 1-dimensional Ising model with long-ranged interactions falling off as a power (2.5 or 3.0) of the inter-spin distance, with a numerator sampled from the normal distribution. The `torus` instances are periodic, D-dimensional ($D = 2, 3$) spin lattices with random couplings ± 1 along the edges of the lattice (denoted by $\langle ij\rangle$ for two neighboring vertices i, j).

7.4 LT as a discretized, imaginary-time Schrödinger evolution

LT describes a particular discrete-time evolution of a spin system under a Hamiltonian H. In this section, we provide a physical underpinning to these dynamics by showing that evolution under LT is closely related to imaginary-time Schrödinger evolution under H.

Given any initial state $|\psi\rangle$ and Hamiltonian H, time-evolution of $|\psi\rangle$ under H is given by the Schrödinger equation $d|\psi\rangle/dt = -iH|\psi\rangle$. The evolution applies a phase to the eigenstates of H proportional to the energy of the state times time, so that low-energy states rotate slowly while highly excited states rotate fast. An analytical tool often employed to access the low-energy spectrum of H is that of analytic continuation to imaginary time. In this, one replaces the time by an imaginary time parameter $\tau := it$, and the (unnormalized) imaginary time Schrödinger equation reads

$$|\dot{\psi}\rangle \equiv d|\psi\rangle/d\tau = -H|\psi\rangle \ . \tag{7.5}$$

The formal solution to this equation is $|\psi(\tau)\rangle = e^{-H\tau}|\psi(0)\rangle$. Note that $|\psi(\tau)\rangle$ is unnormalized, but we keep track of the normalization $\mathcal{N}(|\psi(\tau)\rangle) \equiv \mathcal{N}(\tau) := \sqrt{\langle\psi(\tau)\rangle\,\psi(\tau)}$. In the limit $\tau \to \infty$, and assuming that the ground state of H is non-degenerate, the exponential $e^{-\tau H}$ suppresses contributions from all but the lowest-energy state $|\psi_0\rangle$ of H, which implies that $\lim_{\tau\to\infty}|\psi(\tau)\rangle = |\psi_0\rangle$.

The normalization $\mathcal{N}(\tau)$ has τ-dependence

$$\dot{\mathcal{N}} = \frac{1}{2\sqrt{\langle\psi\rangle\,\psi}} \cdot \left(\langle\dot{\psi}\rangle\,\psi + \langle\psi\rangle\,\dot{\psi}\right) \tag{7.6}$$

$$= -\frac{\langle H\rangle}{\mathcal{N}} \tag{7.7}$$

where $\langle H\rangle := \langle\psi|\,H\,|\psi\rangle$ is the *unnormalized* expectation value of operator H. The normalized expectation value is given by $\langle\langle H\rangle\rangle := \langle H\rangle/\mathcal{N}^2$.

Next, let H be a Hamiltonian acting on n qubits that is diagonal in the Z basis. Any state $|\psi\rangle$ in this Hilbert space can be mapped to a vector of normalized expectation values of the Pauli operators Z_i, where the index i runs over all spins:

$$\begin{aligned} |\psi(\tau)\rangle &\mapsto (\langle\langle Z_1\rangle\rangle, \langle\langle Z_2\rangle\rangle, \ldots, \langle\langle Z_n\rangle\rangle) \\ &=: (v_1, v_2, \ldots, v_n)\ , \end{aligned}$$

where $v_i \in [-1, 1]$ is the classical spin variable that tracks the normalized expectation of Z_i. The imaginary time-evolution of the spins is given by

$$\dot{v_i} = \frac{d}{d\tau}\left(\frac{\langle Z_1\rangle}{\mathcal{N}^2}\right) \tag{7.8}$$

$$= \frac{-2\dot{\mathcal{N}}}{\mathcal{N}^3}\langle Z_i\rangle + \frac{1}{\mathcal{N}^2}\frac{d}{d\tau}\langle\psi|\,Z_i\,|\psi\rangle \tag{7.9}$$

$$= 2v_i\langle\langle H\rangle\rangle - \langle\langle HZ_i + Z_iH\rangle\rangle\,. \tag{7.10}$$

This is essentially an imaginary-time analogue of the Ehrenfest theorem. Since Pauli

operators Z_i square to the identity, a diagonal Hamiltonian H can always be written as

$$H = R_i Z_i + S_i \,, \tag{7.11}$$

for every site i, for some operators R_i, S_i that are not supported on site i. Then, $HZ_i = Z_i H = R_i + S_i Z_i$. Next, we make a mean-field assumption, $\langle\langle H_{\bar{i}} Z_i \rangle\rangle \approx \langle\langle H_{\bar{i}} \rangle\rangle \cdot \langle\langle Z_i \rangle\rangle$, where $H_{\bar{i}}$ is any local operator not supported on site i. Then, it follows that $\langle\langle HZ_i \rangle\rangle \approx \langle\langle R_i \rangle\rangle + v_i \cdot \langle\langle S_i \rangle\rangle$, and $\langle\langle H \rangle\rangle \approx v_i \cdot \langle\langle R_i \rangle\rangle + \langle\langle S_i \rangle\rangle$, which gives

$$\dot{v_i} = 2\left(v_i^2 \cdot \langle\langle R_i \rangle\rangle + v_i \langle\langle S_i \rangle\rangle - \langle\langle R_i \rangle\rangle + v_i \cdot \langle\langle S_i \rangle\rangle\right) \tag{7.12}$$

$$= -2\left(1 - v_i^2\right) \cdot \langle\langle R_i \rangle\rangle \,. \tag{7.13}$$

Next, we make a substitution $u_i := \tanh^{-1} v_i$ which maps the real line onto the open interval $(-1, 1)$. Then, $\dot{v}_i = 1 - \text{sech}^2(u_i)\dot{u}_i = (1 - v_i^2)\dot{u}_i$, therefore we can write

$$\dot{u}_i = -2\langle\langle R_i \rangle\rangle. \tag{7.14}$$

Note now that the term $\langle\langle R_i \rangle\rangle$ is precisely the negative expected value of the force on spin i, $dH/dZ_i = R_i = -F_i$. Finally, an imaginary time evolution discretized into small time steps $\delta\tau$ obeys (in mean field)

$$v_i(\tau + \delta\tau) = \tanh\left(2\delta\tau F_i + \tanh^{-1} v_i\right). \tag{7.15}$$

This equation bears similarity to the update rule for LT. In fact, for v_i sufficiently small

and close to steady state v_i^*, we can expand the inverse tangent as $\tanh^{-1} v_i \approx v_i + v_i^3/3 + \ldots \approx -2v_i^{*3}/3 + v_i \cdot (1 + v_i^{*2})$, which looks linear with a modified slope. In principle, one could directly evolve the u_i variables in time as per Eq. (7.15). However, the v_i have the advantage of being bounded in $[-1, 1]$, while the u_i are unbounded and may suffer from issues of convergence. The above analysis reveals a surprising connection between LT and a discretized, mean-field imaginary time evolution of classical spin expectation values.

Our analysis suggests a generalization of LT to situations where the Hamiltonian is not diagonal in the Z basis. In this context, we can represent each spin i as a 3D rotor $\mathbf{r}_i = (x_i, y_i, z_i) = (\langle\langle X_i \rangle\rangle, \langle\langle Y_i \rangle\rangle, \langle\langle Z_i \rangle\rangle)$ of Pauli expectation values. Since the Paulis square to the identity, a general spin Hamiltonian H can always be written as

$$H = P_i X_i + Q_i Y_i + R_i Z_i + S_i \,, \tag{7.16}$$

for every site i, where P_i, Q_i, R_i, S_i are some Hermitian operators that do not take support on site i. Then, $H_i Z_i + Z_i H_i = 2R_i + 2S_i Z_i$ (and analogously for X_i, Y_i), and therefore

$$\dot{x}_i = -2(1 - x_i^2) \cdot \langle\langle P_i \rangle\rangle \,, \tag{7.17}$$

and similarly for the other coordinates. More succinctly, if we define $\boldsymbol{\rho}_i := \big(\tanh^{-1} x_i, \tanh^{-1} y_i, \tanh^{-1} z_i\big)$, then the imaginary time evolution becomes

$$\dot{\boldsymbol{\rho}}_i = -2\mathbf{F}_i \tag{7.18}$$

where $\mathbf{F}_i = \left(\langle\langle \frac{dH}{dX_i} \rangle\rangle, \langle\langle \frac{dH}{dY_i} \rangle\rangle, \langle\langle \frac{dH}{dZ_i} \rangle\rangle\right) = (\langle\langle P_i \rangle\rangle, \langle\langle Q_i \rangle\rangle, \langle\langle R_i \rangle\rangle)$. Then, we can imagine a generalization of LT that discretizes the above equation and simulates the evolution of a 3D rotor. By "rounding" the expectation values of the final state, we arrive at a product state estimate of the ground state. The study of this generalized algorithm will be left as a subject of future work.

7.5 Hyperparameter optimization

In order to talk about the performance of LT on any given instance, we must first consider variations in performance due to parameter setting and randomness. LT (as implemented here) is a family of algorithms in the hyperparameters c, β, p. Moreover, for fixed hyperparameters, any run of the algorithm has randomness due to the choice of initial spin configuration. Therefore, the energy output at the end of a single run of LT is a random variable dependent on (c, β, p). The median final energy with fixed hyperparameters, however, is a determinate quantity, which we denote $E_{c,\beta,p}$.

$$E_{c,\beta,p} = \text{median}_{\mathbf{v}_0 \in [-1,1]^{\times n}} \text{LT}_{c,\beta,p}(\mathbf{v}_0) \tag{7.19}$$

where, abusing notation, $\text{LT}(\mathbf{v})$ denotes the output energy of LT with input configuration $\mathbf{v}$. By definition, half of the runs of LT are expected to produce an optimum with energy lower than E, making the median energy a useful figure of merit. The true median energy can be approximated in practice by the median value of M independent runs of $\text{LT}_{c,\beta,p}$,

$$\text{median}\left\{E_1, E_2, \ldots, E_M\right\} = \tilde{E} \approx E_{c,\beta,p} \tag{7.20}$$

Since we ultimately wish to study the performance of LT as a whole, the hyperparameters must be fixed via a well-defined procedure that takes as input the instance description and returns an (ideally optimal) hyperparameter setting. The most rigorous criterion is global optimization of the performance with respect to each hyperparameter independently. This is important, e.g. to avoid spurious trends in the runtime scaling that arise from imperfect hyperoptimization.

Since this is a computationally expensive task, we focus first on gaining a better understanding of the effect of the hyperparameter on the algorithm performance and providing formulas to minimize the resources needed for hyperparameter optimization.

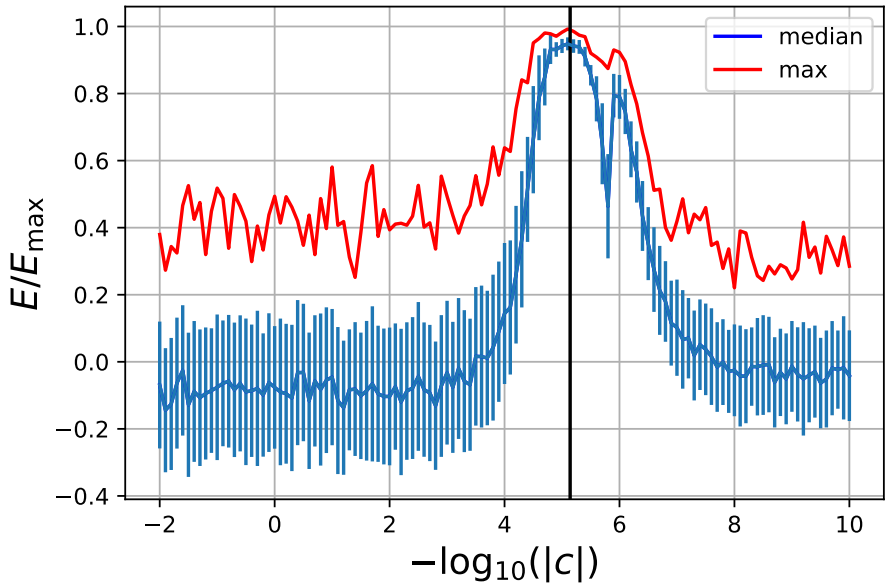


Figure 7.1: Hyperparameter sweep for the instance `ising2.5-100` (seed 5555). We vary c over multiple orders of magnitude, and plot the median (lower line) and max (upper line) value of optimum found, normalized by the global optimum, for several runs of the algorithm. It can be seen that peak performance occurs when $c \sim \bar{c}$ (indicated by the vertical black line).

Response c.

The response c is the sensitivity of the spins to force. Intuitively, setting c too large or too small would make the spins too responsive to displacement or frozen, respectively.

Therefore, we expect a regime for c values where the spins are optimally sensitive to the force, and the algorithm should also perform well in this regime. Given a typical length of spins $v_i \sim 1$ and maximum possible force on spin i, $F_i \sim \sum_{j=1}^{N} |J_{ij}|$, a natural guess for c is the inverse of the maximum force. We define

$$\bar{c} = 2 \left\langle \sum_{j=1}^{N} |J_{ij}| \right\rangle_i^{-1} \tag{7.21}$$

where the brackets $\langle \cdot \rangle_i$ denote a mean over all sites in the graph. The factor of two is chosen purely empirically. As shown in Fig. 7.1, we find that $\bar{c}$ is indeed a natural scale for the response, and optimal performance is typically found to be within an order 1 factor of $\bar{c}$. Hereafter, we use a rescaled hyperparameter $\eta := c/\bar{c}$.

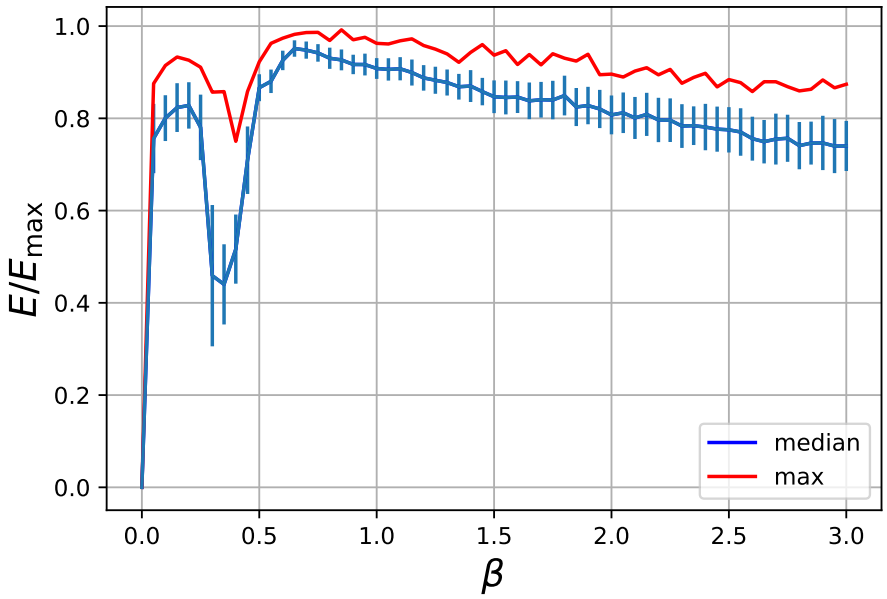


Figure 7.2: Hyperparameter sweep for the instance `ising2.5-100` (seed 5555). We plot the median (blue, lower) and max (red, upper) performance as a function of β (arb. units), for a fixed value of $c \sim \bar{c}$. The performance is sensitive to order 1 variation in β, varying from sub-random (cut fraction 0.5) to close to optimal at $\beta \simeq 0.7$. This behavior is typical across all instances studied.

β

The β parameter scales the value of the input to the $\tanh$ activation function. Intuitively, this enables mapping the displaced spin to the linear response region of the $\tanh$ function for maximum sensitivity. This also ensures that the spin stays of order 1 and therefore sensitive to forces applied in subsequent rounds.

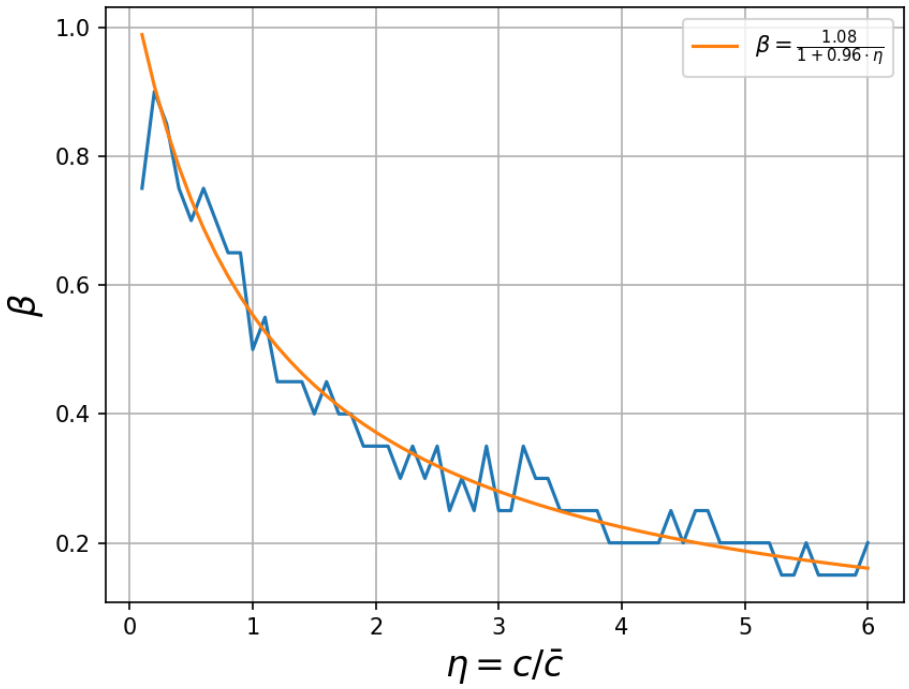


Figure 7.3: Optimal β (arb. units) for a range of η values, plotted for a `torus` instance. For each η, the optimal β was found by grid search. The fit to the functional form given in (7.22) is given by the smooth curve in the figure. The curve profile and quality of fit seen here are typical to all instances studied.

In Fig. 7.2, we show how the choice of β affects mean and best-case performance for a particular instance. The peaks suggest an optimal setting for the value of β. We now make an educated guess for β as we did for the response. For a fixed response $\eta = c/\bar{c}$, spin v_i is displaced as $v_i \to \tanh \beta(v_i + cF_i)$. Since $v_i \leq 1$ for all i, the argument of the tanh function must lie between $[-\beta(1+\eta), +\beta(1+\eta)]$. In order to be maximally sensitive to displacement, we should set β such that $\beta(1+\eta) \sim O(1)$. This gives us a functional dependence between β and η as

$$\beta = \frac{a}{1+b\eta}, \tag{7.22}$$

where we have introduced two fitting parameters a, b. From the available instances, this relationship can be checked by extracting the locus of optimal settings for (η, β) and

fitting them to the above functional form. The results are shown in Fig. 7.3. The quality of fits suggests that the functional dependence given in Eq. (7.22) is accurate. In Fig. 7.4, we show how the coefficients a, b cluster for different problems.

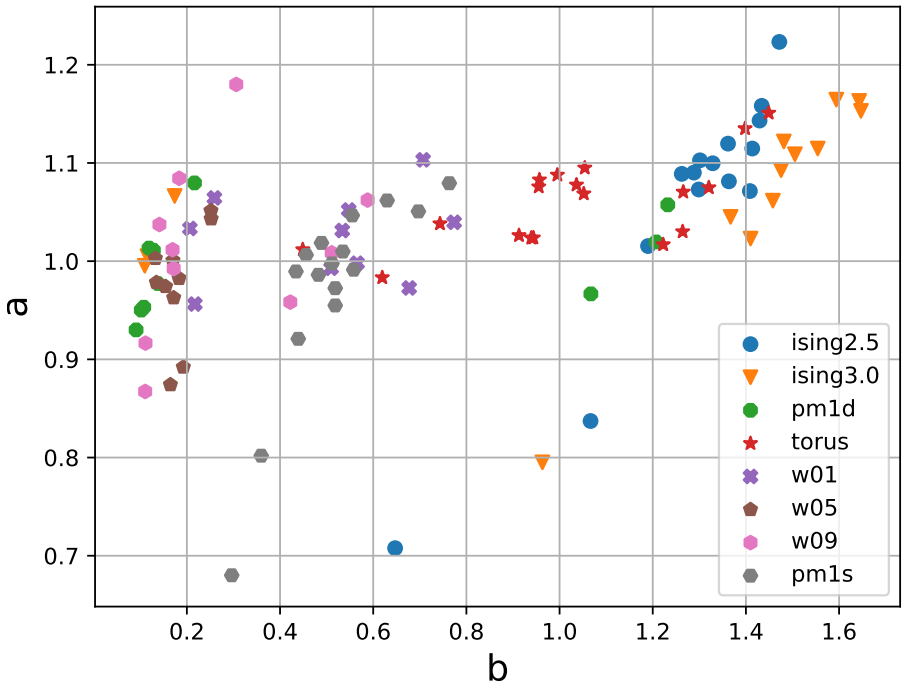


Figure 7.4: Clustering in the fit coefficients a, b in (7.22) for different instances (units arbitrary). We see that the fitting numerator a is close to 1 for most instances, while b varies considerably. There is a reasonable degree of clustering by instance type in b.

From this analysis, we see that given a problem instance, the hyperparameters η, β can be guessed with very little optimization, and tuned further, if necessary, by local search in a range of order 1 in each parameter.

Number of rounds p.

The number of rounds p required by the algorithm is dictated by the convergence to steady state. Qualitatively, this may be connected to the rate of information propagation in the graph, via quantities such as the girth. Unlike for c, β however, a direct guess for p

may be harder to obtain.

Instead, we use a dynamic criterion to set the value of p. Since LT is iterative and closely related to gradient descent, we expect that at some point during the algorithm, the spin vector attains a steady state such that all subsequent displacements are smaller than a given threshold. As the final state is determined by the quadrant containing the vector and not the exact value of the vector, small displacements have a small or no effect on the outcome.

In Fig. 7.5, we plot displacements between successive rounds of a subset of spins in a fixed instance. It is seen that displacements quickly become small; for instance, at $p = 50$, the displacements are of order 10^{-4}. This convergence in the spin values is seen across all spins in a given instance, and all instances studied. Therefore, once the displacement of the state falls below a set threshold, we terminate the algorithm. While this threshold is an additional parameter, it can be set to be sufficiently smaller than the size of the hypercube.

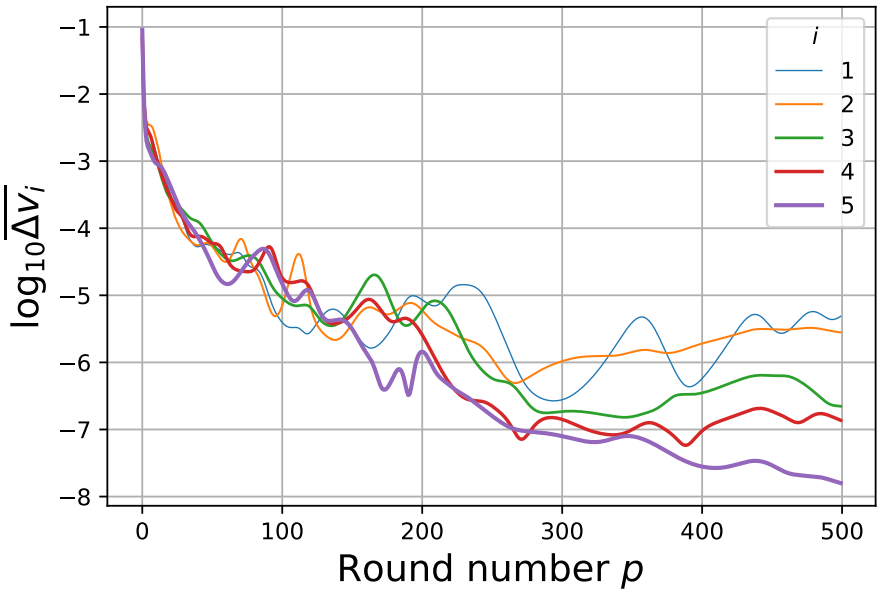


Figure 7.5: Displacement between successive rounds as a function of round number, for five arbitrarily chosen spins from the random, 100-spin instance of type `w05`. The log displacement approaches floating point precision after a short number of rounds, indicating that the updates can be terminated early for the rounding step.

7.6 Dependence of LT dynamics on the hyperparameters

The implementation of LT studied here allows three hyperparameters, namely, the number of rounds p, the response to force c, and β. We have discussed how to set these hyperparameters for a given MAXCUT instance, giving (in the case of c, β) good initial approximations that depend on the instance description, or (in the case of p) a dynamic criterion based on the convergence of the state vector. Here we give a physical description of the system dynamics and show, qualitatively, why it is reasonable to expect such behaviors.

7.6.1 Behavior for a small instance.

For a small instance on 5 spins, with real weights on every edge chosen at random, we show the evolution of the full spin configuration as a function of p. We see that the system always finds the same steady-state configuration regardless of initial state (some examples shown in Fig. 7.6). We can therefore plot how the steady state values of each spin change as a function of β and η, Fig. 7.7. What we see is that the system undergoes a transition in both parameters, from a phase with zero magnetization on each spin to a phase where the spins have a preferred direction. This is broadly consistent with the interpretation of β and η as being analogous to inverse temperature and interaction strength. At high temperatures or low interaction strength, the spins prefer an unmagnetized configuration while at low temperatures or higher interaction, they find non-zero steady state configurations that correspond to low-energy solutions of the optimization problem. From a computational standpoint, the latter regime is of most interest.

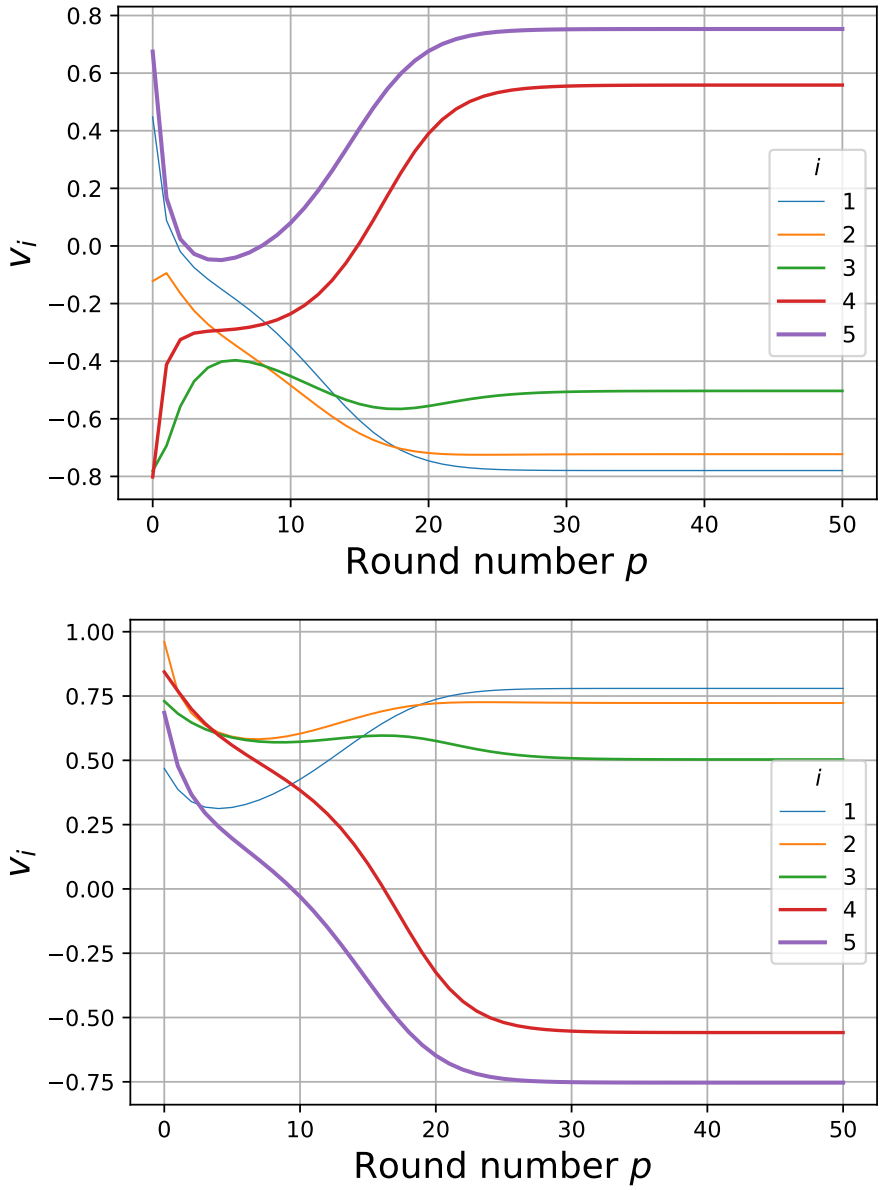


Figure 7.6: Evolution of spins for a small instance ($N = 5$) for two randomly chosen initial configurations (units arbitrary). Despite different initial states, the spins are seen to approach the same steady state solution, up to an overall sign. Larger instances may have multiple steady states.

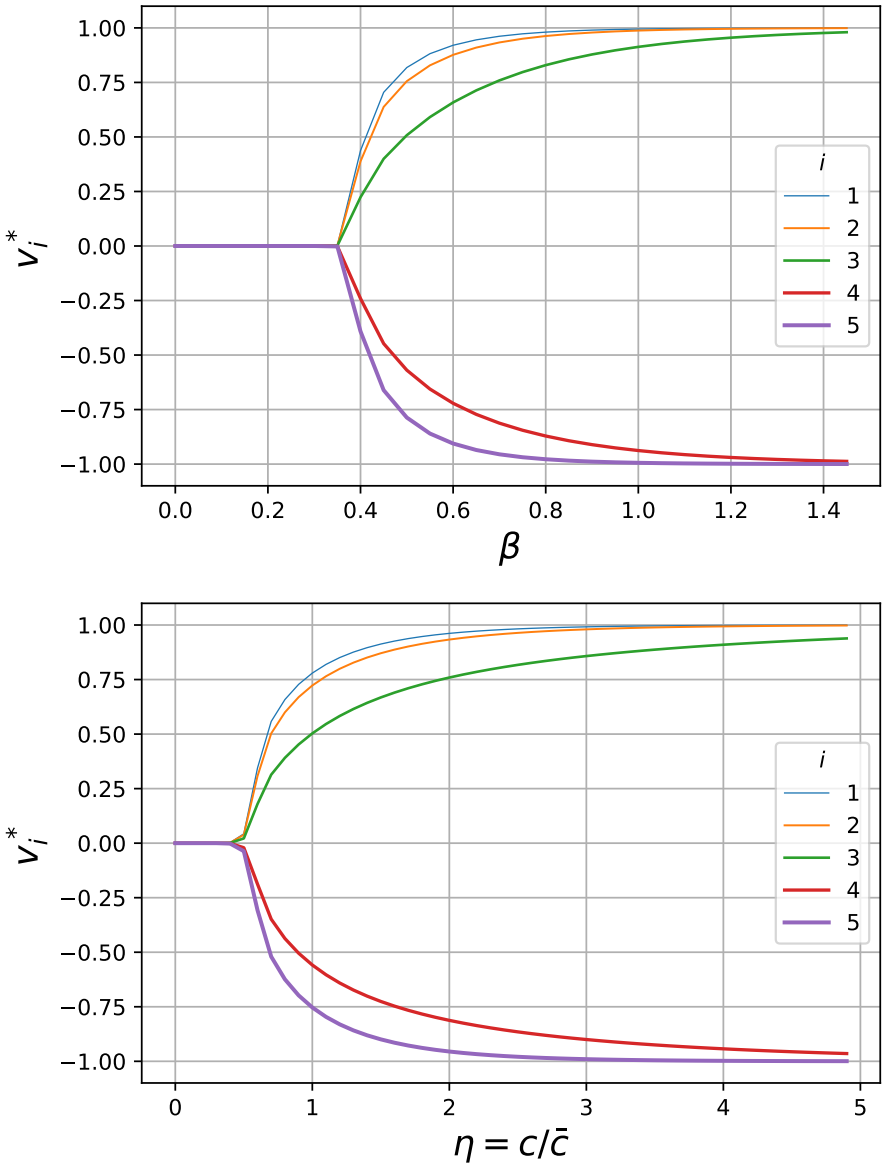


Figure 7.7: Steady-state spin configurations as a function of β (top, $\eta = 2$) and η (bottom, $\beta = 0.7$) for the same instance as Fig. 7.6 (units arbitrary). In both cases, the system transitions from an "unmagnetized" phase for low η, β to a "magnetized" phase at high η, β.

7.6.2 Dynamics near steady-state.

We will analyze the behaviour of LT near a steady state solution $\mathbf{v}^*$ that satisfies $v_i^* = \tanh\left[\beta\left(v_i^* + cF_i^*\right)\right]$ for all spins i. Note that a steady state always exists: the all-zero state $v_i^* = 0$ is an example. More generally, the transcendental equation for steady state,

while not guaranteed to have other solutions, can be approximated as a linear equation when $v_i^* \ll 1$, which has non-zero solutions for particular values of β, c. Generically, we expect other steady solutions lying within the hypercube, and find this to be true in our numerics (e.g., Fig. 7.6).

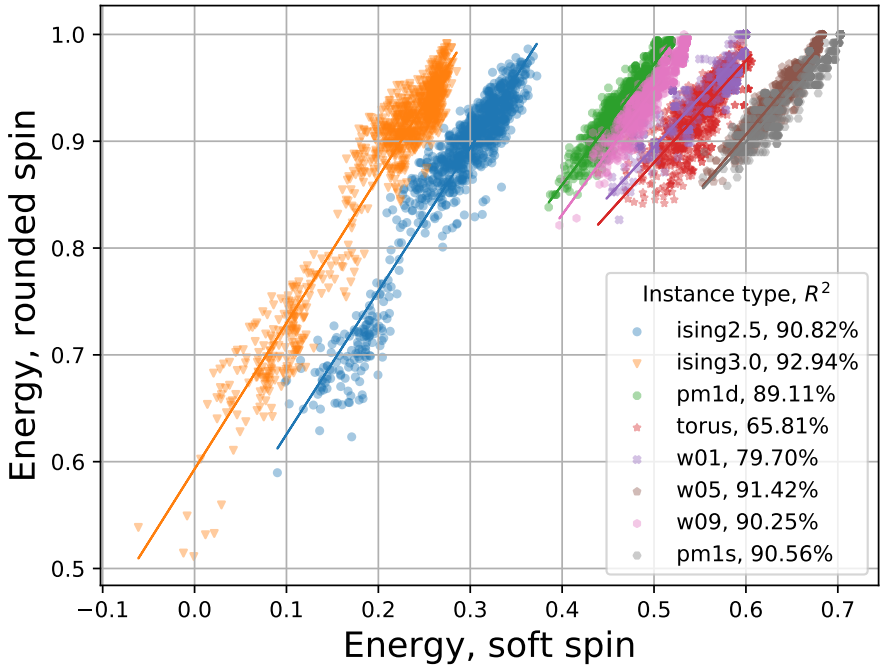


Figure 7.8: The objective function, normalized by the global optimum, evaluated at the soft spin (y axis) and the corresponding rounded spin obtained at the end of an LT run, for several independent runs of the algorithm on eight instances. Each instance is picked from a different instance type. The correlation between the two quantities (given by R^2 values in the legend) indicates that better steady state soft spin configurations tend to map to better feasible solutions.

Suppose, for a given run of the algorithm, the system tends to a particular steady state v^* at long times, with the state at some finite time t given by $\mathbf{v}_t = \mathbf{v}^* + \boldsymbol{\delta}_t$, where

$|\delta_{i,t}| \ll |v_i^*|$. Then, to first order in the displacement, we have

$$v_{i,t+1} = \tanh\left[\beta\left[(\mathbb{1} + cJ)\cdot(\mathbf{v^*} + \boldsymbol{\delta_t})\right]_i\right] \tag{7.23}$$

$$= \tanh\left[\beta\left(v_i^* + cF_i^*\right) + \beta\left[(\mathbb{1} + cJ)\cdot\boldsymbol{\delta}\right]_i\right] \tag{7.24}$$

$$\simeq \tanh\left[\beta\left(v_i^* + cF_i^*\right)\right] \tag{7.25}$$

$$+\beta\left[(\mathbb{1} + cJ)\cdot\boldsymbol{\delta}\right]_i \operatorname{sech}^2\left[\beta\left(v_i^* + cF_i^*\right)\right] \tag{7.26}$$

$$= v_i^* + \beta\left(1 - v_i^{*2}\right)\left[(\mathbb{1} + cJ)\cdot\boldsymbol{\delta}\right]_i \tag{7.27}$$

where we used the steady-state condition, and $\tanh'(x) = 1 - \tanh^2(x)$. Therefore, the displacement at time $t+1$ is

$$\boldsymbol{\delta}_{t+1} \simeq \beta\left(\mathbb{1} - V^{*2}\right)\cdot(\mathbb{1} + cJ)\cdot\boldsymbol{\delta}_t\,, \tag{7.28}$$

where we defined the diagonal matrix $V_{ii}^* = v_i^*$. Therefore, the norm of the displacement vector close to steady state is bounded as

$$|\boldsymbol{\delta}_{t+1}| \lesssim |\beta|\cdot||\left(\mathbb{1} - V^{*2}\right)||\cdot||\left(\mathbb{1} + cJ\right)||\cdot|\boldsymbol{\delta}_t|\,. \tag{7.29}$$

Since $||\left(\mathbb{1} - V^{*2}\right)|| \leq 1$, and $||\mathbb{1} + cJ|| \leq 1 + |c|\cdot||J||$, it follows that

$$|\boldsymbol{\delta}_{t+1}| \lesssim \beta\cdot(1 + c||J||)\cdot|\boldsymbol{\delta}_t|\,, \tag{7.30}$$

assuming $c, \beta \geq 0$. Finally, consider the following properties:

1. Since J has zero diagonal, then by the Gershgorin circle theorem, all eigenvalues of J lie within a disc of radius $\max_i \sum_{j=1}^{n} |J_{ij}|$.

2. If $c = \eta\bar{c}$, where $\bar{c} = \max_i \sum_{j=1}^{n} |J_{ij}|$, then $1 + c||J|| \leq 1 + \eta$.

Therefore, for a choice $\beta \gtrsim \frac{1}{1+\eta}$, we expect

$$|\boldsymbol{\delta}_{t+1}| \lesssim |\boldsymbol{\delta}_t| \tag{7.31}$$

giving the condition for dynamics converging to a steady state. Three observations can be drawn from this:

1. $\bar{c}$ provides a natural unit for the response c.

2. For optimal convergence, we expect the dependence between β and $\eta = c/\bar{c}$ to be given by $\beta \simeq a/(1 + b\eta)$ for some parameters a, b.

3. Under the above circumstances, the trajectory near steady state is stable and follows an exponential convergence towards the steady state solution. Therefore, the algorithm can be "safely" terminated when the displacement is under a certain threshold.

These three observations closely match our empirically derived rules for good performance of LT. This indicates that the steady state solutions may also correlate with the locations of good feasible solutions (given by the nearest hypercube vertex). This can be seen in Fig. 7.8.

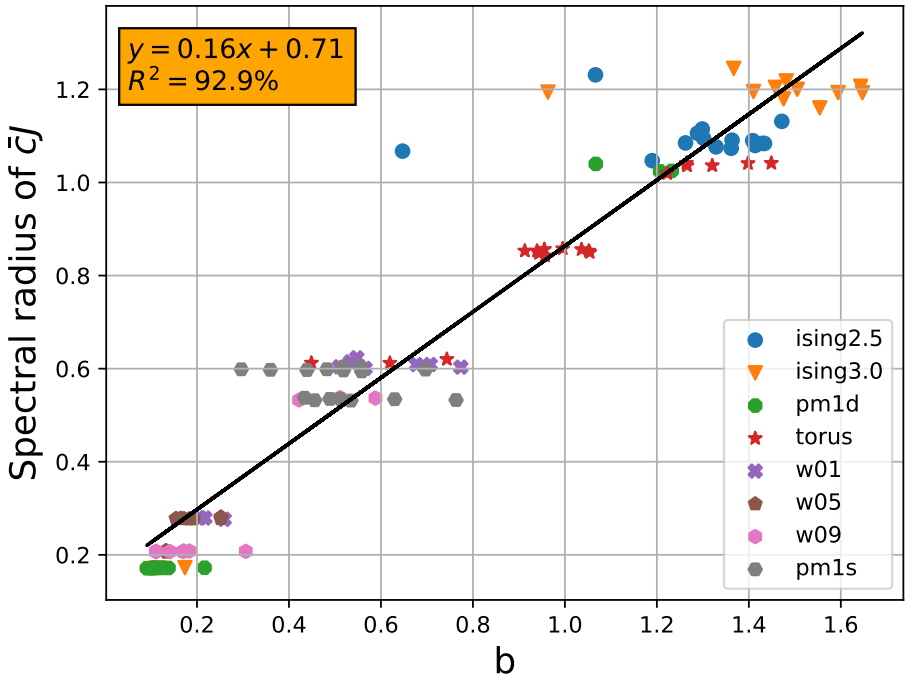


Figure 7.9: Dependence of the fitting parameter b and the spectral radius of the the normalized coupling matrix $\bar{c}J$ (units arbitrary). A linear regression fits the data with $R^2 = 92.9\%$, indicating a strong linear relationship between the two quantities. Therefore, the spectral norm of the coupling matrix can give a good estimate on the fitting parameter b and hence β.

7.6.3 Optimal parameters by instance type.

In fact, the quantity $\bar{c}$ is defined not as the maximum but as a mean, Eq. (7.21). However, the functional relationship $\beta = \frac{a}{1+b\eta}$ is still seen to hold for some a, b.

We can study the dependence of a, b by instance type. As shown in Fig. 7.4, the parameters form clusters by instance type, and the value of a is close to 1 for all instances studied. The value of b varies considerably from instance to instance. Looking at the functional form, it is reasonable to guess that b is related to the spectral radius of the coupling matrix J. While we bound the magnitude of the largest eigenvalue by $\max_i \sum_{j=1}^{n} |J_{ij}|$, the actual value may be smaller, and b may be understood to reflect this correction. We check this conjecture by plotting the relationship between $||J||$ and b for

every problem instance in Fig. 7.9.

This relationship can be particularly useful if the largest eigenvalue of the matrix can be calculated or estimated quickly. Then, by inverting the linear regression shown in Fig. 7.9, one obtains a good initial guess for b. The initial guess for a, on the other hand, is simply 1. This potentially reduces the hyperparameter optimization to local minimization in the single parameter η, which is computationally inexpensive.

Having given a physical description of the algorithm, we now turn to studying its performance on practical problem instances.

7.7 Comparison with Gurobi

Gurobi is a commercial optimization software that solves a broad range of problems including quadratic programming (QP), linear programming (LP), and mixed integer programming. Additionally, the software includes in-built heuristics to find good initial solution candidates quickly, as well as "pre-solve" subroutines and simplify the problem description by eliminating redundant variables or constraints.

In order to use Gurobi, a MAXCUT instance can be relaxed to either a linear program or a quadratic program. Mapped to an LP, the instance is specified by real variables $x_{ij} \in \mathbb{R}$ to each edge, with the inequality constraints $x_{ij} \leq 1$, where $x_{ij} = 1$ if and only if the edge ij is in the cut. Additional constraints follow by observing that not all configurations are feasible: for example, for three edges ij, jk, ki, at most two may be part of a cut. Any feasible solution must satisfy such *cycle* constraints as well, expressible as inequalities of the form $x_{ij}+x_{jk}+x_{ki} \leq 2$. Then, the LP is formulated as maximization

of the objective function $\mathbf{w}^T\mathbf{x}$, subject to the above inequalities, where $\mathbf{w}$ represents the vector of edge weights.

While the number of inequalities to fully characterize the feasible region are exponential in the input, smaller sets of constraints can suffice for good approximate solutions. The *separation problem*, which asks whether a candidate solution is in the polytope or, if not, to give a hyperplane separating it from the polytope, is shown to be polynomial-time in n for a particular choice of constraints [230]. This allows a polynomial-time, cutting-planes algorithm for finding and including violated constraints dynamically for each successive iteration of the LP until success. Theoretical results suggest, however, that solutions to the LP relaxation can be far from the optimal value in general, due to a large *integrality gap* (see, e.g, [231]). The formulation is somewhat unnatural for MAXCUT, and is also seen to perform poorly in practice due to a blowup in the number of variables.

A more natural formulation is as a QP with linear constraints, where every vertex i is assigned to one variable s_i, and the problem is expressed as

$$\max \ \frac{1}{4}\mathbf{s}^T \cdot W \cdot \mathbf{s}$$

$$\text{s.t.} \ -1 \leq s_i \leq 1.$$

Here, the matrix W is the weighted graph Laplacian, with $w_{ii} = \sum_{k \neq i} w_{ik}$ and $W_{ij} = -w_{ij}$ for $i \neq j$. The QP formulation is readily generalized to a semi-definite program by promoting the spin variables to vectors of unit length on a m-dimensional sphere. This forms the basis for the (optimal) Goemans-Williamson algorithm, [232], and is generally

the formulation of choice for exact solvers for MAXCUT. While Gurobi does not support an SDP formulation, the QP formulation is supported, with solutions via interior point methods (specifically, a parallel barrier method) and the simplex method. We therefore input our instances into Gurobi as QPs.

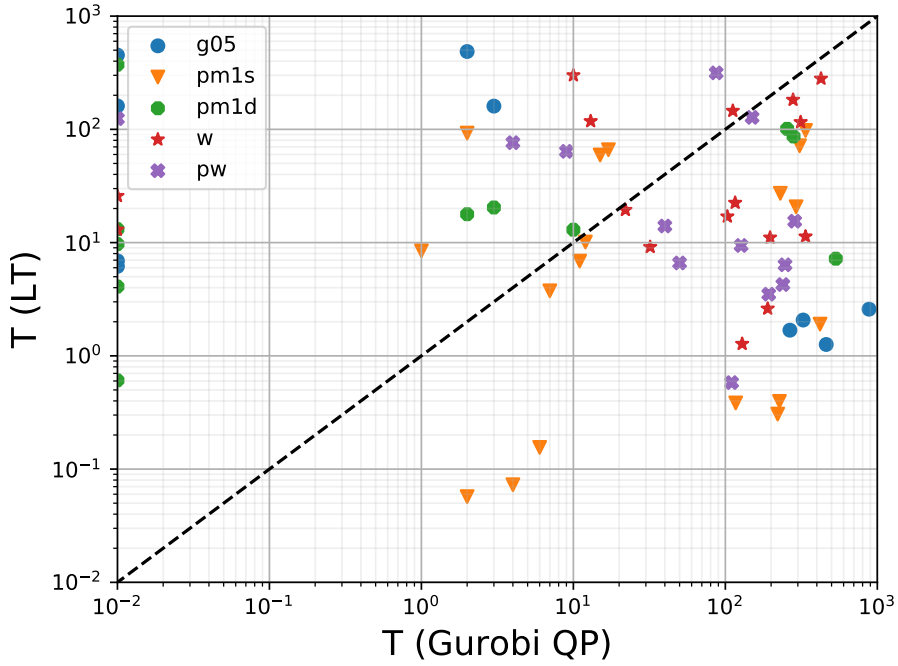


Figure 7.10: Performance of LT compared against Gurobi for several benchmarking instances. The performance metric used is median time (in seconds) taken to find the optimum (over 10 runs), with a timeout of 10^3 seconds. Timed out instances are not shown: Out of 130 instances, LT and Gurobi timed out on 47 and 22 instances, respectively, including 7 instances where both timed out. Times faster than a certain threshold are reported by Gurobi as 0s (corresponding to points along the left edge). LT and Gurobi find optima faster than each other in an equal number of instances, with no clear instance-dependent advantange. The speedup on either side is in some cases up to three orders of magnitude.

Then, we can compare the time to find optimal solution for Gurobi and LT on our benchmarking instances. The time has to be carefully defined in each case for a fair comparison. Gurobi is a deterministic algorithm, except for an initial (optional) heuristic step for proposing an initial solution candidate which takes a small fraction of the total runtime. The algorithm terminates when the optimum is found and proved. The latter

typically requires additional time to improve the upper bound on the optimum until it matches the best optimum found. Since we use benchmarking instances that have known optima, we define runtime leniently as the time to find (but not necessarily prove) the optimal solution.

On the other hand, LT is randomized due to the random choice of initial state, and multiple runs are necessary to gather statistics on the performance. Therefore, we define runtime as the median performance over 30 independent runs of LT for each instance. Furthermore, since LT requires parameter tuning, we allow up to 20 s of hyperparameter tuning by grid search in β, η that is not considered part of the runtime. Note that the results of Sec. 7.5 suggest that the parameters can be set automatically, either adaptively as for p or by a well-motivated formula for η, β, without the need for a full grid search.

Then, as shown in Fig. 7.10, the runtime performance of LT and Gurobi can be compared directly on every instance. We see that there is significant spread in performance for every problem type, for both LT and Gurobi. Promisingly, there are instances in every problem class for which LT is significantly faster than Gurobi.

The comparison with Gurobi illustrates that there may be cases where properly tuned LT can outperform state-of-the-art solvers at a fraction of the time cost. It is pertinent to ask whether the success of LT over other solvers can be predicted in advance, using instance data (or quantities derived from it). We briefly address this question.

The most obvious performance indicator is the number of variables n. The instances used in the time comparison with Gurobi were of size 60, 80, or 100. Another elementary indicator is clause density, or the mean number of clauses per variable, which for a

weighted instance is the average row sum of the graph adjacency matrix, $m := \sum_{i,j} J_{ij}/n$. We also compute the average row sum of the absolute value of the weight matrix $\bar{m} := \sum_{i,j} |J_{ij}|/n$. Finally, the misfit parameter μ measures the degree of frustration in the model. More precisely, it is the ratio of the ground state energy of the model to the ground state energy of a frustration-free reference system. For a given MAXCUT instance, a reference system with all weights J_{ij} replaced by their negative absolute values $-|J_{ij}|$ is frustration-free, with a ground state energy of $-\sum_{i<j} |J_{ij}|$. On the other hand, the ground state energy of the original instance is bounded below by $-\sum_{i<j} J_{ij}$. Therefore, we define misfit as

$$\mu := \frac{\sum_{i<j} J_{ij}}{\sum_{i<j} |J_{ij}|} \,. \tag{7.32}$$

Then, we ask: How well does a given performance indicator predict the runtime of LT (or Gurobi) on a randomly chosen instance? More formally, treating the runtime and indicator as random variables X, Y respectively, the predictive power can be expressed as the conditional entropy $H(Y|X)$, defined as

$$H(Y|X) := -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)} \,, \tag{7.33}$$

where the sum is taken over the support sets of X, Y. Informally, $H(Y|X)$ quantifies the number of additional bits needed to specify Y given knowledge of X. The largest possible value of $H(Y|X)$ is $\log |\mathcal{X}|$ for a discrete sample space $\mathcal{X}$, corresponding in our case to the number of bins used to group the runtimes. We report the conditional entropy normalized by this maximum, so that a normalized entropy of 0 (1) corresponds

Predictor	Gurobi	LT
n	0.73	0.69
m	0.68	0.63
$\bar{m}$	0.66	0.59
μ	0.56	0.53

Table 7.2: A tabulation of the normalized conditional entropy (as defined in Eq. (7.33)) of different performance predictors with the runtime of Gurobi and LT on the benchmarking instances. Zero indicates perfect prediction, while 1 corresponds to no predictability. The real-valued predictors $m, \bar{m}, \mu$ were binned into 20 equally spaced intervals, and the runtime was binned into 20 logarithmic intervals spanning the range 0.01s to 1000 s, with an additional bin for timed-out instances ($t > 1000$s).

to perfect (no) predictability. The results are presented in Table 7.2. Relative to Gurobi, the performance of LT is marginally more predictable using the instance data. However, clearly discernable relationships between the performance and any of the indicators studied here could not be obtained using the instance data available, suggesting the need for further systematic study.

7.8 Comparison with gradient descent

An inspection of the LT implementation reveals that the algorithm is operationally very similar to a gradient descent algorithm. The difference lies only in the fact that we apply a nonlinear $\tanh$ wrapper to each spin value in every step, while gradient descent is fully linear. This raises a natural question: does LT offer any advantage to gradient descent?

We formalize this comparison. The MAXCUT Hamiltonian does not have an extremum over $\mathbb{R}^n$, as all of its second (and higher-order) derivatives in any single variable vanish. Put differently, the Hessian of the cost function in the spin variables has zero on-diagonal entries, and is therefore trace zero. So, the Hessian is indefinite everywhere,

implying that no point can take an extremal value. This implies that a gradient descent algorithm must constrain the state vector to lie within a closed region of $\mathbb{R}^n$; then the optima are guaranteed to lie on the boundary of this region. The natural choice of region is the n-dimensional hypercube $H_n := [-1, 1]^{\times n}$, whose vertices correspond to feasible solutions to the MAXCUT problem. Then, any step that displaces the state vector outside H_n must be modified to obey the constraint. We implement this by applying a cutoff function to each spin at the end of every displacement step. The form of this function is as follows:

$$\mathrm{cutoff}(x) = \mathrm{sgn}(x) \cdot \min\{1, |x|\}. \tag{7.34}$$

When applied to each spin as $\mathrm{cutoff}(\beta v_i)$, this function has the effect of projecting every spin component that exceeds an allowed range $[-1/\beta, 1/\beta]$ onto the closest boundary of the range. The free parameter β controls how wide the allowed range should be.

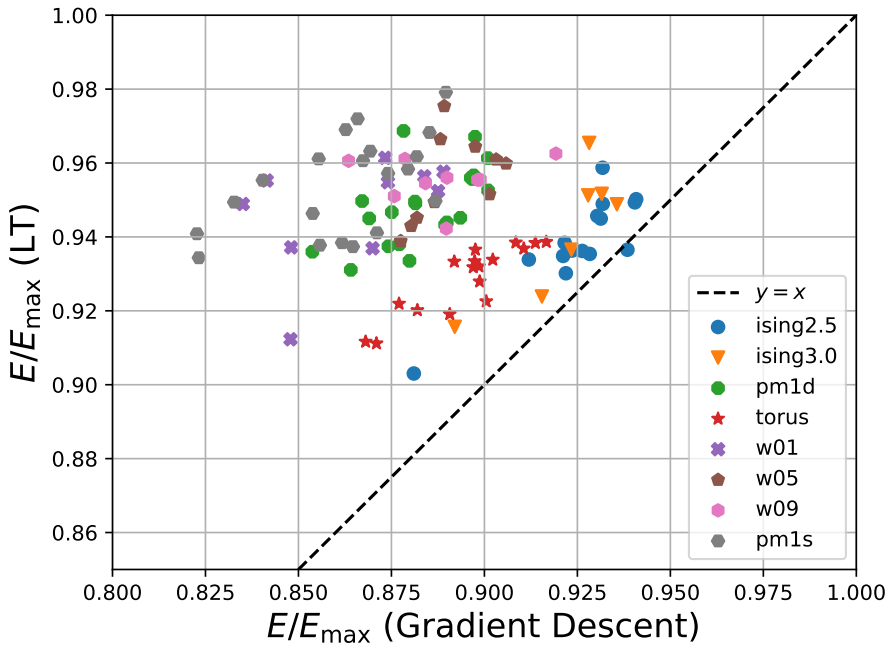


Figure 7.11: Performance of LT and gradient descent, given by the energy obtained as a fraction of the maximum (with 1 being optimal), across different benchmarking instances. For each instance shown, the performance has been averaged over 100 trials after a pre-optimization of the hyperparamaters.

The full algorithm may then be written down:

1. Initialize all spins uniformly at random, $v_i \in [-1, 1]$.

2. Apply displacement to spin $v_i \mapsto v_i + c \cdot F_i$ where $F_i = \partial H / \partial v_i$.

3. $v_i \mapsto \text{cutoff}(\beta v_i)$.

4. After p rounds, round each spin to its sign, ± 1.

It is now apparent that GD mirrors LT, with the difference lying in the choice of onsite activation function used: LT uses the tanh function while GD uses a hard cutoff function. Both algorithms have identical free parameters p, c, β that play the same or similar functional roles in each case. Then, we can compare the performances of these

algorithms on the same instances. In Fig. 7.11, we see that LT beats GD on average for the instances studied. This suggests that the specific form of LT that uses a tanh function offers an advantage over a hard cutoff function. This choice also corresponds to the underlying physics described in Sec. 7.4.

7.9 Discussion

The benchmarking of our implementation of LT on the MAXCUT instances gives evidence that LT can perform well in certain practical problem settings. We find that the LT hyperparameters can be set using simple rules that obviate the need for a full, global hyperoptimization, making the algorithm particularly lightweight.

It remains to be seen how well LT fares on problems other than MAXCUT. We expect LT to show similar performance in closely related quadratic unconstrained binary (QUBO) problems. More generally, we observe that the algorithm itself is specified by a domain relaxation, and a notion of derivative of the objective function with respect to each variable. These are minimal requirements found in many optimization problems, for example mixed integer linear programs. Therefore, an interesting open question is whether LT can be adapted for use in these settings as well.

The analysis in Sec. 7.4 suggests an alternative description of the algorithm as a discretized simulation of imaginary-time dynamics in a spin system. It is interesting whether this picture can be pursued to design improvements or variations to the algorithm, or generalize it to other settings, for instance, on problems like quantum SAT where the problem Hamiltonian is not diagonalizable in any local basis. It would also be interesting

to compare LT to other heuristic algorithms that mimic low-energy physical dynamics such as simulated annealing, simulated quantum annealing, or substochastic Monte Carlo.

Chapter 8: Conclusion

In this thesis, we have studied several aspects of near-term quantum (or NISQ) computation. For a summary of the thesis, we refer the reader to Chapter 1. Instead, here we comment on the larger context surrounding our work, and end with some speculation about the future.

8.1 Open problems

The first part of the thesis studies problems related to the connectivity between qubits in quantum architectures. Our analysis simplifies the qubits as nodes in a graph, and connects two qubits with an edge if they can be made to interact via two-qubit gates (or Hamiltonians). We consider the problem of qubit routing on graphs of limited connectivity, and compare different architectures on the basis of structural and functional performance metrics. There are many open directions here for future work, such as the development of routing algorithms in the presence of defective qubits (which is ongoing), and the modeling of noise in the device via, e.g., weighted graphs. But aside from these natural generalizations, it is interesting and important to tackle the larger problem of circuit compilation on quantum architectures. Loosely, one can state this problem in the following manner: Given a high-level description of a quantum algorithm, what is

the optimal way to execute it on the quantum hardware? It is not hard to see that this question involves many sub-problems. What is the definition of 'optimal'? What is the optimal gate description of the algorithm? How to compile these gates into an executable operation sequence? What should the underlying connectivity look like? Routing is simply one part of this stack that allows distant gates to be executed on the device. Therefore, it would be interesting to expand the scope of the problem 'vertically' in the stack.

There is a rich array of open problems in variational algorithms as well. There is considerable interest in these algorithms due to the abundance of hard optimization problems in several domains such as quantum chemistry, constraint satisfaction, and, to some extent, quantum simulation of quantum field theories [233]. Variational algorithms are also interesting because of how well-suited they seem to be for NISQ implementation. But behind the elegant simplicity of variational algorithms lies a rich yet poorly understood phenomenology. This is due in part to the fact that variational algorithms are really algorithm *frameworks* that can express a multitude of quantum dynamics depending on the values of parameters used, and to get the optimal circuit for a given problem one has to first perform an optimization over the parameter space. Therefore, in order to understand variational algorithms, it is necessary to understand the parameter landscapes of variational algorithms.

In this regard, we know of several guiding principles: the adiabatic theorem for QAO [26], the fact that Trotterized QAO is a special case of QAOA [28], or that bang-bang control is known to be sufficient for optimality on non-singular, linear control problems [23]. But these results alone do not provide a complete description of the

quantum dynamics, and in fact, they are often inapplicable in practice. For example, QAOA rarely approximates Trotterized QAO, because it is primarily designed to be a low-depth, parameterized circuit. The bang-bang optimality guarantee breaks down due to the ubiquitousness of singular time periods in the optimal control schedule [31]. And running QAO at a rate inversely proportional to the square of the smallest spectral gap is sometimes too conservative a strategy, as the adiabatic unstructured search protocol of Roland and Cerf demonstrates [178]. The theory of variational algorithms is incomplete, and there are (at least) two ways to build on it. The first is an empirical approach that involves designing heuristic variational algorithms and discovering properties about the cost landscape and its optima by trial and error. The second approach works from first principles to prove guarantees on variational algorithms of a possibly idealized form. These two approaches inform one another. Indeed, our empirical results on asymptotic QAOA parameter curves in Chapter 6 have led to a subsequent first-principles approach to understand the connections between quantum annealing, QAOA, and the underlying optimal control theory [31, 234].

8.2 Perspectives about the future

The original motivations behind quantum computation run very deep: to represent nature in its entirety, and, conversely, to harness the full representational power of nature to computational ends. There is something profound about this idea that makes quantum computing worth studying regardless of its fate. Having said that, I believe that large-scale quantum computers will eventually be realized and they will be useful (and maybe even

indispensible). Based on forecasts on the progress of the field, it could be a few decades (if not more) before universal, fault-tolerant quantum computers become widely available commercially [235]. Over that period, quantum computers will not only grow in scale, but likely also change in nature with the advent of transformative technologies. Quantum computing in its mature form may be unrecognizable today, much like the classical computation of yesteryear looks very different than it does today. With this growth, the way quantum computing is represented (and understood) will change, becoming more removed from the hardware layer. Perhaps this shift towards abstraction will bring about more efficient ways to conceptualize quantum dynamics, and it is difficult to truly fathom how different quantum science could look in that new paradigm.

Appendix A: Appendices to Chapter 6

A.1 Quantum Approximate Optimization Algorithm (QAOA)

The QAOA is an approximate optimization algorithm first introduced in 2014 by Farhi *et al.* [28], and has since enjoyed growing interest. The QAOA uses alternating evolutions under two non-commuting operators, typically a problem (or cost) Hamiltonian H_A that encodes the cost function on the diagonal in (say) the σ^x basis, and a transverse term $H_B = -\sum_{i=0}^{N} \sigma_i^y$ that generates transitions between bit strings, such that the initial state $|+\rangle_y^{\otimes N}$ evolves into an approximate ground state of H_A.

Practically, the most valuable feature of the QAOA seems to be its "learnability" via a classical outer loop optimizer, where the discovery of the evolution angles in the optimal QAOA schedule is achieved via the discovery of structure in the angle sequences [194,197]. These patterns are seen quite generally across local Hamiltonian problems, and while steps towards a theory describing optimal QAOA sequences have been taken [31], several questions surrounding it remain open. Regardless, the structure in optimal QAOA schedules may be harnessed to implement approximate state preparation in a scalable manner and with a low overhead on quantum resources. We present a new heuristic method that helps achieves this goal.

First, we discuss how to discover optimal QAOA1 schedules, i.e., QAOA schedules

for $p = 1$.

A.1.1 QAOA, $p = 1$

Despite its apparent simplicity, the $p = 1$ QAOA (or QAOA1) can be a powerful state preparation ansatz. For example, hardness-of-sampling results are known for QAOA1 circuits [191], closely mirroring the hardness of sampling from instantaneous quantum polynomial (IQP) circuits (see next section for details). Furthermore, it is known that the performance of the QAOA1 for certain combinatorial optimization problems can be competitive with the best classical algorithms for the same problems [236]. Another desirable feature of the QAOA1 for local spin Hamiltonians is the tractability of computing energy expectation values, as observed in [28]. A very similar result has also been known in the setting of quantum dynamics [205,206]. For a two-local transverse field spin Hamiltonian as in Eq. (1) in the main text, this leads to a formula for the energy expectation under a state produced by the QAOA1, starting from the product state $|+\rangle^{\otimes N}$. These formulas are applicable to many cases of interest in quantum state preparation and optimization. Importantly, the time complexity to compute the formula is $O(N^3)$ in the worst case, making it tractable to optimize the QAOA1 protocols for large spin chains.

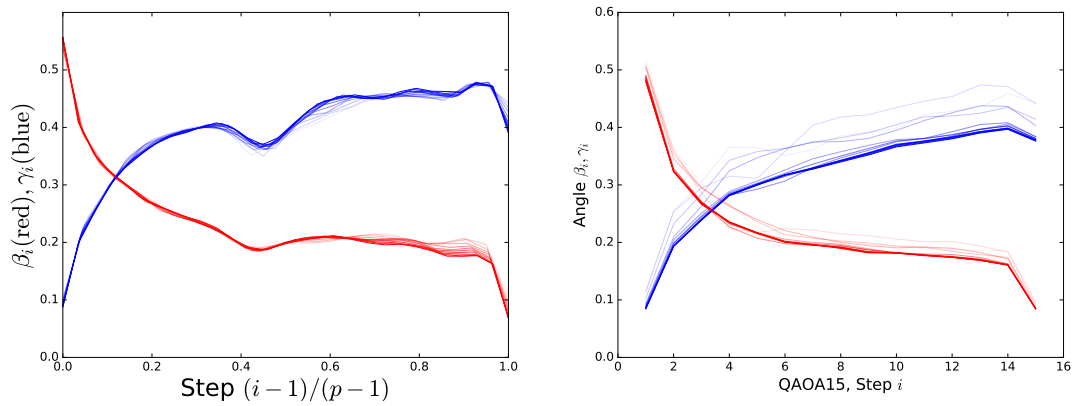


Figure A.1: Convergence in p and N. Convergence of optimal angle curves with increasing QAOA layers p (left), and number of spins N (right). The p-convergence plot was generated for an $N = 8$ spin system, for p ranging from 20 up to 30, with higher p shaded darker. The N-convergence figure was generated for a 15 layer QAOA, for N in the range of 4 to 14, with higher N curves shaded darker.

A.1.2 QAOA, $p > 1$

The general analytical formula for $p = 1$ does not extend to the case where we apply the QAOA for more than one layer. Here, we must turn to classical numerical methods to find the optimal QAOA angles β_i, γ_i for each layer i. For p layers, this is an optimization on a $2p$-dimensional space that grows exponentially with the depth of the circuit. However, numerics done here and in [194, 197] have identified the existence of minima that exhibit patterns in the optimal QAOA angles, namely that the angles, when plotted as a function of their index i, form smooth curves for any p. While this observation points to a deeper theoretical mechanism at play, it does not directly simplify the optimization problem, since we must still search over all approximately smooth sequences of the angles. Zhou *et al.* [194] have exploited the smoothness of the functions by carrying out searches in the Fourier domain. Here, we follow a different route that arises from some novel observations of these family of minima.

For each p, denote the special optimal angles by $\left\{\left(\boldsymbol{\beta}^{*(p)}, \boldsymbol{\gamma}^{*(p)}\right)\right\}_p$, which we can also think of as a pair of angle curves (as a function of step index i). As p is varied, we may think of these minima as a family. We numerically find that this family exhibits the following desirable features (for p sufficiently large):

1. The angles are non-negative, small and bounded.

2. For p sufficiently large, the two angle sequences $\beta^{*(p)}$ and $\gamma^{*(p)}$ are approximately smooth.

3. The angle sequence $\beta^{*(p)}$ (and correspondingly, $\gamma^{*(p)}$) when viewed as a function on the normalized time parameter $s_i = \frac{i-1}{p-1}$, is convergent in the parameter p. In other words, as p is increased, the angle sequences $\beta^{*(p)}$ and $\gamma^{*(p)}$ approach a smooth, asymptotic curve (See Fig. A.1.)

4. The energy expectation $E(\beta^{*(p)}, \gamma^{*(p)})$ approaches the global minimum as $p \to \infty$, and hence this family is asymptotically optimal.

The significance of the first point is that in experimental settings, large evolution times are infeasible to implement due to decoherence, so these minima correspond to practicable QAOA protocols. The third and fourth points suggest an inductive algorithm where a locally optimal schedule for a given p may be discovered using the optimal schedule for $p-1$ as a prior.

Point 3 in the above list is a novel observation that allows us to construct a heuristic that is efficiently scalable for large p. The main idea behind this construction is that the minimal angle curves for a larger p may be guessed from the optimal curve of a smaller

$p' < p$ by interpolation. It should be noted that the angles in these curves remain roughly the same size as p increases. Furthermore this size is large enough that interpretation of QAOA as a Trotterized product series is not feasible with the corresponding error terms being non-vanishing. Therefore, while tempting, it is not theoretically accurate to interpret these curves as a Trotterized annealing path. The theoretical underpinnings of these curves are still under investigation.

Using the above points, we use a bootstrapping algorithm to find the optimal angle sequences, $\beta^{*(p)}$ and $\gamma^{*(p)}$, for a given p, as described below. Let $q = 1, \ldots, p$ denote an intermediate angle index. Then:

1. For $q = 1$, use an analytic formula to find $\beta^{*(1)}$ and $\gamma^{*(1)}$.
2. For $q = 2$, choose an initial guess of $\beta^{(2)} = \left(\beta^{*(1)}, \beta^{*(1)} - 0.2\right)$ and $\gamma^{(2)} = \left(\gamma^{*(1)}, \gamma^{*(1)} + 0.2\right)$.
3. Perform a local optimization of $\beta^{(2)}$ and $\gamma^{(2)}$ in order to find $\beta^{*(2)}$ and $\gamma^{*(2)}$.
4. Repeat the next steps (5-7) for $q = 3, \ldots, p$.
5. Create interpolating functions through the angle sequences, $\beta^{*(q-1)}$ and $\gamma^{*(q-1)}$, using the normalized time $s_i = \frac{i-1}{q-2}$ as the independent parameter (we use a linear interpolation for $q = 3$ and cubic for $q > 3$).
6. Choose the initial guesses for $\beta^{(q)}$ and $\gamma^{(q)}$ by sampling the interpolating function from (5) at evenly spaced points separated by a normalized time distance of $\Delta s = 1/(q-1)$.
7. Perform a local optimization of $\beta^{(q)}$ and $\gamma^{(q)}$ in order to find $\beta^{*(q)}$ and $\gamma^{*(q)}$.

The resulting angles $\beta^{*(p)}$ and $\gamma^{*(p)}$ should be at least a good local minimum of the energy expectation value and approaches the global minimum as $p \to \infty$.

The $q = 2$ interpolation in step 2 is based on our observation that the β angles tend to curve down at the end and the γ angles tend to curve up.

An important feature of our algorithm is that its asymptotic runtime is expected to be efficient in p. This feature is predicated on the previous result that the angle curves are generally convergent as p tends to infinity. The argument proceeds as follows: if we assume a maximal deviation of the initial guess for layer q to be $\epsilon_q \geq 0$, then the total l_2-norm distance between the initial guess and the optimized curve is no greater than $\epsilon_q\sqrt{q}$, by the Cauchy-Schwarz inequality. Therefore, the local search algorithm is confined to a ball of radius at most $\epsilon_q\sqrt{q}$, and for a fixed error tolerance, the convergence time for a standard local optimizer is $O(\epsilon_q^2 q)$. Summing over convergence times for all from $q = 1, \ldots, p$, we have

$$T = O\left(\sum_{q=1}^{p} q\epsilon_q^2\right) \leq O(p^2) \tag{A.1}$$

The last inequality above comes about as follows: while the summand depends on the convergence rate of the sequence $\{\epsilon_q\}_{q=1}^{p}$, it is upper bounded by $O(q)$ for a converging set of paths and an initial error ϵ_1 of order 1. The latter is true since our angle search domain is bounded and independent of N. Therefore, the sum is no greater than $O(p^2)$. In practice, even faster runtimes are possible. Therefore, the bootstrap algorithm exploits the structure of the special minima and provides a scalable route to multi-step QAOA for the long-range TFIM. In fact, as discussed in the supplement and in [31],

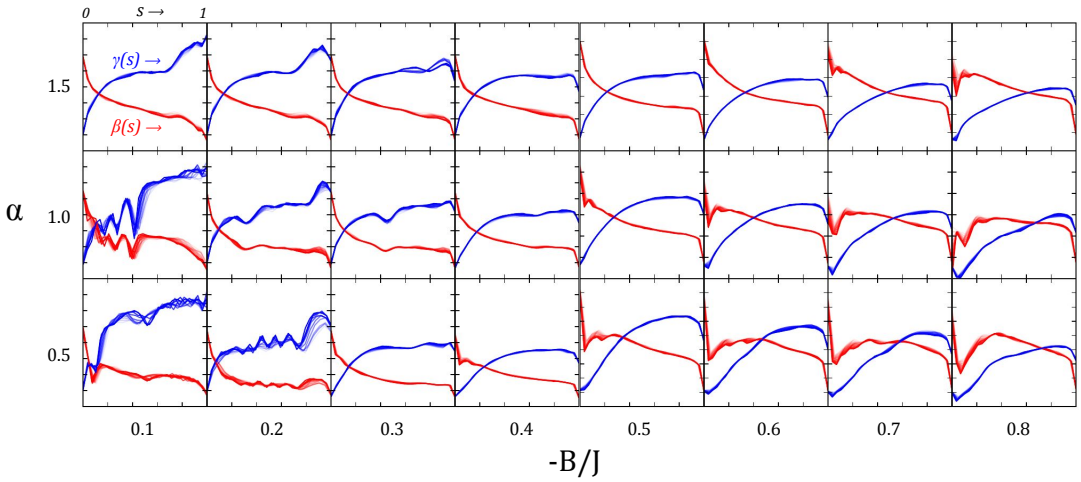


Figure A.2: Angle sequence curves. A collage of angle sequence curves, arranged by the Hamiltonian parameters for which they were computed. In each subplot, curves for different p ranging from 20 to 30 are overlaid, with higher p curves shaded darker. The horizontal axis represents fractional step $s = (i-1)/(p-1)$ ranging from 0 to 1, while the vertical axis gives the value of the angles β (red), and γ (blue) in the range $[0, 0.6]$. The subplots are arranged horizontally by $-B/J_0$, increasing from 0.1 to 0.8 in steps of 0.1 (from left to right), and vertically by the long-range power $\alpha = 0.5, 1.0, 1.5$ (bottom to top). This collage shows the persistence of structure in the optimal angle sequences for a range of Hamiltonians within the same family.

there is mounting numerical evidence that the path approach applies across a very general variety of models on discrete as well as continuous systems.

A.1.3 Convergence in N

In the previous sections, we introduced a bootstrap algorithm that is asymptotically efficient in the number of layers p. However, in order to be fully scalable the algorithm must also be scalable in the system size N. This may not be possible in general (say for random spin models), as the optimized angles for a particular small system may have no bearing on the angles for a larger system. However, for the long-range TFIM, and indeed any translationally-invariant model with a well-defined notion of metric and dimension arising from the functional form of the coupling coefficients J_{ij}, it is reasonable to expect

that the optimized angles depend on system size in a predictable way. This is indeed the case for the long-range TFIM. There, it can be seen that the angle curves for varying N appear similar in shape. Usefully, the curves also appear to be *convergent* to an idealized curve for a hypothetical continuous, long-range spin chain. Once again, this feature suggests that the optimized QAOA angle curves for small systems may be used as initial guesses for larger systems *within the same Hamiltonian family*.

While it is not clear (due to numerical limitations) how fast the curves converge, we argue that the rate should be weakly dependent (or independent) of the system size N. For a given coupling function (such as inverse power-law) that decays as a function of distance, we define a characteristic length scale, which may be called the *skin depth* δ, that is the number of sites from the boundary that the coupling is a factor of e smaller than the nearest-neighbour value. In other words, we define δ such that $J_{i,i+\delta} \sim J_{i,i+1}/e$. Clearly, δ is independent of the system size N and depends only on the parameters of the coupling function. For instance, for the long-range TFIM, $\delta \sim e^{1/\alpha}$. As N tends to infinity, the *fractional* skin depth δ/N then "falls away" and becomes vanishing with respect to the bulk region of the chain. Now, we make the assumption that any deviations in the optimal QAOA schedules from N to $N+1$ arise from change in the fractional skin depth, which is reasonable for a translationally invariant model. The incremental change in the fractional skin depth from N to $N+1$ is $\delta/N - \delta/(N+1) \sim O(1/N^2)$. Therefore, if the change in the optimal QAOA curves ϵ_N (in, say, l_1-norm distance) is a smooth function of the the fractional skin depth, then we expect it to vary as $\epsilon_N \sim 1/\text{poly}(N)$. Therefore, the total running time of a bootstrap from small system sizes to a given size N

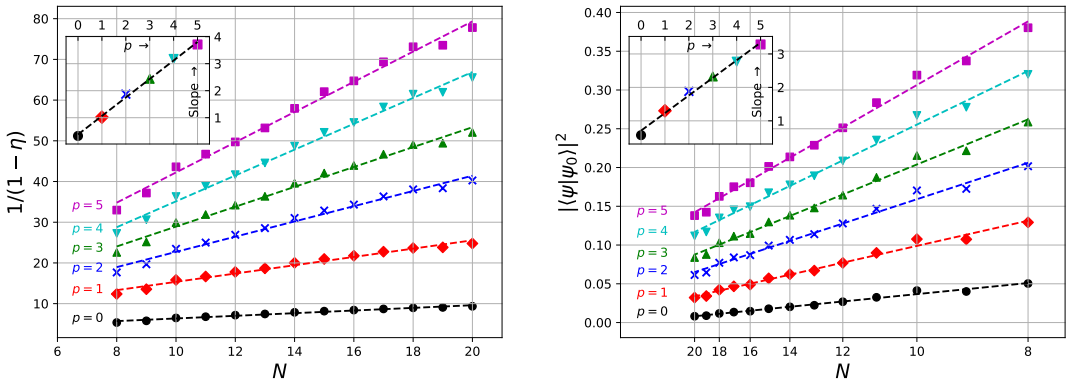


Figure A.3: Performance scaling in p, N. Behaviour of performance parameters η (left) and squared ground state overlap (right) with increasing number of spins N (x axis) and p (colors), for ideal power-law coupling with $\alpha = 1.1$. We find that for each p, $1/(1-\eta)$ grows linearly in N with a slope that depends on p. (Inset) The slope is linear in p, suggesting that the performance converges to 1 as $\eta \sim 1 - 1/(pN)$. On the right, we empirically observe that $|\langle\psi|\psi_0\rangle|^2 \sim p/N$, indicating that constant overlap with the ground state can be achieved with linear depth QAOA. The x axis has been scaled as $1/N$ so that the linear relationship with the squared overlap is apparent. The inset shows the linear trend with p.

should be $O\left(\sum_{k=1}^{N} 1/\text{poly}(N)\right)$ which is sub-linear in N. Combining this observation with the convergence in p, we see that for a given Hamiltonian family, optimized QAOA angle curves for small p may be used as a rubric for the optimization for longer circuit depths. Furthermore, if the Hamiltonian is translationally-invariant with decaying interactions, the optimized QAOA schedules are expected to scale with N as well. Therefore, the state preparation procedure under the QAOA for such a Hamiltonian family is scalable in circuit "volume", for a wide range of Hamiltonian parameters (Fig. A.2). This is our main theoretical contribution in this work.

A.1.4 Scaling of η in p, N

Our performance parameter η, defined as

$$\eta \equiv \frac{E(\boldsymbol{\beta}, \boldsymbol{\gamma}) - E_{max}}{E_{gs} - E_{max}}, \tag{A.2}$$

measures how close (in energy) the prepared state is to the ground state of the system. As described in previous sections, the optimal angle curves for QAOA appear to converge to a smooth, hypothetical curve, as a function of p as well as N. We show that under the assumption that such a curve exists, there is a fast heuristic for finding optimal angles for any finite p that is time-efficient in p and the number of spins N (when used in conjunction with the quantum device). In this section, we show that not only is the search efficient, but the quality of the optimum is numerically seen to improve with p, N as well.

In Fig. A.3, we show the result of the numerical study. We chose as the target Hamiltonian an idealized transverse field Ising model with inverse power-law couplings, with the power $\alpha = 1.1$ chosen to closely mimic the experimental Hamiltonian. The number of spins was varied from $N = 8$ to 20. Via DMRG, the critical value of the transverse field for a finite chain can be located by maximizing the von Neumann entropy at half-cut. This was done independently for each value of N. Then, using our heuristic, we located the optimal angle curve, and computed η for the final state prepared using this angle sequence, for each N. The plot shows the trend of $1/(1-\eta)$ with N, for a range of $p = 0, 1, 2, 3, 4, 5$, with 0 corresponding to a trivial protocol where the initial state is returned. While the number of spins could not be extended beyond 20 due to computational limitations, the trend is clear. We see that $1/(1-\eta)$ grows linearly with N and p (inset). While the linear trend in N is encouraging, we similarly expect the inverse spectral gap (and indeed, the density of low-lying states) to increase with N. Empirically for the target Hamiltonian, we observe a gap scaling of $\sim 1/N^2$. Assuming the density of low-lying states scales similarly, this suggests that the squared overlap with the ground state should fall off with N. Numerics confirm this expectation and indicate a scaling of

the squared overlap of $|\langle\psi|\psi_0\rangle|^2 \sim p/N$.

The linear scaling with p for both the energy and fidelity metric, combined with a polynomial-time search heuristic, suggests that for any desired energy (or probability) threshold ϵ, our approach allows us to approximate the state to within $1-\epsilon$ (in energy or fidelity) in time and number of layers that scale as poly$(N, 1/\epsilon)$.

A.1.5 Characteristic scale for η

The figure of merit η characterizes how close the final state is to the ground state of the system. At $\eta = 0$, the system is in the highest excited configuration, while $\eta = 1$ corresponds to a perfectly prepared ground state. QAOA, starting from the initial state $|+\rangle^{\otimes n}$, gives a state with figure of merit $\eta \in [0, 1]$, from the initial value of η_0. The difference between the final η and η_0 indicate the success of our QAOA protocol.

While η is normalized to the range $[0, 1]$, differences in η are still somewhat arbitrary. In long-range Ising models with a transverse field η_0 is not 0 but typically greater than 0.5, making the difference in η an unsatisfactory metric of success. Therefore, in addition to the initial and final η, we must provide a characteristic scale for η that quantifies the typical deviation from η_0. A natural choice is the standard deviation of η for QAOA with random angles.

For QAOA1 with evolution angles β, γ, it is possible to estimate the standard deviation analytically as a function of the underlying model parameters B and J_0 and on the number of qubits N. This derives from the analytical formula for the energy

expectation $E(\beta, \gamma)$ which can be stated as follows:

$$E(\beta, \gamma) = E_I + E_{II} + E_{III} \tag{A.3}$$

where

$$E_I = B \sum_{i=1}^{N} \prod_{k \neq i} \cos(2\gamma J_{ik}) \tag{A.4}$$

$$E_{II} = -\frac{\sin(4\beta)}{2} \sum_{i,j} J_{ij} \sin(2\gamma J_{ij}) \prod_{k \neq i,j} \cos(2\gamma J_{ik}) \tag{A.5}$$

$$E_{III} = -\frac{\sin^2(2\beta)}{4} \sum_{s=\pm 1, i, j} J_{ij} \prod_{k \neq i,j} \cos(2\gamma (J_{ik} + (-1)^s J_{jk})) \tag{A.6}$$

where the Hamiltonian has long-range power law couplings $J_{ij} \sim \frac{1}{|i-j|^\alpha}$ (with $J_{ii} = 0$), and a transverse field of strength B. Then, our goal is to compute the standard deviation (normalized by the spectral bandwidth $\Delta := E_{max} - E_{gs}$),

$$\frac{\sigma_E}{\Delta} = \frac{\sqrt{\langle E^2 \rangle_{\beta,\gamma} - \langle E \rangle_{\beta,\gamma}^2}}{\Delta} \tag{A.7}$$

which gives us the characteristic scale for η. We define the average $\langle \cdot \rangle_{\beta,\gamma}$ as

$$\langle f \rangle_{\beta,\gamma} := \lim_{T_\beta, T_\gamma \to \infty} \frac{1}{4 T_\beta T_\gamma} \int_{-T_\beta}^{T_\beta} \int_{-T_\gamma}^{T_\gamma} f(\beta, \gamma) d\beta d\gamma \tag{A.8}$$

In the limit, the average is precisely the constant term of the Fourier transform of f. Since the function is a sum of trigonometric monomials, its moments over the angle variables β, γ can be computed analytically term by term. We will need the following properties of

the coupling function:

1. (Symmetry) Since the inverse power law only depends on distance between nodes, we have $J_{ij} = J_{(2j-i)j}$ In other words, the inverse power-law is symmetric under a lateral flip (or "mirroring"). We assume a finite, open chain, and therefore couplings J_{ij} with $|j-i| > N-j$ do not have an image under mirroring.

2. (Incommensurateness) The coupling strengths J_{ij} are, in general, mutually indivisible irrational numbers whose sums and differences are also irrational and mutually distinct, e.g. for $i \neq j, k \neq l$, $J_{ik} \pm J_{jk} \neq J_{il} \pm J_{jl}$ (with a very small set of exceptions due to, say, symmetry).

The mean $\langle E \rangle_{\beta,\gamma}$ consists of three parts corresponding to the terms E_I, E_{II}, E_{III}. Performing the β integral first, we see that $\langle E_{II} \rangle_{\beta,\gamma} = 0$. Next, we may argue that in products of the form $\prod_k \cos(2\gamma J_{ik})$, the cosine factors are of degree one if they have no mirror images, and degree two otherwise. The only way to have a non-zero expectation is if all terms are systematically paired up by mirroring, so that the overall product is quadratic in a product of cosines. For the summand in E_I, this can only happen if N is odd and i is exactly at the center of the chain, in which case the average is $B/2^{(N-1)/2}$. When N is even, the mean is 0. Finally, for general i, j the last term is zero by property 2, since the cosines are generically incommensurate and therefore barring very few exceptions, most phases do not cancel out. However, in the special case that i, j are mirror images, i.e. $i = N - j$, we have perfectly paired terms when N is even (and one unpaired term at $k = \lfloor N/2 \rfloor$ when N is odd). Counting all occurrences of this case, the mean is approximately $\frac{1}{2^{N/2+1}} \sum_{i=1}^{N} J_{i(N-i)} \lesssim N J_0 / 2^{N/2}$ where J_0 is the nearest-neighbor coupling

in the chain. Note that asymptotically in N, $\langle E\rangle_{\beta,\gamma} \sim O(N/2^{N/2})$ which approaches 0 in the infinite N limit.

Next, we estimate the term $\langle E\rangle^2_{\beta,\gamma}$. By the orthogonality of trigonometric polynomials in β, we first have that $\langle E\rangle^2_{\beta,\gamma} = \langle E_I\rangle^2_{\beta,\gamma} + \langle E_{II}\rangle^2_{\beta,\gamma} + \langle E_{III}\rangle^2_{\beta,\gamma}$. Therefore, we estimate each term separately. As before, we require that the cosines pair up so that their phases can cancel. First, we have

$$\langle E_I\rangle^2_{\beta,\gamma} = B^2 \sum_{i,j} \prod_{k=1}^{N} \cos(2\gamma J_{ik}) \cos(2\gamma J_{jk}) \tag{A.9}$$

Each summand is a product of $2N$ cosines, and only survives averaging if every cosine is paired. This happens exactly when either $i = j$ or $i = N - j$ (There is also the "disconnected" contribution that cancels with the mean). In each case, the squared cosines give a factor of $1/2$ from averaging. Moreover, using mirror symmetry we can have fourth powers of some of the cosines, which give a factor $3/8$ from averaging. In all, the mean (minus the disconnected part) is no greater than

$$\langle E_I\rangle^2_{\beta,\gamma} \lesssim 4NB^2 \left(\frac{3}{8}\right)^{(N-1)/2} \tag{A.10}$$

A similar reasoning for E_{II}, E_{III} give us the following estimates:

$$\langle E_{II}\rangle^2_{\beta,\gamma} \lesssim \frac{1}{4} N J_0^2 \left(\frac{3}{8}\right)^{(N-1)/2} \tag{A.11}$$

$$\langle E_{III}\rangle^2_{\beta,\gamma} \lesssim \frac{3}{16} N J_0^2 \left(\frac{3}{8}\right)^{(N-1)/2} \tag{A.12}$$

Finally, this gives

$$\langle E\rangle^2_{\beta,\gamma} \lesssim N\left(\frac{3}{8}\right)^{N/2}\left[8B^2+J_0^2\right] \sim O(N\cdot(3/8)^{N/2}) \tag{A.13}$$

Therefore, we see that the standard deviation $\sigma_\eta = \sigma_E/\Delta \sim \frac{\sqrt{8B^2+J_0^2}}{\Delta}\cdot N^{1/4}\,(3/8)^{N/4}$, which is exponentially suppressed for large N. For $N = 20$ ions, we have $N^{1/4}\cdot(3/8)^{N/4} \sim 0.02$. While this is already small, the normalization $\frac{\sqrt{8B^2+J_0^2}}{\Delta}$ will have an additional linear N factor in the denominator, making the scale for η about 0.002. Therefore, a typical final QAOA performance of $\eta \gtrsim 0.95$ is several standard deviations above a typical $\eta_0 \sim 0.85$.

A.2 Evidence for hardness of sampling from general QAOA circuits

In this section we expand upon previous work [28] that gives evidence for exact sampling hardness of QAOA circuits, using the techniques of Refs. [208, 209] to give evidence for hardness of approximate sampling. First we relabel the bases $Y \to X \to Z$ so that the $p = 1$ experiment is equivalent to preparing a state $|\psi_0\rangle = \uparrow\rangle_x^{\otimes N}$, evolving under a Hamiltonian H_z diagonal in the computational basis, followed by a uniform rotation $\tilde{H} = e^{-i\beta\sum_i \sigma_i^x}$ and measurement in the computational basis. Following Ref. [28], it suffices to consider QAOA circuits with $\beta = \pi/4$. The output state is $\tilde{H}^{\otimes N} e^{-i\gamma H_z} H^{\otimes N}|0^N\rangle$ for some cost function C diagonal in the computational basis.

A.2.1 Generalized gap of a function

The main idea behind proving exact sampling hardness is to examine a particular output amplitude, say the amplitude of the $|0^N\rangle$ basis state. In Ref. [208], the output state after a so-called IQP circuit (which only differs from the one here in that the final rotation is a global Hadamard $H^{\otimes N}$ instead of $\tilde{H}^{\otimes N}$) has an amplitude proportional to a quantity known as the *gap* of a Boolean function, $\mathrm{gap}(f) = \sum_{x:f(x)=0} 1 - \sum_{x:f(x)=1} 1$, the difference in the number of inputs that map to 1 and the number of inputs that map to 0 under f. Finding the gap of a general function is a GapP-complete problem. This is a very hard problem since the class GapP includes #P, which in turn includes the whole of NP. The authors of Ref. [208] prove that the gap of a degree-3 polynomial over $\mathbb{Z}_2$, f, may be expressed as an output amplitude of an IQP circuit. They also show that the finding the gap of such functions f is still GapP-complete. Following Ref. [208], we examine the $|0^N\rangle$ output amplitude of a QAOA state:

$$\langle 0^N| \tilde{H}^{\otimes N} e^{-i\gamma H_z} H^{\otimes N} |0^N\rangle = \frac{1}{2^N} \sum_{x,y} \langle y| \, i^{\sum_i y_i + \tilde{f}(x)} \, |x\rangle \,, \tag{A.14}$$

where now we define the function $\tilde{f}$ to have the range $\mathbb{Z}_4$ and the Hamiltonian H_z satisfies $e^{-i\gamma H_z} |x\rangle = i^{\tilde{f}(x)} |x\rangle$ for a computational basis state $|x\rangle$. The output amplitude is thus proportional to a 'generalized gap' $\mathrm{ggap}(f) := \sum_{x:f(x)=0} 1 + i \sum_{x:f(x)=1} 1 + i^2 \sum_{x:f(x)=2} 1 + i^3 \sum_{x:f(x)=3} 1$ of a function $f(x) = \tilde{f}(x) + \mathrm{wt}(x)$, where $\mathrm{wt}(x)$ is the Hamming weight of x. This modified function $f(x)$ is also a degree-3 polynomial over $\mathbb{Z}_4$. Note that this restriction to degree-3 comes from the fact that the gates Z, CZ

and CCZ are universal for classical computation (indeed, the Toffoli alone is universal for classical computation) and there is a natural degree-3 polynomial coming from this construction. The quantity we have defined, $\mathrm{ggap}(f)$, can be easily shown to be GapP-hard to compute, by reducing gap to ggap. This suffices for exact sampling hardness assuming the polynomial hierarchy (PH) does not collapse.

A.2.2 Approximate sampling hardness

For approximate sampling hardness, we need two other properties, namely anti-concentration and a worst-to-average case reduction. Anti-concentration of a circuit roughly says that the output probability is sufficiently spread out among all possible outcomes so that not many output probabilities are too small. We choose a random family of QAOA circuits by choosing H_z such that the function $f(x)$ is a degree-3 polynomial $\sum_{i,j,k} a_{i,j,k} x_i x_j x_k + \sum_{i,j} b_{i,j} x_i x_j + \sum_i c_i x_i$ with uniformly random weights $b_{i,j}$ and $c_i \in \mathbb{Z}_4$. Anti-concentration then follows from the Paley-Zygmund inequality and Lemma 4 of the Supplemental Material of Ref. [208] (with $r = s = 4$).

Finally, we need to show that the problem of approximating the generalized gap is average-case hard. Currently, no scheme for quantum computational supremacy has achieved this, and the best known result in this direction is in Ref. [209], where the authors show a worst-to-average case reduction for the problem of *exactly* computing an output probability of a random quantum circuit. The authors remark that their techniques may be extended to any distribution parametrized by a continuous variable. In principle, we have such a parameter γ available here, which continuously changes the parameters $b_{i,j}$

and c_i. However, we have only shown anti-concentration when the weights $b_{i,j}$ and c_i are chosen from a finite set. It remains to be seen whether one can have the property of anti-concentration and average-case hardness holding at the same time for some specific QAOA output distribution.

A.3 Trapped-ion experimental systems

In this work two quantum simulators have been used, referred to as system 1 and 2. System 1 [202] is a room-temperature ion-trap apparatus, consisting of a 3-layer linear Paul trap with transverse center-of-mass (COM) motional frequency $\nu_{\text{COM}} = 4.7$ MHz and axial center-of-mass frequencies ν_x ranging from 0.39 to 0.6 MHz depending on the number of trapped ions. In this system Langevin collisions with the residual background gas in the ultra high vacuum (UHV) apparatus are the main limitation to ion chain lifetime [237]. These events can melt the crystal and eject the ions from the trap because of rf-heating or other mechanisms.

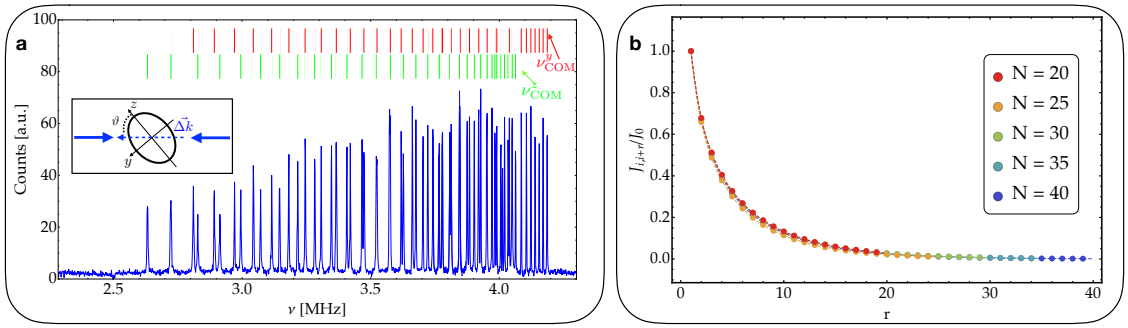


Figure A.4: System 2 characterization. (a) Sideband resolved spectroscopy of a 32 ion chain with frequencies $\nu^y_{\mathrm{COM}} = 4.18$ MHz and $\nu^z_{\mathrm{COM}} = 4.06$ MHz, with both transverse families identified. Inset: geometrical configuration of the global Raman beams (blue arrows) with respect to the transverse principal axes of the trap (black arrows). The ellipsoid shows qualitatively an equipotential surface of the trap. (b) Average spin-spin interaction matrix element $J_{i,i+r}$ as a function of ion separation $r = |i - j|$ for the data taken in Fig. 2c in the main text, calculated with the system parameters directly measured with sideband spectroscopy, using Eq. (A.16). The results are normalized to the average nearest-neighbour coupling J_0 for each system size.

System 2 [32] is a cryogenic ion-trap apparatus based on a linear blade trap with four segmented gold coated electrodes. The trap is held at 6.5 K in a closed cycle cryostat, where differential cryo-pumping reduces the background pressure at low 10^{-12} Torr level, which allows for long storage times of large ion chains. For this reason system 2 has been used to perform the QAOA with a large number of qubits (Fig. 2b) or when a large number of measurements was required (Fig. 4). The two transverse trap frequencies are $\nu^y_{\mathrm{COM}} = 4.4$ MHz and $\nu^z_{\mathrm{COM}} = 4.26$ MHz, and the axial frequency ranges from 0.27 to 0.46 MHz.

A.3.1 State preparation

The qubit is initialized by applying resonant 369.5 nm light for about 20 μs to optically pump into the $|\downarrow\rangle_z$ state. To perform global rotations in the Bloch sphere, we apply two far-detuned, non-copropagating Raman beams whose beatnote is tuned to the

hyperfine splitting $\nu_0 = 12.642821$ GHz of the clock states ${}^2S_{1/2}\,|F=0, m_F=0\rangle$ and ${}^2S_{1/2}\,|F=1, m_F=0\rangle$ encoding the qubit [238]. State preparation in our implementation of the QAOA requires qubit initialization in the $\downarrow\rangle_z$ state by optically pumping the ions and then a global rotation into the $\uparrow\rangle_y$ state using stimulated Raman transitions. We detect the state of each ion at the end of each experimental sequence using state-dependent fluorescence, with single site resolution. In order to improve the accuracy of global qubit rotations, we employ a composite pulse sequence based on the dynamical decoupling BB1 scheme [239]. This allows us to compensate for inhomogeneity due to the Raman beam's Gaussian profile and achieve nearly 99% state preparation fidelity. The BB1 four pulse sequence is:

$$U_1(\pi/2) = e^{-i\frac{\pi}{2}\sigma_i^{\theta}} e^{-i\pi\sigma_i^{3\theta}} e^{-i\frac{\pi}{2}\sigma_i^{\theta}} e^{-i\frac{\pi}{4}\sigma_i^{x}},$$

where after the first $\pi/2$ rotation $e^{-i\frac{\pi}{4}\sigma_i^x}$, three additional rotations are applied: a π-pulse along an angle $\theta = \cos^{-1}(-1/16) = 93.6\,\mathrm{deg}$, a 2π-pulse along 3θ, and another π-pulse along θ. The axes of these additional rotations are in the x-y plane of the Bloch sphere with the specified angle referenced to the x-axis.

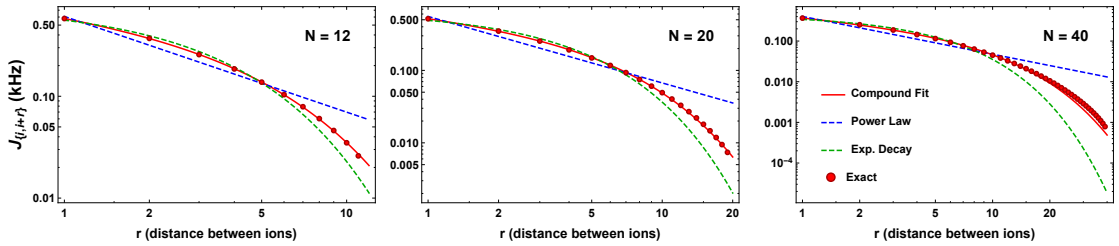


Figure A.5: Log-log plot of spin-spin interactions: red points represent the average Ising couplings between spins separated by distance $r = |i - j|$, calculated from experimental parameters using Eq. A.15. These plots show the exact average couplings and fits corresponding to the $N = 12$ and $N = 20$ gradient descent experiments (Fig. 3 in the main text) and the $N = 40$ exhaustive search experiment (Fig. 2c in the main text). The power law fit (blue dashed curve) fails to match the couplings for larger spin separations, as does an exponential fit (green dashed curve). The compound formula (Eq. A.18) fits well the actual couplings for all spin separations, even for a chain of 40 ions. The fitted parameters $\{J_0, \alpha', \beta'\}$ for N = 12, 20, and 40 are $\{0.580, 0.322, 0.229\}$, $\{0.517, 0.318, 0.181\}$, and $\{0.369, 0.383, 0.134\}$ respectively.

A.3.2 Generating the Ising Hamiltonian

We generate spin-spin interactions by employing a spin dependent force with a pair of non-copropagating 355 nm Raman beams, with a wavevector difference Δk aligned along the transverse motional modes of the ion chain. The two off-resonant Raman beams are controlled using acousto-optic modulators which generate two interference beatnotes at frequencies $\nu_0 \pm \mu$ in the Mølmer-Sørensen configuration [240]. In the Lamb-Dicke regime, the laser-ion interaction gives rise to the effective spin-spin Hamiltonian in Eq. (1) in the main text, where the coupling between the i-th and j-th ion is:

$$J_{ij} = \Omega^2 \nu_R \sum_m \frac{b_{im} b_{jm}}{\mu^2 - \nu_m^2}. \tag{A.15}$$

Here Ω is the Rabi frequency, $\nu_R = h\Delta k^2/(8\pi^2 M)$ is the recoil frequency, ν_m is the frequency of the m-th normal mode, b_{im} is the eigenvector matrix element for the i-th

ion's participation to the m-th normal mode ($\sum_i |b_{im}|^2 = \sum_m |b_{im}|^2 = 1$) [241], and M is the mass of a single ion.

Differently from system 1, where the wavevector difference Δk of the Raman beams is aligned along one of the principal axes of the trap, in system 2 the spin-spin interaction stems from the off-resonant coupling to both families of transverse normal modes. Eq. (A.15) is then generalized to:

$$\begin{aligned} J_{ij} &= J_{ij}^{y} + J_{ij}^{z}, \\ J_{ij}^{\ell} &= \Omega_{\ell}^{2} \nu_{R}^{\ell} \sum_{m} \frac{b_{im} b_{jm}}{\mu^2 - \left(\nu_{m}^{\ell}\right)^2}, \ \ell = y, z, \end{aligned} \tag{A.16}$$

where ν_R^ℓ is the recoil frequency given by the projection of the Raman wavevector Δk along the two transverse principal axes of the trap $\ell = y, z$. We infer an angle $\vartheta \sim 40^o$ between Δk and the z principal axis (see inset in Fig. A.4a) from the ratio between the resonant spin-phonon couplings to the two transverse COM modes. Before every experiment, we perform Raman sideband cooling on both the COM and the two nearby tilt modes for both transverse mode families.

As we scale up the number of qubits (see Fig. 2c in the main text), we vary the axial confinement in order to maintain a self-similar functional form of the spin-spin interaction (see Fig. A.4b). For the data in Fig. 2c in the main text, we set the detuning to $\delta = \mu - \omega_{\mathrm{COM}}^{y} = 2\pi \times 45$ kHz and the axial frequency to $\nu_x = 0.46, 0.37, 0.36, 0.31, 0.27$ MHz, for $N = 20, 25, 30, 35, 40$ respectively. For the data in Fig. 4 in the main text, the detuning is $\delta/2\pi = 45$ kHz and the $\nu_x = 0.54$ MHz.

A.3.3 Fitting Ising Couplings to Analytic Form

By directly measuring trap parameters and spin-phonon couplings, we can calculate the spin-spin interaction matrix J_{ij} with Eqs. (A.15) and (A.16). However, in order to efficiently compute the ground state energy of the Hamiltonian in Eq. (1) (see main text) for $N \gtrsim 25$ using DMRG, we approximate the Ising couplings using a translational invariant analytic function of the ion separation $r = |i - j|$. For $N < 20$ the spin-spin coupling J_{ij} between the two qubits at distance r is well approximated by a power law decay:

$$J_{ij} \approx \frac{J_0}{r^\alpha}, \tag{A.17}$$

where, as stated in the main text, J_0 is the average nearest-neighbor coupling and α is the power law exponent [203]. However for larger system sizes, this approximation fails to capture the actual decay of the interaction matrix.

In order to use the DMRG algorithm to accurately compute the ground state energies, we developed a compound function to better fit our couplings. This function is a product of a power law decay and an exponential decay parametrized by J_0, α' and β':

$$J_{ij} \approx \frac{J_0}{r^{\alpha'}} e^{-\beta'(r-1)} \tag{A.18}$$

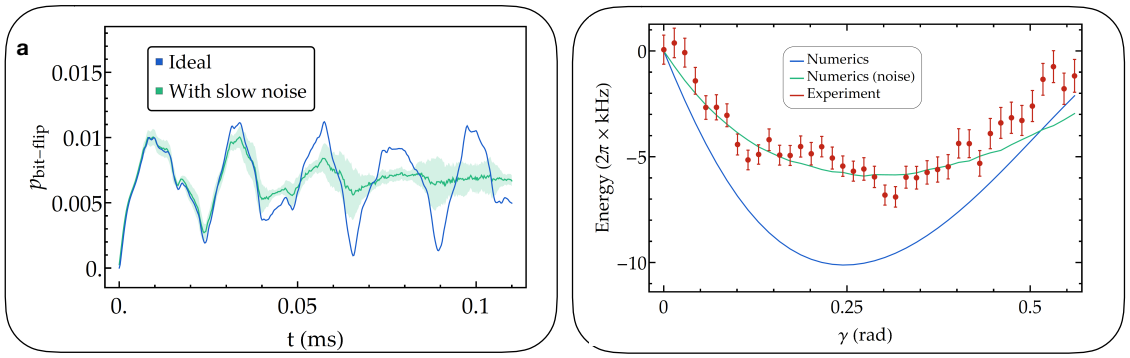
As seen in Fig. A.5, this functional form fits well the exact Ising couplings even for a chain of 40 ions, while both a power law and a pure exponential fit diverge significantly.

A.3.4 State Detection

We detect the ion spin state by globally rotating all the spins into the measurement basis with a composite BB1 $\pi/2$ pulse as described above, to rotate the x or y basis into the z basis), followed by the scattering of resonant laser radiation on the $^2\mathrm{S}_{1/2}\,|F=1\rangle \leftrightarrow {}^2\mathrm{P}_{1/2}\,|F=0\rangle$ cycling transition (wavelength near 369.5 nm and radiative linewidth $\gamma/2\pi \approx 20$ MHz). If the atom is projected in the $\uparrow\rangle_z$ "bright" state, it fluoresces strongly, while if projected in the $\downarrow\rangle_z$ "dark" state it fluoresces almost no photons because the laser is far from resonance [238].

In both systems the fluorescence of the ion chain is imaged onto an Electron Multiplying Charge Coupled Device (EMCCD) camera (Model Andor iXon Ultra 897) using an imaging objective with 0.4 numerical aperture and a magnification of 90x for both systems. The fluorescence of each ion covers roughly a 7x7 array of pixels on the EMCCD. After collecting the fluorescence for an integration time of 0.65 (1) ms for system 1 (2), we use a binary threshold to determine the state of each ion, discriminating the quantum state of each ion with near 98% (97%) accuracy in system 1 (2). The residual 2 (3)% errors include off-resonant optical pumping of the ion between spin states during detection as well as detector cross-talk between adjacent ions, readout noise, and background counts.

In system 2 the individual ion range-of-interests (ROIs) on the camera are updated with periodic diagnostic images, acquired by applying a nearly resonant cooling laser for 50 ms so that each ion fluoresces strongly regardless of its state. The signal to background noise ratio in the diagnostic shots is larger than 100, yielding precise knowledge of the



, we explain most of the discrepancy between our experimental performance and the ideal QAOA energy output.

Figure A.6: Errors in trapped-ion quantum simulator: (a) Phonon-assisted bit-flips per ion predicted by evolving the coherent off-resonant spin-phonon drive for 12 ions. The simulation includes slow drifts of the trap frequency and of the laser power over 500 shots, each including a Hamiltonian evolution of 0.11 ms, with $\delta/2\pi = 45$ kHz and $\Omega/2\pi = 440$ kHz. The shaded region is defined as the average p_i plus and minus one standard deviation (see main text for details). (b) Energy as a function of the γ parameter scan for Fig. 4 in the main text. Taking into account our total bit-flip error budget together with uncompensated light shift

ions' center locations and taking into account the slow $\sim 2\,\mu$m pk-pk drift due to thermal expansion/contraction of the cryostat. Ion separations range from 1.5 μm to 3.5 μm depending on the trap settings and the distance from the chain center, and are always much larger than the resolution limit of the imaging system. We utilize the pre-determined ion centers to process the individual detection shots and optimize the integration area on the EMCCD camera to collect each ion's fluorescence while minimizing cross-talk. We estimate cross-talk to be dominated by fluorescence from nearest-neighbor, which can cause a dark ion to be erroneously read as bright.

A.3.5 Error sources

The fidelity of the quantum simulation is limited by experimental noise that causes the system to depart from the ideal evolution and that can have several sources that are

reviewed below. One important error source is off-resonant excitation of motional modes of the ion chain, which causes residual spin motion-entanglement. When the motion is traced out at the end of the measurement this results in a finite probability of an unwanted bit-flip. The probability of this error to occur on the ith ion [202] is proportional to $p_i \sim \sum_{m=1}^{N} (\eta_{im}\Omega/\delta_m)^2$, where $\eta_{im} = b_{im}\sqrt{\nu_R/\nu_{\mathrm{COM}}}$ (see Eq. (A.15)) and $\delta_m = \mu - \omega_m$ is the beatnote detuning from the m-th normal mode. We trade off a lower error for a weaker spin-spin coupling by choosing a δ_{COM} such that $(\eta_{\mathrm{COM}}\Omega/\delta_{\mathrm{COM}})^2 \lesssim 1/10$. By considering the off-resonant contributions of all the modes (see Fig. A.6), we estimate the phonon error to cause about 1% bit-flip per ion. Additionally, bit-flip errors are affected by fluctuations in the trap frequency and laser light intensity at the ions' location. To take this into account, we included slow drifts and fluctuations of the trap frequency and of the laser power on the timescale of 500 experimental repetitions assuming noise spectral density falling as $1/f$. Given our typical trap frequency and laser power fluctuations, we assume a relative standard deviation $\Delta\Omega/\Omega \sim 2\%$ and $\Delta\delta_{\mathrm{COM}}/\delta_{\mathrm{COM}} \sim 9\%$ over the timescale required to average over quantum projection noise and we end up estimating an average bit-flip probability $p_i \sim 1\%$ (see Fig. A.6a). Moreover, laser intensity, beam steering and trap frequency slow drifts over the time scale of a few hours required for data-taking cause averaging over different Ising parameters J_0. In particular, beam steering fluctuations create an imbalance between the red and blue $\nu_0 \pm \mu$ beatnotes at the ions, producing an effective B_z noisy field, that has been estimated to be as high as $\sim 0.65 J_0$. To take into account these drifts, we calculated several evolutions sampling from a Gaussian distribution of values of B_z and J_0, using as a variance the standard deviations ($\sigma_{J_0} = 0.18 J_0$ and $\sigma_{B_z} = 0.4 J_0$) observed in the experiment. Another source

of bit-flip errors is imperfect detection. Off-resonant pumping limits our average detection fidelity to $98\%(97\%)$ for system 1 (2). A detection error is equivalent to a random bit-flip event so the two errors will sum up. A specific source of noise in system 2 is mechanical vibrations at 41 Hz and 39 Hz due to residual mechanical coupling to the cryostat [32]. This is equivalent to phase-noise on the Raman beams, which leads to dephasing of the qubits. Other less important noise sources are related to off-resonant Raman scattering errors during the Ising evolution (estimated in $7 \cdot 10^{-5}$ per ion) and RF heating of the transverse COM motional mode of the ion chain in system 1.

In Fig. A.6b, we plot the experimentally measured energy as a function of γ, and the corresponding theoretical curves with and without incorporating errors. Using the time dependent average bit-flip probability evolution that we estimated from our error model considering phonons and detection errors and averaging over slow drifts in experimental parameters J_0 and B_z, we get a good agreement with the experimental data (see also Fig. 2c in the main text, where the same parameters have been used), showing that we have a good understanding of the noise sources in our system.

Bibliography

[1] Yuri Manin. *Computable and Uncomputable*. Sovetskoye Radio, 1980.

[2] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of statistical physics*, 22(5):563–591, 1980.

[3] Richard P Feynman. Simulating physics with computers. In *Feynman and computation*, pages 133–153. CRC Press, 2018.

[4] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[5] Martin Suchara, Arvin Faruque, Ching-Yi Lai, Gerardo Paz, Frederic T Chong, and John Kubiatowicz. Comparing the overhead of topological and concatenated quantum error correction. *arXiv preprint arXiv:1312.2316*, 2013.

[6] Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, and Jay M. Gambetta. Validating quantum computers using randomized model circuits. *Phys. Rev. A*, 100:032328, Sep 2019.

[7] C. Monroe and J. Kim. Scaling the ion trap quantum processor. *Science*, 339(6124):1164–1169, 2013.

[8] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Physical Review A*, 89(2), 2 2014.

[9] Teresa Brecht, Wolfgang Pfaff, Chen Wang, Yiwen Chu, Luigi Frunzio, Michel H. Devoret, and Robert J. Schoelkopf. Multilayer microwave integrated quantum circuits for scalable quantum computing. *npj Quantum Information*, 2(16002), 2016.

[10] Ying Li and Simon C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X*, 7:021050, Jun 2017.

[11] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.*, 119:180509, Nov 2017.

[12] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 10 2019.

[13] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, Apr 2008.

[14] Aniruddha Bapat, Zachary Eldredge, James R. Garrison, Abhinav Deshpande, Frederic T. Chong, and Alexey V. Gorshkov. Unitary entanglement construction in hierarchical networks. *Physical Review A*, 98(6), 2018.

[15] Andrew M. Childs, Eddie Schoute, and Cem M. Unsal. Circuit transformations for quantum architectures. In Wim van Dam and Laura Mancinska, editors, *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, volume 135 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:24, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[16] Andrew Y. Guo, Minh C. Tran, Andrew M. Childs, Alexey V. Gorshkov, and Zhe-Xuan Gong. Signaling and scrambling with strongly long-range interactions. *Phys. Rev. A*, 102:010401, Jul 2020.

[17] Matthäus Halder, Alexios Beveratos, Nicolas Gisin, Valerio Scarani, Christoph Simon, and Hugo Zbinden. Entangling independent photons by time measurement. *Nature physics*, 3(10):692–695, 2007.

[18] Aniruddha Bapat, Eddie Schoute, Alexey V Gorshkov, and Andrew M Childs. Nearly optimal time-independent reversal of a spin chain. *arXiv preprint arXiv:2003.02843*, 2020.

[19] Aniruddha Bapat, Andrew M Childs, Alexey V Gorshkov, Samuel King, Eddie Schoute, and Hrishee Shastri. Quantum routing with fast reversals. *arXiv preprint arXiv:2103.03264*, 2021.

[20] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[21] Zoë Holmes, Kunal Sharma, M Cerezo, and Patrick J Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *arXiv preprint arXiv:2101.02138*, 2021.

[22] Arthur W Leissa. The historical bases of the rayleigh and ritz methods. *Journal of Sound and Vibration*, 287(4-5):961–978, 2005.

[23] L. S. Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.

[24] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[25] Z. C. Yang, A. Rahmani, A. Shabani, H. Neven, and C. Chamon. Optimizing variational quantum algorithms using pontryagin's minimum principle. *Physical Review X*, 7(2):1–8, 2017.

[26] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum Computation by Adiabatic Evolution. *arXiv e-prints*, pages quant–ph/0001106, Jan 2000.

[27] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Phys. Rev. E*, 58:5355–5363, Nov 1998.

[28] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[29] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014.

[30] Aniruddha Bapat and Stephen Jordan. Bang-bang control as a design principle for classical and quantum optimization algorithms. *Quantum Info. Comput.*, 19(5–6):424–446, May 2019.

[31] Lucas T. Brady, Christopher L. Baldwin, Aniruddha Bapat, Yaroslav Kharkov, and Alexey V. Gorshkov. Optimal protocols in quantum annealing and quantum approximate optimization algorithm problems. *Phys. Rev. Lett.*, 126:070505, Feb 2021.

[32] G Pagano, P W Hess, H B Kaplan, W L Tan, P Richerme, P Becker, A Kyprianidis, J Zhang, E Birckelbaw, M R Hernandez, Y Wu, and C Monroe. Cryogenic trapped-ion system for large scale quantum simulation. *Quantum Science and Technology*, 4(1):014004, 2019.

[33] M. B. Hastings. Classical and Quantum Bounded Depth Approximation Algorithms. *arXiv:1905.07047*, May 2019.

[34] Aniruddha Bapat and Stephen P. Jordan. Approximate optimization of the maxcut problem with a local spin algorithm. *Phys. Rev. A*, 103:052413, May 2021.

[35] Rodney Van Meter and Kohei M. Itoh. Fast Quantum Modular Exponentiation. *Phys. Rev. A*, 71(5):052320, 2005.

[36] Rodney Van Meter, W. J. Munro, Kae Nemoto, and Kohei M. Itoh. Arithmetic on a distributed-memory quantum multicomputer. *ACM J. Emerg. Technol. Comput. Syst.*, 3(4):1–23, 2008.

[37] Muhammad Ahsan and Jungsang Kim. Optimization of Quantum Computer Architecture Using a Resource-Performance Simulator. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1108–1113. IEEE Conference Publications, 2015.

[38] T.S. Metodi, D.D. Thaker, and A.W. Cross. A Quantum Logic Array Microarchitecture: Scalable Quantum Data Movement and Computation. In *38th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'05)*, pages 305–318. IEEE, 2005.

[39] L.-M. Duan and C. Monroe. Colloquium: Quantum networks with trapped ions. *Rev. Mod. Phys.*, 82:1209–1224, Apr 2010.

[40] M H Devoret and R J Schoelkopf. Superconducting Circuits for Quantum Information: An Outlook. *Science*, 339(6124):1169–74, 2013.

[41] Philipp Kurpiers, Paul Magnard, Theo Walter, Baptiste Royer, Marek Pechal, Johannes Heinsoo, Yves Salathé, Abdulkadir Akin, Simon Storz, J-C Besse, et al. Deterministic quantum state transfer and remote entanglement using microwave photons. *Nature*, 558(7709):264–267, 2018.

[42] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring Network Structure, Dynamics, and Function Using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11–15, Pasadena, CA USA, 2008.

[43] L. Barrière, C. Dalfó, M. A. Fiol, and M. Mitjana. The generalized hierarchical product of graphs. *Discrete Mathematics*, 309(12):3871–3881, June 2009.

[44] C.D. Godsil and B.D. McKay. A new graph product and its spectrum. *Bulletin of the Australian Mathematical Society*, 18(1):21–28, 1978.

[45] J. J. Bollinger, Wayne M. Itano, D. J. Wineland, and D. J. Heinzen. Optimal frequency measurements with maximally correlated states. *Phys. Rev. A*, 54(6):R4649–R4652, 1996.

[46] Zachary Eldredge, Michael Foss-Feig, Jonathan A. Gross, S. L. Rolston, and Alexey V. Gorshkov. Optimal and secure measurement protocols for quantum sensor networks. *Phys. Rev. A*, 97:042337, Apr 2018.

[47] Zachary Eldredge, Zhe-Xuan Gong, Jeremy T. Young, Ali Hamed Moosavian, Michael Foss-Feig, and Alexey V. Gorshkov. Fast Quantum State Transfer and Entanglement Renormalization Using Long-Range Interactions. *Phys. Rev. Lett.*, 119(17):170503, 2017.

[48] S. Bravyi, M. B. Hastings, and F. Verstraete. Lieb-Robinson Bounds and the Generation of Correlations and Topological Quantum Order. *Phys. Rev. Lett.*, 97(5):050401, 2006.

[49] Gregory Bentsen, Yingfei Gu, and Andrew Lucas. Fast scrambling on sparse graphs. *Proceedings of the National Academy of Sciences*, 116(14):6689–6694, 2019.

[50] Martí Cuquet and John Calsamiglia. Growth of Graph States in Quantum Networks. *Phys. Rev. A*, 86(4):042304, 2012.

[51] Antonio Ać in, J. Ignacio Cirac, and Maciej Lewenstein. Entanglement Percolation in Quantum Networks. *Nat. Phys.*, 3(4):256–259, 2007.

[52] S Perseguers, G J Lapeyre, D Cavalcanti, M Lewenstein, and A Acín. Distribution of entanglement in large-scale quantum networks. *Rep. Prog. Phys.*, 76(9):096001, 2013.

[53] K. Kieling, T. Rudolph, and J. Eisert. Percolation, Renormalization, and Quantum Computing with Nondeterministic Gates. *Phys. Rev. Lett.*, 99(13):130501, 2007.

[54] Martí Cuquet and John Calsamiglia. Entanglement Percolation in Quantum Complex Networks. *Phys. Rev. Lett.*, 103(24):240503, 2009.

[55] Martí Cuquet and John Calsamiglia. Limited-Path-Length Entanglement Percolation in Quantum Complex Networks. *Phys. Rev. A*, 83(3):032319, 2011.

[56] Yang Wang, D. Chakrabarti, Chenxi Wang, and C. Faloutsos. Epidemic Spreading in Real Networks: An Eigenvalue Viewpoint. In *22nd International Symposium on Reliable Distributed Systems, 2003. Proceedings.*, pages 25–34. IEEE Comput. Soc, 2003.

[57] Duncan J. Watts and Steven H. Strogatz. Collective Dynamics of 'small World' Networks. *Nature*, 393(6684):440–442, 1998.

[58] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

[59] Réka Albert and Albert-László Barabási. Statistical Mechanics of Complex Networks. *Rev. Mod. Phys.*, 74(1):47–97, 2002.

[60] S. Perseguers, M. Lewenstein, A. Aćin, and J. I. Cirac. Quantum Random Networks. *Nat. Phys.*, 6(7):539–543, 2010.

[61] C. Di Franco and D. Ballester. Optimal Path for a Quantum Teleportation Protocol in Entangled Networks. *Phys. Rev. A*, 85(1):010303, 2012.

[62] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller. Quantum Repeaters: The Role of Imperfect Local Operations in Quantum Communication. *Phys. Rev. Lett.*, 81(26):5932–5935, 1998.

[63] Frank Harary. The number of linear, directed, rooted, and connected graphs. *Trans. Amer. Math. Soc.*, 78:445–463, 1955.

[64] Mikhail Gromov. *Metric Structures for Riemannian and Non-Riemannian Spaces*. Modern Birkhäuser Classics. Birkhäuser Basel, 2007.

[65] Wei Chen, Wenjie Fang, Guangda Hu, and Michael W. Mahoney. On the Hyperbolicity of Small-World and Treelike Random Graphs. *Internet Math.*, 9(4):434–491, 2013.

[66] Poonsuk Lohsoonthorn. *Hyperbolic Geometry of Networks*. PhD Thesis, University of Southern California, Los Angeles, CA, USA, 2003.

[67] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic Geometry of Complex Networks. *Phys. Rev. E*, 82(3):036106, 2010.

[68] Fabien de Montgolfier, Mauricio Soto, and Laurent Viennot. Treewidth and Hyperbolicity of the Internet. In *Proceedings of the 2011 IEEE 10th International Symposium on Network Computing and Applications*, NCA '11, pages 25–32, Washington, DC, USA, 2011. IEEE Computer Society.

[69] Marián Boguñá, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the Internet with Hyperbolic Mapping. *Nat. Commun.*, 1:62, 2010.

[70] Alicia J Kollár, Mattias Fitzpatrick, and Andrew A Houck. Hyperbolic lattices in circuit quantum electrodynamics. *Nature*, 571(7763):45–50, 2019.

[71] Sang-il Oum. Rank-Width and Vertex-Minors. *J. Comb. Theory Ser. B*, 95(1):79–100, 2005.

[72] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear Time Solvable Optimization Problems on Graphs of Bounded Clique Width. pages 1–16. Springer, Berlin, Heidelberg, 1998.

[73] Charles E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.*, C-34(10):892–901, 1985.

[74] Michael William Newman. *The Laplacian Spectrum of Graphs*. Master's Thesis, University of Manitoba, Winnipeg, Canada, 2000.

[75] Bojan Mohar. The Laplacian Spectrum of Graphs. In *Graph Theory, Combinatorics, and Applications*, pages 871–898. Wiley, 1991.

[76] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[77] Donny Cheung, Dmitri Maslov, and Simone Severini. Translation Techniques Between Quantum Circuit Architectures. In *Workshop on Quantum Information Processing*, 2007.

[78] Norbert M. Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A. Landsman, Kenneth Wright, and Christopher Monroe. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310, 3 2017.

[79] Robert S. Pindyck and Daniel L. Rubinfeld. *Microeconomics*. Pearson, 2013.

[80] Zoltán Füredi. Graphs of Diameter 3 with the Minimum Number of Edges. *Graphs Comb.*, 6(4):333–337, 1990.

[81] A. J. Hoffman and R. R. Singleton. On Moore Graphs with Diameters 2 and 3. *IBM J. Res. Dev.*, 4(5):497–504, 1960.

[82] Mirka Miller and Jozef Sirán. Moore graphs and beyond: A survey of the degree/diameter problem. *Electron. J. Comb.*, 1000:DS14–Dec, 2005.

[83] Guillermo Pineda-Villavicencio and David R. Wood. The degree-diameter problem for sparse graph classes. *Electron. J. Comb.*, 22(2):2.46, 2015.

[84] Zachary Eldredge et al. unitary-modular. `https://github.com/zeldredge/unitary-modular`. GitHub repository.

[85] D. Maslov, S. M. Falconer, and M. Mosca. Quantum circuit placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(4):752–763, 2008.

[86] M. Pedram and A. Shafaei. Layout optimization for quantum circuits with linear nearest neighbor architectures. *IEEE Circuits and Systems Magazine*, 16(2):62–74, 2016.

[87] Thomas Häner, Damian S Steiger, Krysta Svore, and Matthias Troyer. A software methodology for compiling quantum programs. *Quantum Science and Technology*, 3(2):020501, feb 2018.

[88] Damian S. Steiger, Thomas Häner, and Matthias Troyer. ProjectQ: An open source software framework for quantum computing. *Quantum*, 2:49, 2018.

[89] Konstantin Andreev and Harald Racke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, October 2006.

[90] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[91] R. Kleinberg. Geographic Routing Using Hyperbolic Space. In *IEEE INFOCOM 2007*, pages 1902–1909. IEEE, 2007.

[92] E. Jonckheere and P. Lohsoonthorn. Geometry of Network Security. In *Proceedings of the 2004 American Control Conference*, volume 2, pages 976–981 vol.2, 2004.

[93] Victor Chepoi, Feodor F. Dragan, Bertrand Estellon, Michel Habib, Yann Vaxès, and Yang Xiang. Additive Spanners and Distance and Routing Labeling Schemes for Hyperbolic Graphs. *Algorithmica*, 62(3-4):713–732, 2012.

[94] Zachary Eldredge, Leo Zhou, Aniruddha Bapat, James R. Garrison, Abhinav Deshpande, Frederic T. Chong, and Alexey V. Gorshkov. Entanglement bounds on the performance of quantum computing architectures. *Phys. Rev. Research*, 2:033316, Aug 2020.

[95] A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg. Implementation of shor's algorithm on a linear nearest neighbour qubit array. *Quantum Info. Comput.*, 4(4):237–251, July 2004.

[96] Peter Høyer and Robert Špalek. Quantum fan-out is powerful. *Theory of Computing*, 1(5):81–103, 2005.

[97] David P. DiVincenzo. The Physical Implementation of Quantum Computation. *Fortschr. Phys.*, 48(9-11):771–783, 2000.

[98] H. J. Kimble. The quantum internet. *Nature*, 453(7198):1023–1030, June 2008.

[99] R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather. Efficient distributed quantum computing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153), 2013.

[100] Sougato Bose. Quantum communication through spin chain dynamics: an introductory overview. *Contemp. Phys.*, 48(1):13–30, jan 2007.

[101] Elliott H. Lieb and Derek W. Robinson. The finite group velocity of quantum spin systems. *Communications in Mathematical Physics*, 28:251–257, 1972.

[102] J. Eisert, M. Cramer, and M. B. Plenio. Colloquium: Area laws for the entanglement entropy. *Rev. Mod. Phys.*, 82(1):277–306, feb 2010.

[103] Jeongwan Haah, Matthew Hastings, Robin Kothari, and Guang Hao Low. Quantum algorithm for simulating real time evolution of lattice hamiltonians. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 350–360. IEEE, IEEE, October 2018.

[104] W. Dür, G. Vidal, J. I. Cirac, N. Linden, and S. Popescu. Entanglement capabilities of nonlocal hamiltonians. *Physical Review Letters*, 87(13), September 2001.

[105] A. M. Childs, D. W. Leung, F. Verstraete, and G. Vidal. Asymptotic entanglement capacity of the ising and anisotropic heisenberg interactions. *Quantum Information and Computation*, 3(2):97–105, 2003.

[106] A. M. Childs, D. W. Leung, and G. Vidal. Reversible simulation of bipartite product hamiltonians. *IEEE Transactions on Information Theory*, 50(6):1189–1197, 2004.

[107] C. H. Bennett, A. W. Harrow, D. W. Leung, and J. A. Smolin. On the capacities of bipartite hamiltonians and unitary gates. *IEEE Transactions on Information Theory*, 49(8):1895–1911, July 2003.

[108] Sergey Bravyi. Upper bounds on entangling rates of bipartite Hamiltonians. *Physical Review A*, 76(5), November 2007.

[109] Karel Van Acoleyen, Michaël Mariën, and Frank Verstraete. Entanglement Rates and Area Laws. *Physical Review Letters*, 111(17), October 2013.

[110] Koenraad M. R. Audenaert. Quantum skew divergence. *Journal of Mathematical Physics*, 55(11):112202, November 2014.

[111] Michaël Mariën, Koenraad M. R. Audenaert, Karel Van Acoleyen, and Frank Verstraete. Entanglement rates and the stability of the area law for the entanglement entropy. *Communications in Mathematical Physics*, 346(1):35–73, 2016.

[112] Sougato Bose. Quantum communication through an unmodulated spin chain. *Phys. Rev. Lett.*, 91(20), November 2003.

[113] Giulia Gualdi, Vojtech Kostak, Irene Marzoli, and Paolo Tombesi. Perfect state transfer in long-range interacting spin chains. *Phys. Rev. A*, 78(2), August 2008.

[114] Matthias Christandl, Nilanjana Datta, Artur Ekert, and Andrew J. Landahl. Perfect state transfer in quantum spin networks. *Physical Review Letters*, 92(18), May 2004.

[115] Matthias Christandl, Nilanjana Datta, Tony C. Dorlas, Artur Ekert, Alastair Kay, and Andrew J. Landahl. Perfect transfer of arbitrary states in quantum spin networks. *Physical Review A*, 71(3), March 2005.

[116] L. Campos Venuti, C. Degli Esposti Boschi, and M. Roncaglia. Qubit teleportation and transfer across antiferromagnetic spin chains. *Physical Review Letters*, 99(6), August 2007.

[117] Leonardo Banchi, Abolfazl Bayat, Paola Verrucchi, and Sougato Bose. Nonperturbative entangling gates between distant qubits using uniform cold atom chains. *Physical Review Letters*, 106(14), April 2011.

[118] N. Y. Yao, L. Jiang, A. V. Gorshkov, Z.-X. Gong, A. Zhai, L.-M. Duan, and M. D. Lukin. Robust quantum state transfer in random unpolarized spin chains. *Phys. Rev. Lett.*, 106(4), jan 2011.

[119] C. Di Franco, M. Paternostro, and M. S. Kim. Perfect state transfer on a spin chain without state initialization. *Phys. Rev. Lett.*, 101(23), dec 2008.

[120] Claudio Albanese, Matthias Christandl, Nilanjana Datta, and Artur Ekert. Mirror inversion of quantum states in linear registers. *Physical Review Letters*, 93(23), November 2004.

[121] Tao Shi, Ying Li, Zhi Song, and Chang-Pu Sun. Quantum-state transfer via the ferromagnetic chain in a spatially modulated field. *Physical Review A*, 71(3), March 2005.

[122] Peter Karbach and Joachim Stolze. Spin chains as perfect quantum state mirrors. *Physical Review A*, 72(3), 2005.

[123] Robert Raussendorf. Quantum computation via translation-invariant operations on a chain of qubits. *Physical Review A*, 72(5), nov 2005.

[124] Joseph Fitzsimons and Jason Twamley. Globally controlled quantum wires for perfect qubit transport, mirroring, and computing. *Physical Review Letters*, 97(9), sep 2006.

[125] P. Kumar and S. Daraeizadeh. Parity-based mirror inversion for efficient quantum state transfer and computation in nearest-neighbor arrays. *Physical Review A*, 91(4), April 2015.

[126] David T. Stephen, Hendrik Poulsen Nautrup, Juani Bermejo-Vega, Jens Eisert, and Robert Raussendorf. Subsystem symmetries, quantum cellular automata, and computational phases of quantum matter. *Quantum*, 3:142, may 2019.

[127] Trithep Devakul and Dominic J. Williamson. Universal quantum computation using fractal symmetry-protected cluster phases. *Physical Review A*, 98(2), aug 2018.

[128] Robert Raussendorf, Cihan Okay, Dong-Sheng Wang, David T. Stephen, and Hendrik Poulsen Nautrup. Computationally universal phase of quantum matter. *Physical Review Letters*, 122(9), mar 2019.

[129] Morten Kjaergaard, Mollie E Schwartz, Jochen Braumüller, Philip Krantz, Joel I-J Wang, Simon Gustavsson, and William D Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11:369–395, 2020.

[130] C. H. Bennett, J. I. Cirac, M. S. Leifer, D. W. Leung, N. Linden, S. Popescu, and G. Vidal. Optimal simulation of two-qubit hamiltonians using general local operations. *Physical Review A*, 66(1), July 2002.

[131] G. Vidal, K. Hammerer, and J. I. Cirac. Interaction cost of nonlocal gates. *Physical Review Letters*, 88(23):237902, 2002.

[132] Noga Alon, F. R. K. Chung, and R. L. Graham. Routing permutations on graphs via matchings. *SIAM Journal on Discrete Mathematics*, 7(3):513–530, 5 1994.

[133] Norman Y Yao, Chris R Laumann, Alexey V Gorshkov, Hendrik Weimer, Liang Jiang, J Ignacio Cirac, Peter Zoller, and Mikhail D Lukin. Topologically protected quantum state transfer in a chiral spin liquid. *Nature Communications*, 4(1):1–8, 2013.

[134] Hong Yao and Steven A Kivelson. Exact chiral spin liquid with non-abelian anyons. *Physical Review Letters*, 99(24):247203, 2007.

[135] Alexei Kitaev. Anyons in an exactly solved model and beyond. *Annals of Physics*, 321(1):2–111, 2006.

[136] Yu-An Chen. Exact bosonization in arbitrary dimensions. *Physical Review Research*, 2(3):033527, 2020.

[137] Aniruddha Bapat, Andrew M. Childs, Dhruv Devulapalli, Alexey V. Gorshkov, and Eddie Schoute. In preparation, 2021.

[138] Donald E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, 2 edition, 1998.

[139] Roy Oste and Joris Van der Jeugt. Tridiagonal test matrices for eigenvalue computations: Two-parameter extensions of the clement matrix. *J. Comput. Appl. Math.*, 314:30–39, apr 2017.

[140] Ole H Hald. Inverse eigenvalue problems for jacobi matrices. *Linear Algebra and Its Applications*, 14(1):63–85, 1976.

[141] John Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018.

[142] D. W. Berry, A. M. Childs, and R. Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809, 10 2015.

[143] C. A. Fuchs and J. van de Graaf. Cryptographic distinguishability measures for quantum-mechanical states. *IEEE Transactions on Information Theory*, 45(4):1216–1227, 1999.

[144] Doug McClure and Jay Gambetta. Quantum computation center opens, 2019.

[145] Mehdi Saeedi, Robert Wille, and Rolf Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Information Processing*, 10(3):355–377, 6 2011.

[146] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 41:1–41:6, New York, NY, USA, 2013. ACM.

[147] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Qubit placement to minimize communication overhead in 2d quantum architectures. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 1 2014.

[148] 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*, pages 178–183. IEEE, 2015.

[149] Prakash Murali, Jonathan M. Baker, Ali Javadi Abhari, Frederic T. Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In Iris Bahar, Maurice Herlihy, Emmett Witchel, and Alvin R. Lebeck, editors, *ASPLOS '19*, pages 1015–1029. The Association for Computing Machinery, 2019.

[150] Alwin Zulehner and Robert Wille. Compiling SU(4) quantum circuits to IBM QX architectures. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference on - ASPDAC '19*. ACM Press, 2019.

[151] Indranil Banerjee and Dana Richards. New results on routing via matchings on graphs. In Ralf Klasing and Marc Zeitoun, editors, *Fundamentals of Computation Theory*, number 10472 in Lecture Notes in Computer Science, pages 69–81. Springer, 2017.

[152] Louxin Zhang. Optimal bounds for matching routing on trees. *SIAM Journal on Discrete Mathematics*, 12(1):64–77, 1 1999.

[153] S. Lakshmivarahan, Sudarshan K. Dhall, and Leslie L. Miller. Parallel sorting algorithms. volume 23 of *Advances in Computers*, pages 321–323. Elsevier, 1984.

[154] V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 148–157, 1993.

[155] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13(1-2):180–210, February 1995.

[156] Michael A. Bender, Dongdong Ge, Simai He, Haodong Hu, Ron Y. Pinter, Steven Skiena, and Firas Swidan. Improved bounds on sorting by length-weighted reversals. *Journal of Computer and System Sciences*, 74(5):744 – 774, 2008.

[157] Ron Pinter and Steven Skiena. Genomic sorting with length-weighted reversals. *Genome informatics. International Conference on Genome Informatics*, 13:103–11, 01 2002.

[158] Thach Cam Nguyen, Hieu Trung Ngo, and Nguyen Bao Nguyen. Sorting by restricted-length-weighted reversals. *Genomics, Proteomics & Bioinformatics*, 3(2):120 – 127, 2005.

[159] Samuel King, Eddie Schoute, and Hrishee Shastri. reversal-sort. `https://gitlab.umiacs.umd.edu/amchilds/reversal-sort`.

[160] M. Schwartz and P. O. Vontobel. Improved lower bounds on the size of balls over permutations with the infinity metric. *IEEE Transactions on Information Theory*, 63(10):6227–6239, 2017.

[161] Torleiv Kløve. Spheres of permutations under the infinity norm- permutations with limited displacement, 2008.

[162] I. Tamo and M. Schwartz. Correcting limited-magnitude errors in the rank-modulation scheme. *IEEE Transactions on Information Theory*, 56(6):2551–2560, 2010.

[163] Herbert Robbins. A remark on stirling's formula. *The American Mathematical Monthly*, 62(1):26–29, 1955.

[164] T. Hogg and D. Portnov. Quantum optimization. *Information Sciences*, 128(3-4):181–197, 2000.

[165] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2), 2019.

[166] E. Farhi and H. Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018.

[167] I. H. Kim and B. Swingle. Robust entanglement renormalization on a noisy quantum computer. *arXiv preprint arXiv:1711.07500*, 2017.

[168] D. Wecker, M. B. Hastings, and M. Troyer. Training a quantum optimizer. *Physical Review A*, 94(2):022309, 2016.

[169] G. Verdon, J. Pye, and M. Broughton. A universal training algorithm for quantum deep learning. *arXiv preprint arXiv:1806.09729*, 2018.

[170] E. Farhi, J. Goldstone, and S. Gutmann. Quantum adiabatic evolution algorithms versus simulated annealing. *arXiv preprint quant-ph/0201031*, 2002.

[171] L. T. Brady and W. van Dam. Spectral-gap analysis for efficient tunneling in quantum adiabatic optimization. *Physical Review A*, 94(3):032309, 2016.

[172] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli. Convergence and finite-time behavior of simulated annealing. *Advances in applied probability*, 18(3):747–771, 1986.

[173] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[174] B. Hajek. Cooling schedules for optimal annealing. *Mathematics of operations research*, 13(2):311–329, 1988.

[175] B. Gidas. Nonstationary markov chains and convergence of the annealing algorithm. *Journal of Statistical Physics*, 39(1-2):73–131, 1985.

[176] S. Jansen, M.-B. Ruskai, and R. Seiler. Bounds for the adiabatic approximation with applications to quantum computation. *Journal of Mathematical Physics*, 48(10):102111, 2007.

[177] A. Elgart and G. A. Hagedorn. A note on the switching adiabatic theorem. *Journal of Mathematical Physics*, 53(10):102202, 2012.

[178] J. Roland and N. J. Cerf. Quantum search by local adiabatic evolution. *Physical Review A*, 65(4):042308, 2002.

[179] R. D. Somma, D. Nagaj, and M. Kieferová. Quantum speedup by quantum annealing. *Physical review letters*, 109(5):050501, 2012.

[180] R. Chakrabarti and H. Rabitz. Quantum control landscapes. *International Reviews in Physical Chemistry*, 26(4):671–735, 2007.

[181] Armin Rahmani and Claudio Chamon. Optimal control for unitary preparation of many-body states: Application to luttinger liquids. *Physical review letters*, 107(1):016402, 2011.

[182] M. Lapert, Y. Zhang, S. J. Glaser, and D. Sugny. Towards the time-optimal control of dissipative spin-1/2 particles in nuclear magnetic resonance. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 44(15):154014, 2011.

[183] B. Bonnard, S. J. Glaser, and D. Sugny. A review of geometric optimal control for quantum systems in nuclear magnetic resonance. *Advances in Mathematical Physics*, 2012, 2012.

[184] L. Kong and E. Crosson. The performance of the quantum adiabatic algorithm on spike hamiltonians. *International Journal of Quantum Information*, 15(02):1750011, 2017.

[185] B. W. Reichardt. The quantum adiabatic optimization algorithm and local minima. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 502–510, New York, NY, USA, 2004. ACM.

[186] Elizabeth Crosson and Aram W. Harrow. Simulated quantum annealing can be exponentially faster than classical simulated annealing. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 714–723, 2016.

[187] Jacob Bringewatt, William Dorland, Stephen P. Jordan, and Alan Mink. Diffusion monte carlo approach versus adiabatic computation for local hamiltonians. *Phys. Rev. A*, 97:022323, Feb 2018.

[188] Matthew B. Hastings. Obstructions to classically simulating the quantum adiabatic algorithm. *Quantum Info. Comput.*, 13(11–12):1038–1076, November 2013.

[189] P. Richerme, C. Senko, J. Smith, A. Lee, S. Korenblit, and C. Monroe. Experimental performance of a quantum simulator: Optimizing adiabatic evolution and identifying many-body ground states. *Phys. Rev. A*, 88:012334, Jul 2013.

[190] C. Kokail, C. Maier, R. van Bijnen, T. Brydges, M. K. Joshi, P. Jurcevic, C. A. Muschik, P. Silvi, R. Blatt, C. F. Roos, and P. Zoller. Self-verifying variational quantum simulation of lattice models. *Nature*, 569(7756):355–360, 2019.

[191] Edward Farhi and Aram W Harrow. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv:1602.07674*, 2016.

[192] Seth Lloyd. Quantum approximate optimization is computationally universal. *arXiv e-prints*, page arXiv:1812.11075, Dec 2018.

[193] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[194] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X*, 10:021067, Jun 2020.

[195] Zhang Jiang, Eleanor G. Rieffel, and Zhihui Wang. Near-optimal quantum circuit for grover's unstructured search using a transverse field. *Physical Review A*, 95(6), Jun 2017.

[196] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Physical Review A*, 97(2), Feb 2018.

[197] Gavin E Crooks. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv:1811.08419*, 2018.

[198] Glen Bigan Mbeng, Rosario Fazio, and Giuseppe Santoro. Quantum annealing: A journey through digitalization, control, and hybrid quantum variational schemes. *arXiv preprint arXiv:1906.08948*, 2019.

[199] Wen Wei Ho and Timothy H. Hsieh. Efficient variational simulation of non-trivial quantum states. *SciPost Phys.*, 6:29, 2019.

[200] Wen Wei Ho, Cheryne Jonay, and Timothy H. Hsieh. Ultrafast State Preparation via the Quantum Approximate Optimization Algorithm with Long Range Interactions. *arXiv:1810.04817*, Oct 2018.

[201] Thomas Koffel, M. Lewenstein, and Luca Tagliacozzo. Entanglement entropy for the long-range ising chain in a transverse field. *Physical Review Letters*, 109(26), Dec 2012.

[202] K. Kim, M.-S. Chang, R. Islam, S. Korenblit, L.-M. Duan, and C. Monroe. Entanglement and tunable spin-spin couplings between trapped ions using multiple transverse modes. *Physical Review Letters*, 103(12), Sep 2009.

[203] D. Porras and J. I. Cirac. Effective Quantum Spin Systems with Trapped Ions. *Phys. Rev. Lett.*, 92:207901, May 2004.

[204] Daniel Jaschke, Michael L. Wall, and Lincoln D. Carr. Open source matrix product states: Opening ways to simulate entangled many-body quantum systems in one dimension. *Computer Physics Communications*, 225:59 – 91, 2018.

[205] D Dylewsky, JK Freericks, ML Wall, AM Rey, and M Foss-Feig. Nonperturbative calculation of phonon effects on spin squeezing. *Physical Review A*, 93(1):013415, 2016.

[206] Stuart Hadfield. Quantum algorithms for scientific computing and approximate optimization. *arXiv:1805.03265*, 2018.

[207] Matteo M. Wauters, Glen Bigan Mbeng, and Giuseppe E. Santoro. Polynomial scaling of QAOA for ground-state preparation: Taming first-order phase transitions. *arXiv 2003.07419*, 2020.

[208] Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. *Phys. Rev. Lett.*, 117:080501, Aug 2016.

[209] Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. On the complexity and verification of quantum random circuit sampling. *Nat. Phys.*, 15(2):159–163, 2019.

[210] Sheng-Tao Wang and Lu-Ming Duan. Certification of Boson Sampling Devices with Coarse-Grained Measurements. *arXiv 1601.02627*, Jan 2016.

[211] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, September 2017.

[212] Cornelius Hempel, Christine Maier, Jonathan Romero, Jarrod McClean, Thomas Monz, Heng Shen, Petar Jurcevic, Ben P. Lanyon, Peter Love, Ryan Babbush, Alán Aspuru-Guzik, Rainer Blatt, and Christian F. Roos. Quantum chemistry calculations on a trapped-ion quantum simulator. *Phys. Rev. X*, 8:031022, Jul 2018.

[213] Yunseong Nam, Jwo-Sy Chen, Neal C Pisenti, Kenneth Wright, Conor Delaney, Dmitri Maslov, Kenneth R Brown, Stewart Allen, Jason M Amini, Joel Apisdorf, et al. Ground-state energy estimation of the water molecule on a trapped-ion quantum computer. *npj Quantum Information*, 6(1):1–6, 2020.

[214] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. Schuyler Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, Colm A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, Robert S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, Blake R. Johnson, M. Reagor, M. P. da Silva, and C. Rigetti. Unsupervised Machine Learning on a Hybrid Quantum Computer. *arXiv e-prints*, page arXiv:1712.05771, Dec 2017.

[215] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292(5516):472–475, 2001.

[216] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Advances in quantum metrology. *Nature Photonics*, 5:222, 03 2011.

[217] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[218] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[219] M. Jarret, S. P. Jordan, and B. Lackey. Adiabatic optimization versus diffusion Monte Carlo methods. *Phys. Rev. A*, 94:042318, 2016.

[220] Boaz Barak, Ankur Moitra, Ryan O'Donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, and John Wright. Beating the Random Assignment on Constraint Satisfaction Problems of Bounded Degree. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*, volume 40 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 110–123, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[221] Juho Hirvonen, Joel Rybicki, Stefan Schmid, and Jukka Suomela. Large cuts with local algorithms on triangle-free graphs. *The Electronic Journal of Combinatorics*, pages P4–21, 2017.

[222] A. Lucas. Ising formulations of many NP problems. *Front. Physics*, 2:5, 2014.

[223] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, April 2007.

[224] Samuel Burer, Renato DC Monteiro, and Yin Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM Journal on Optimization*, 12(2):503–521, 2002.

[225] Frauke Liers, Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. *Computing Exact Ground States of Hard Ising Spin Glass Problems by Branch-and-Cut*, chapter 4, pages 47–69. John Wiley & Sons, Ltd, 2004.

[226] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307, 2010.

[227] Z. Wang, A. Marandi, K. Wen, R. L. Byer, and Y. Yamamoto. Coherent Ising machine based on degenerate optical parametric oscillators. *Phys. Rev. A*, 88:063853, 2013.

[228] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber. Strengths and weaknesses of weak-strong cluster problems: A detailed overview of state-of-the-art classical heuristics versus quantum approaches. *Phys. Rev. A*, 94:022337, 2016.

[229] Angelika Wiegele. Binary Quadratic and Max Cut (Biq Mac) Library. `http://biqmac.uni-klu.ac.at/biqmaclib.html`, 2007.

[230] Francisco Barahona and Ali Ridha Mahjoub. On the cut polytope. *Mathematical programming*, 36(2):157–173, 1986.

[231] Wenceslas Fernandez de la Vega and Claire Kenyon-Mathieu. Linear programming relaxations of maxcut. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 53–61, USA, 2007. Society for Industrial and Applied Mathematics.

[232] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[233] Stephen P. Jordan, Keith S. M. Lee, and John Preskill. Quantum algorithms for quantum field theories. *Science*, 336(6085):1130–1133, 2012.

[234] Lucas T. Brady, Lucas Kocia, Przemyslaw Bienias, Aniruddha Bapat, Yaroslav Kharkov, and Alexey V. Gorshkov. Behavior of analog quantum algorithms, 2021.

[235] Jaime Sevilla and C Jess Riedel. Forecasting timelines of quantum computing. *arXiv preprint arXiv:2009.05045*, 2020.

[236] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. *arXiv:1412.6062*, Dec 2014.

[237] David Wineland, C. Monroe, W. M. Itano, D. Leibfried, B. E. King, and D. M. Meekhof. Experimental issues in coherent quantum-state manipulation of trapped atomic ions. *J. Res. Natl. Inst. Stand. Technol.*, 103:259–328, 1998.

[238] S. Olmschenk, K. C. Younge, D. L. Moehring, D. N. Matsukevich, P. Maunz, and C. Monroe. Manipulation and detection of a trapped yb^{+} hyperfine qubit. *Phys. Rev. A*, 76:052314, Nov 2007.

[239] Kenneth R. Brown, Aram W. Harrow, and Isaac L. Chuang. Arbitrarily accurate composite pulse sequences. *Phys. Rev. A*, 70:052318, Nov 2004.

[240] Anders Sørensen and Klaus Mølmer. Quantum computation with ions in thermal motion. *Phys. Rev. Lett.*, 82:1971–1974, Mar 1999.

[241] D. F. V. James. Quantum dynamics of cold trapped ions with application to quantum computation. *Applied Physics B*, 66:181, 1998.