Towards the First Practical Applications of Quantum Computers

by

Kevin J. Sung

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2020

Doctoral Committee:

Professor Christopher Peikert, Co-Chair
Adjunct Professor Yaoyun Shi, Co-Chair
Dr. Ryan Babbush, Google
Professor John P. Hayes
Professor Emeritus Kim Winick

Kevin J. Sung
kevjsung@umich.edu
ORCID iD: 0000-0001-6459-6374

To my family

Acknowledgments

This thesis was made possible by my supportive advisors Yaoyun Shi and Christopher Peikert, my friends, and my family. I would like to thank the Google AI Quantum team for hosting me while I performed much of the research in this thesis. I cannot express how lucky I feel to have had access to their state-of-the-art quantum computing hardware. The experiments on this hardware that I present in this thesis were the result of large team-wide collaborations.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Algorithms

List of Appendices

Abstract

Noisy intermediate-scale quantum (NISQ) computers are coming online. The lack of error-correction in these devices prevents them from realizing the full potential of fault-tolerant quantum computation, a technology that is known to have significant practical applications, but which is years, if not decades, away. A major open question is whether NISQ devices will have practical applications.

In this thesis, we explore and implement proposals for using NISQ devices to achieve practical applications. In particular, we develop and execute variational quantum algorithms for solving problems in combinatorial optimization and quantum chemistry. We also execute a prototype of a protocol for generating certified random numbers. We perform our experiments on a superconducting qubit processor developed at Google. While we do not perform any quantum computations that are beyond the capabilities of classical computers, we address many implementation challenges that must be overcome to succeed in such an endeavor, including optimization, efficient compilation, and error mitigation. In addressing these challenges, we push the limits of what can currently be done with NISQ technology, going beyond previous quantum computing demonstrations in terms of the scale of our experiments and the types of problems we tackle. While our experiments demonstrate progress in the utilization of quantum computers, the limits that we reached underscore the fundamental challenges in scaling up towards the classically intractable regime. Nevertheless, our results are a promising indication that NISQ devices may indeed deliver practical applications.

CHAPTER 1

Introduction

1.1 Quantum computing in the NISQ era

1.1.1 The promise of quantum computing

Many computational problems of great interest are difficult to solve using classical digital computers. One example of such a problem is the simulation of quantum mechanical systems such as molecules or arrangements of atoms on a lattice. The essential difficulty seems to be the fact that storing the full representation of a quantum state on a classical computer requires resources that scale exponentially in the size of the physical system being simulated. As a result, the properties of even moderately-sized molecules with a few dozens of atoms cannot be computed to high precision. This difficulty led Feynman to speculate in the early 1980s that simulating quantum systems could be performed efficiently using a *quantum* computer [1].

Despite the enormous scientific and industrial importance of the simulation of quantum systems, the developing theory of quantum computing remained largely a curiosity until 1994, when Shor discovered an efficient quantum algorithm for integer factorization [2]. An explosion of interest in the field ensued, leading to much progress since. In particular, Feynman's original vision of using quantum computers for quantum simulation was fleshed out, and today this is one of the most anticipated applications of quantum computers [3, 4, 5].

1.1.2 Quantum computing in the present and near future

The project of building a general-purpose quantum computer capable of realizing significant practical applications is still in its infancy. Quantum computers are far more susceptible to noise than classical computers, so some form of error-correction is likely required

to scale them up. While quantum error-correction is known to be possible, it incurs a significant overhead in the number of qubits required [6]. A fault-tolerant quantum computer capable of executing general quantum computations will probably not be built for years, if not decades. Nevertheless, the goal of building a quantum computer is actively being pursued by many institutions, corporate as well as academic, and significant hardware advances have been made. In particular, noisy intermediate-scale quantum (NISQ) computers are now coming online [7].

NISQ computers have 50 to a few hundred qubits and two-qubit gate error rates above 0.1% or so. The presence of errors severely limits the size of the quantum circuits that can be reliably executed using these devices; we can expect to execute perhaps a few thousand gates before the noise overwhelms any signal. Despite this limitation, these devices will be able to perform computations that are beyond the reach of classical computers [8]. A major open question is whether NISQ devices will be able to perform *useful* computations beyond the reach of classical computers, and more generally, whether they will have practical applications. It is unlikely that a computational advantage of NISQ devices over classical computers will be proven mathematically. Rather, we will discover the answer to this question by experimentation. As these devices become available, we will try things out and see what happens.

1.1.3 Challenges of NISQ computing

Any attempt to use NISQ computers for practical applications will have to overcome many challenges. Due to the extremely limited number of gates that can be reliably executed, circuits will need to be compiled very efficiently, taking into consideration the details of the specific hardware platform such as the available native gates and restrictions on qubit connectivity. For example, near-term superconducting qubit processors have qubits that are laid out on a planar grid, and two-qubit gates can be performed only between neighboring qubits. Because NISQ computers suffer from noise, they will not be able to execute moderately-sized circuits with high fidelity. Therefore, for applications that require high fidelity, some form of error mitigation will be required.

In addition to the quantum computing challenges presented by NISQ computers, there are also classical computing challenges. One of the leading candidates for NISQ applications is a class of hybrid quantum-classical algorithms in which a classical optimization algorithm is used to optimize a quantum objective function. The choice of this classical optimization algorithm can have a significant impact on performance.

In this thesis, we explore and implement proposals for using NISQ devices to achieve

practical applications. While we do not perform any quantum computations that are beyond the capabilities of classical computers, we address many implementation challenges that must be overcome to succeed in such an endeavor, including the challenges of efficient compilation, error mitigation, and optimization mentioned above. In addressing these challenges, we push the limits of what can currently be done with NISQ technology, going beyond previous quantum computing demonstrations in terms of the scale of our experiments and the types of problems we tackle. While our experiments demonstrate progress in the utilization of quantum computers, the limits that we reached underscore the fundamental challenges in scaling up towards the classically intractable regime. For most applications, reaching the classically intractable regime will require further innovations.

1.2 Overview of results

1.2.1 Using models to improve optimizers for variational quantum algorithms

One of the most promising potential applications of NISQ computers is in approximately solving difficult optimization problems in computer science, quantum chemistry, and other fields. Variational quantum algorithms are the major paradigm in which this application has been studied [9, 10]. In this paradigm, one devises a quantum circuit that depends on a number of parameters. For a fixed setting of the parameters, the quantum computer is used to execute the circuit, and some observables are measured. A classical optimization algorithm is used to suggest new parameters to try, with the goal being to minimize some cost function of the measured observables. The offloading of significant computational effort onto a classical computer and the fact that even small quantum circuits can generate highly entangled states that cannot be feasibly simulated make this paradigm especially suited to NISQ devices. The choice of classical optimization algorithm can have a significant impact on performance.

In Chapter 2, we introduce two methods for the classical optimization subroutine of variational quantum algorithms. The methods are surrogate model-based algorithms designed to improve reuse of collected data. They do so by utilizing a least-squares quadratic fit of sampled function values within a moving trusted region to estimate the gradient or a policy gradient. To make fair comparisons between optimization methods, we develop experimentally relevant cost models designed to balance efficiency in testing and accuracy with respect to cloud quantum computing systems. Our results underscore the need to both use relevant cost models and optimize hyperparameters of existing optimization methods

for competitive performance. We find that the methods we introduce perform well overall and have several practical advantages in realistic experimental settings.

1.2.2 Quantum approximate optimization on a superconducting qubit processor

In Chapter 3, we report the results of an experiment in which we executed the Quantum Approximate Optimization Algorithm (QAOA) [9] on a superconducting qubit processor. The QAOA is a variational quantum algorithm used to solve problems in combinatorial optimization. Like past QAOA experiments, we study performance for problems defined on the connectivity graph of our hardware; however, we also apply the QAOA to the Sherrington-Kirkpatrick model and 3-regular MaxCut, both high dimensional graph problems requiring significant compilation. Our experiments on these abstract problems that do not directly fit on our hardware are the largest such experiments to date.

Experimental scans of the QAOA energy landscape show good agreement with theory across even the largest instances studied (23 qubits) and we are able to perform variational optimization successfully using the optimizer introduced in Chapter 2. For problems defined on the planar graph of our hardware we obtain an approximation ratio that is independent of problem size and observe, for the first time, that performance increases with circuit depth. For problems requiring compilation, performance decreases with problem size but still provides an advantage over random guessing for circuits involving several thousand gates. This behavior highlights the challenge of using near-term quantum computers to optimize problems on graphs differing from hardware connectivity. As these graphs are more representative of real world instances, our results advocate for more emphasis on such problems in the developing tradition of using the QAOA as a holistic benchmark of quantum processors.

1.2.3 Preparing Slater determinants and fermionic Gaussian states

In Chapter 4, we describe an efficient algorithm for performing an essential primitive in quantum simulation, the preparation of Slater determinants. Our algorithm improves on prior art, and we generalize the construction to also give an algorithm for the preparation of fermionic Gaussian states. Our algorithms use only nearest-neighbor interactions on a linear array, making them well-adapted to the connectivity restrictions on current superconducting qubit architectures.

1.2.4 Hartree-Fock on a superconducting qubit processor

The Hartree-Fock method is a variational method for approximately solving the electronic structure problem. Importantly, it can be performed efficiently on a classical computer. It is central to classical and quantum electronic structure calculations and often serves as a starting point for more sophisticated methods.

In Chapter 5, we report the results of an experiment in which we implement the Hartree-Fock method as a variational quantum algorithm to simulate quantum chemistry with up to 12 qubits. The circuits we use are adapted from those given in Chapter 4 for the preparation of Slater determinants. We simulate linear chains of up to 12 hydrogen atoms, and also model the isomerization mechanism of diazene. We demonstrate error-mitigation strategies which dramatically improve results. For the hydrogen chains of lengths 6 and 8, we achieve chemical accuracy of 1.5 milli-Hartrees in computing the Hartree-Fock energy for various bond lengths, and for the larger systems, we still obtain qualitative agreement in the shape of the bond dissociation curves. In our simulation of diazene, we are able to resolve the energy difference between two competing isomerization mechanisms. Our experiments constitute the largest demonstration of solving quantum chemistry problems on quantum computers to date. However, even with our error-mitigation techniques, we were unable to scale to larger systems due to the stringent fidelity requirements needed to accurately compute molecular energies. This difficulty underscores a fundamental challenge in scaling towards the classically intractable regime for this application.

1.2.5 Generating certified random numbers on a superconducting qubit processor

In Chapter 6, we report our implementation of a prototype of a protocol for generating certified random numbers on NISQ devices. Certified random number generation produces random numbers whose randomness, or unpredictability, can be certified to a skeptical client. They have several applications, including the generation of cryptographic keys and the implementation of lotteries. Because the laws of classical physics are deterministic, certified randomness can only be generated using quantum processes.

The certified randomness protocol we implement is based on an unpublished proposal by Scott Aaronson. Unlike previous proposals for certified randomness [11], the validity of our protocol depends on a computational hardness assumption regarding the task of sampling from the output distributions of quantum circuits such as those used in the demonstration of quantum supremacy [8]. In our implementation, we focused on the exercise of the software infrastructure needed to run this protocol as a service accessed through

the Internet. In particular, we used fully automatic calibration of our quantum processor, a procedure that suffers from some temporary limitations which prevent us from achieving experimental parameters that would provide true certification of randomness. Nevertheless, our results demonstrate a proof of concept.

1.3 Outline of thesis

In Chapter 2, we introduce two model-based optimizers for variational quantum algorithms and give the results of numerical benchmarks comparing them with alternative optimizers. In Chapter 3, we report the results of an experimental implementation of the QAOA on a superconducting processor, including the results of variational optimization using the optimizer introduced in Chapter 2. In Chapter 4, we give efficient algorithms for preparing Slater determinants and fermionic Gaussian states on a quantum computer. In Chapter 5, we report the results of an experimental implementation of a variational quantum algorithm that realizes the Hartree-Fock method for simulating quantum chemistry on a superconducting processor, using circuits adapted from the algorithms given in Chapter 4. In Chapter 6, we report the implementation of a prototype of a protocol for generating certified random numbers on a superconducting processor.

1.3.1 Works appearing

The work in Chapter 2 is contained in [12] and has been submitted for publication. The work in Chapter 3 is contained in [13] and has been submitted for publication. The work in Chapter 4 is contained in [14]. The work in Chapter 5 is contained in [15] and has been accepted for publication in *Science*. The work in Chapter 6 is ongoing at the time of writing and will be found in [16].

The works [13], [15], and [16] were large collaborations that involved performing experiments on quantum computing hardware. For all of these works, the author of this thesis wrote significant parts of the software used to execute the experiment, including, but not limited to, the optimizer used in [13]. For the works [13] and [16], the author also participated in the design of the experiments and collected and analyzed significant portions of the experimental data. In [12], the author appears as first author, but benefited greatly from discussions with collaborators. In [14], the author appears as second author, and Chapter 4 only presents a portion of that work to which the author made major contributions.

Other works to which the author has contributed as a graduate student, but which have not been included in this thesis, can be found in [8, 17, 18, 19].

CHAPTER 2

Using Models to Improve Optimizers for Variational Quantum Algorithms

2.1 Introduction

With recent developments in quantum hardware, including the ability to perform select tasks faster than classical supercomputers [8], the push towards practical applications on these devices has intensified. Variational quantum algorithms are among the top candidates for early applications on noisy intermediate-scale quantum (NISQ) computers [20, 10, 7]. These algorithms can be used to approximate ground energies of Hamiltonians or find approximate solutions to discrete optimization problems. A main component of these algorithms is the minimization of some function of a parameterized quantum state, where that function is measured using the quantum computer. Commonly, the function is the expectation value of a Hamiltonian, determined by the problem of interest. The presence of sampling error and gate errors makes the function stochastic, and the stochasticity due to sampling error is fundamental to measuring the values on a quantum device. The output of this stochastic function is fed to a classical optimizer, and it is those optimizers and constraints presented by real devices that we will focus on here.

As the classical optimizers are at the core of variational quantum algorithms, their performance can determine the resources required to solve a problem. Non-linear optimization of continuous functions of the type that exist in variational quantum algorithms are commonplace in fields like machine learning, but quantum systems offer unique trade-offs that must be considered to improve efficiency. Given the current focus on these algorithms and the core role played by the optimizer, there have been a number of works evaluating the performance of optimizers for different problems and contexts. For example, at least two experimental implementations of variational algorithms [20, 21] used the Nelder-Mead simplex algorithm [22] to optimize the objective function. Other experimental implementations [23, 24, 25, 26, 27] used algorithms including Simultaneous Perturbation

Stochastic Approximation (SPSA) [28], Bayesian optimization [29], particle swarm optimization [30], dividing rectangles [31], and gradient descent. In addition, there have been a number of numerical investigations of optimization in the context of variational quantum algorithms. Several of these studies introduce novel heuristics and test them numerically on example problems [32, 33, 34, 35, 36, 37]. Other work [38, 39, 40, 41, 42, 43] has compared the performance of methods including Nelder-Mead, limited-memory Broyden-Fletcher-Goldfarb-Shanno, [44], Constrained Optimization By Linear Approximation [45], Powell's method [46], SPSA, RBFOpt [47], Stable Noisy Optimization by Branch and Fit [48], Bound Optimization by Quadratic Approximation [49], Mesh Adaptive Direct Search [50], implicit filtering [51], and policy-gradient-based reinforcement learning [52].

There is a considerable body of work in evaluating optimizers for use in variational algorithms, but not all of these works use cost metrics relevant to quantum experiments. For example, it is common to evaluate a suite of optimizers based on the number of optimizer iterations required for convergence to a local optima, using noiseless function evaluations. However, the inherent quantum nature of the sampling procedure implies that the first iteration could have taken an unbounded amount of experimental time in such a setup (noiseless evaluation), and hence conclusions based on such studies may not be applicable to experiments. A meaningful comparison of these methods must treat the stochastic nature of the objective function and related costs in terms of experimental time to solution to properly compare methods. While some past works do account for the effect of stochastic noise [36, 37], in this work we additionally incorporate other experimental parameters into our cost models. In developing our models, we focus on the case of superconducting quantum computers accessed through the Internet, though our models can be easily modified for other architectures. We account for parameters such as the sampling rate of the quantum processor and the latency induced by communicating over the Internet. The proper choice of optimizer ultimately depends on the details of the experiment constraints.

In consideration of constraints we did not find satisfied in other methods, we introduce two surrogate model-based optimization algorithms we call Model Gradient Descent (MGD) and Model Policy Gradient (MPG) and numerically compare their performance against commonly used methods. In particular, we target the tendency for local methods to under-utilize the existing history of function evaluations. We have successfully used MGD in an experimental implementation of the Quantum Approximate Optimization Algorithm [9] on a superconducting qubit processor; this experiment will be described in Chapter 3. We perform systematic tuning of optimizer hyperparameters before comparison for all methods, and measure performance using estimates of actual wall clock time needed in a realistic experimental setting. An important, though unsurprising, implication of our re-

sults is that hyperparameter tuning under the correct cost models is crucial for performance in practice. Our results also highlight the importance of different cost model features and how constraints influence the optimal choice of optimizer. Stochastic optimizers that incorporate randomness found to be generally more robust and efficient.

2.2 Background

A ubiquitous problem in physics and chemistry is to find the lowest eigenvalue and eigenvector of a Hamiltonian H describing a physical system. The Hamiltonian is a Hermitian matrix that determines how the system evolves in time via the Schrödinger equation, and its eigenvalues are often referred to as energies. Typically the Hamiltonian can be written as a sum of terms that each act on only part of the system,

$$H = \sum_{j=1}^{m} H_j. \tag{2.1}$$

For example, each H_j could be a tensor product of a few Pauli operators with the identity. The problem of finding the lowest eigenvalue and eigenvector of H is called the ground state problem. Often it is sufficient to calculate just the eigenvalue, and the formalization of this problem as a decision problem is QMA-complete [53]. Nevertheless, practical instances of the problem may admit approximate solution. Note that the ground state problem can also encode combinatorial optimization problems that do not have an obvious physical interpretation.

The lowest eigenvalue of H is called the ground state energy, and it can be expressed as

$$E_0 = \min_{|\psi\rangle} \langle\psi|H|\psi\rangle \tag{2.2}$$

where the minimization is over all normalized quantum states. The variational method in quantum mechanics is a method of finding an approximation to E_0. It starts with a parameterization $|\boldsymbol{\theta}\rangle$ of a set of quantum states, where $\boldsymbol{\theta}$ is a list of real numbers that is allowed to vary, yielding different states. A parameterization of a set of quantum states is called an *ansatz*. The variational method works by minimizing the energy over the ansatz:

$$E_0 \approx \min_{\boldsymbol{\theta}} \langle\boldsymbol{\theta}|H|\boldsymbol{\theta}\rangle. \tag{2.3}$$

The actually calculated value will be somewhat greater than the true ground state energy

E_0.

In a variational quantum algorithm, the ansatz state $|\boldsymbol{\theta}\rangle$ is prepared on a quantum computer, and the expectation value $\langle\boldsymbol{\theta}|H|\boldsymbol{\theta}\rangle$ or its derivatives are estimated with measurements. A classical optimization algorithm is used to suggest new parameters, and this process is iterated with the goal of finding parameters that minimize the expectation value.

Variational quantum algorithms are a promising candidate for execution on NISQ computers because they make frugal use of quantum resources. In particular, the ansatz state $|\boldsymbol{\theta}\rangle$ can be chosen so that it can be implemented with the small number of gates that will be feasible to execute on NISQ computers. One goal in the NISQ era will be to prepare ansatz states that cannot be simulated on a classical computer. While we do not know whether the preparation of such states will yield a computational advantage over classical algorithms, we will try it and see how it performs.

2.3 Problems studied and cost models

2.3.1 Problems studied

As the performance of an optimizer can be intimately tied to the problem studied, it is important to look at a range of problems in evaluating their relative performance. As two of the most common areas studied in variational quantum algorithms are combinatorial optimization and ground state preparation of fermionic systems, we select these for our sample problems. Here we aim to clarify the details of the systems, circuit ansatze, and initial parameters modeled in our numerical tests.

While multi-modality of cost functions is an important consideration in variational quantum algorithms, it turns out that even optimization within a single convex basin can be challenging enough to warrant independent investigation due to constraints imposed by the quantum device. To this end, we assume throughout that we have knowledge of an initial guess which is in the convex vicinity of an optimum and our goal is simply to converge to that local optimum. Several strategies have been proposed for choosing such an initial guess in contexts including optimization and chemistry [33, 54, 39, 32].

2.3.1.1 Max-Cut on 3-regular graphs

The maximum cut problem (Max-Cut) is widely studied and known to be NP-hard. The problem is specified by an undirected graph on n vertices and the goal is to label each vertex with either 0 or 1 in order to maximize the number of edges whose vertices have

different labels. This cost function is represented by the Hamiltonian

$$C = \sum_{\langle i,j \rangle} \frac{1}{2}(I - Z_i Z_j), \tag{2.4}$$

where Z_j is the standard Pauli Z operator applied to qubit j which is node j on the graph, and $\langle i, j \rangle$ ranges over the edges of the graph. The goal is to find a computational basis state that *maximizes* the Hamiltonian.

We use the Quantum Approximate Optimization Algorithm (QAOA) [9] ansatz used to approximately solve the Max-Cut problem on random 3-regular graphs. The QAOA ansatz depends on the number of rounds, $p > 0$, and is parameterized by $2p$ real numbers $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$. The ansatz is

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U_B(\beta_p) U_C(\gamma_p) \cdots U_B(\beta_1) U_C(\gamma_1) |+\rangle^{\otimes n}, \tag{2.5}$$

where

$$U_C(\gamma) = e^{-i\gamma C}, \quad U_B(\beta) = e^{-i\beta B}, \quad B = \sum_{i=1}^{n} X_i, \tag{2.6}$$

and $|+\rangle^{\otimes n}$ is the uniform superposition of all 2^n computational basis states.

For our numerics, we focus on a randomly chosen instance to minimize the number of uncontrolled variables. Moreover, for QAOA focusing on a single instance is justified because the optimization landscape has been shown to concentrate for different randomly chosen instances [54]. To obtain an initial guess for this problem, we classically computed a locally optimal parameter vector and then perturbed it with a uniformly random vector of length 0.1. At $p = 1$ the optimal parameter vector had a length of 0.462, and at $p = 5$, 1.285.

In our numerics we report the approximation ratio

$$\frac{\langle \boldsymbol{\gamma}, \boldsymbol{\beta} | C | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle}{C_{\max}} \tag{2.7}$$

where $C_{\max} = \max_z \langle z | C | z \rangle$. The goal is to maximize this value, which falls in the range $[0, 1]$.

2.3.1.2 Sherrington-Kirpatrick model

Another model we consider is the Sherrington-Kirkpatrick (SK) model [55], which is a canonical example of a frustrated spin glass. The Hamiltonian is given by

$$H = \sum_{i<j} J_{ij} Z_i Z_j \tag{2.8}$$

where J_{ij} is selected uniformly at random from $\{-1, 1\}$. We use the QAOA ansatz to approximate the solution of this problem, by minimizing the expected cost.

Again, for our numerics we focus on a single randomly generated instance, where generality of performance is supported by concentration results in QAOA. As an initial guess for this problem, we classically computed a locally optimal parameter vector and then perturbed it with a uniformly random vector of length 0.1. At $p = 1$ the optimal parameter vector had a length of 0.452, and at $p = 5$, 1.044.

For comparison between problems, we normalize energy values E to new values E' by the formula

$$E' = \frac{E - E_{max}}{E_{min} - E_{max}} \tag{2.9}$$

where E_{min} and E_{max} are the lowest and highest eigenvalues of the Hamiltonian, respectively. Thus we are in fact maximizing this normalized energy value, which falls in the range $[0, 1]$.

2.3.1.3 Hubbard model

We study the task of approximating the ground state energy of the 2-dimensional Hubbard model [56], a widely studied model that has resisted exact solution for decades in large size limits. It is believed to be relevant to understanding high-temperature superconductivity [57]. The Hamiltonian of the Hubbard model is

$$\begin{aligned} H &= -t \sum_{\langle i,j\rangle,\sigma} (a^\dagger_{i,\sigma} a_{j,\sigma} + a^\dagger_{j,\sigma} a_{i,\sigma}) \\ &\quad + U \sum_i a^\dagger_{i,\uparrow} a_{i,\uparrow} a^\dagger_{i,\downarrow} a_{i,\downarrow} && (2.10) \\ &= T + V && (2.11) \\ &= T_h + T_v + V && (2.12) \end{aligned}$$

where the $a_{i,\sigma}$ are fermionic annihilation operators, $\langle i, j\rangle$ ranges over edges in the lattice, $\sigma \in \{\uparrow, \downarrow\}$ is a spin degree of freedom, and we have split the sum into the hopping term T and interaction term V. T is further decomposed into sub-terms T_h and T_v corresponding to horizontal and vertical edges, respectively. We set $t = 1$ and $U = 4$ for our numerical experiments, which corresponds to a regime of modest correlation ill-suited for mean-field methods.

We use a "Hamiltonian variational" ansatz similar to the one in ref. [32]. It is inspired by the idea of state preparation via adiabatic evolution. Similar to QAOA, our ansatz has a basic circuit repeated p times, but for flexibility it is varied non-uniformly with respect to hopping. The basic circuit has three parameters which we call θ_h, θ_v, and θ_U, and it approximates a unitary of the form

$$\exp[-i(\theta_h T_h + \theta_v T_v + \theta_U V)] \tag{2.13}$$

The approximation is achieved using a second-order Trotter step based on the fermionic swap network [58], in which a swap network is used to apply the terms of the Hamiltonian and then the same network is applied but in reverse order. This is similar to the ansatz used in ref. [32] but corresponds to a different ordering of terms. In total there are $3p$ parameters.

We study the model at half-filling. Our numerics are performed on the 2×2 system, which under standard encodings corresponds to an 8 qubit system. For our initial state we use a ground state of the hopping term that is precisely described in Appendix A. This state is easy to prepare on a quantum computer and is expected to be adiabatically connected to the ground state of H for modest values of t/U. For our initial guess, we set the parameters so that the ansatz circuit consists of a sequence of second-order Trotter steps approximating the dynamics of the time-dependent Hamiltonian $H(t) = T + (t/A)V$ for $t \in [0, A]$, where $A = 0.1 \cdot Up$. This choice is motivated by the idea of state preparation via adiabatic evolution.

As with the Sherrington-Kirkpatrick model, we normalize energy values E to new values E' by the formula

$$E' = \frac{E - E_{max}}{E_{min} - E_{max}} \tag{2.14}$$

where E_{min} and E_{max} are the lowest and highest eigenvalues of the Hamiltonian, respectively. Thus we are in fact maximizing this normalized energy value, which falls in the range $[0, 1]$.

2.3.2 Cost models

An essential element of developing and improving optimizers for variational algorithms is an accurate cost model that respects the quantum nature of the problem and imperfections of the device. Studies that restrict evaluation of optimizers to abstract "number of iterations" using perfect function queries can yield faulty conclusions and hide the implication that a single function evaluation to that precision could have taken years or more. A core challenge is the stochastic nature of the function evaluation and shot limited precision in the estimates. Moreover, imperfections in the device and implementation can complicate matters. Unfortunately, without a quantum device, precise simulation of the impact of noise can be prohibitively expensive, and so a balance must be struck between accuracy and cost effectiveness of the simulations to maximize applicability. Here we detail how we construct our models to strike this balance.

We restrict our interest to minimizing the expected energy of a Hamiltonian H with efficient Pauli expansions $H = \sum_j \alpha_j P_j$ (in the case of the Hubbard model (2.10), the Jordan-Wigner Transformation [59] is applied to obtain the Pauli expansion), so the objective function is

$$f(\boldsymbol{\theta}) = \langle \boldsymbol{\theta} | H | \boldsymbol{\theta} \rangle, \tag{2.15}$$

where $|\boldsymbol{\theta}\rangle$ represents the ansatz state with parameters $\boldsymbol{\theta}$. Most of the optimizers that we present results for use queries to the objective function without any additional kinds of queries, but we also present results for stochastic gradient descent, which queries the gradient.

2.3.2.1 Objective function queries

The exact estimator used to query the objective function on the quantum device can take a wide variety of forms depending on factors in the device and the problem of interest. At a glance, however, a query to the objective function is often answered by measuring the expectation values of the terms P_j and using the coefficients α_j to form an estimate of $f(\boldsymbol{\theta})$. When simulated in the most accurate way, the measurement of each individual term implies a variance on the estimate which is state-dependent, and functions like a Bernoulli random variable. Moreover, the variance of that measurement can be influenced by parallel measurements being performed, even when they commute [10]. Trade-offs in the influences of these factors have inspired recent research in developing more efficient estimators with a given number of samples [60, 61, 62, 63, 64]. However, perfect emulation of these proposals can be prohibitively expensive, even in classical simulation of small systems, and

hence it is desirable to develop models of the process that strike a good balance between accuracy and simulation cost so that the full variational process can be simulated on a range of systems.

In the cases of Max-Cut and the Sherrington-Kirkpatrick model, the Hamiltonian is diagonal and all of its terms can be measured simultaneously in one shot. In our numerical experiments, we simulated these measurements directly. However, for non-diagonal Hamiltonians such as the Hubbard model, we take a different strategy.

As there are many terms in the sum, which are typically evaluated by repeated and independent measurement, a Gaussian random function query turns out to be a good and extremely cost effective model. That is, in our simulations a query to the objective function is modeled as

$$f(\boldsymbol{\theta}) = \langle \boldsymbol{\theta}|H|\boldsymbol{\theta}\rangle + \mathcal{N}(0, \lambda^2/M) \tag{2.16}$$

$\langle \boldsymbol{\theta}|H|\boldsymbol{\theta}\rangle$ is evaluated exactly, $\mathcal{N}(\mu, \sigma^2)$ is a normal random variable with mean μ and variance σ^2, and M is the number of repeated experiment repetitions. Here, the variance is estimated using a known lower bound for common measurement strategies, previously derived for the general case

$$\lambda^2 = (\sum_j |\alpha_j|)^2 \tag{2.17}$$

which empirically we have observed to be loose when compared with exact models, but qualitatively matches the behavior and overestimates the number of measurements by a factor of 2 in many cases. We note that a wealth of other strategies have been developed to shrink the effective variance for a fixed number of queries M [60, 61, 62, 63, 64], but we do not consider them in detail here. The dependence of the variance of the estimate on the number of samples represents a key trade-off we consider in many algorithms here, as some optimizers can tolerate heavier amounts of noise than others, and hence we take the number of shots at each iterate to be an important hyperparameter. In our numerical experiments on the Hubbard model, we simulated queries by computing the exact expectation value and then artificially adding noise drawn from a normal distribution, using this bound to determine the variance of the distribution for a specified number of measurement shots.

2.3.2.2 Gradient queries

For optimizers that use analytic gradient queries, we assume that queries to the gradient of the objective function are answered by applying the "parameter-shift rule" [65, 66, 67].

This is a method of obtaining an unbiased estimator of the gradient without using ancilla qubits, and applies to ansatze of the form

$$|\boldsymbol{\theta}\rangle = \exp(-i\theta_p A_p)\cdots\exp(-i\theta_1 A_1)|\psi\rangle \tag{2.18}$$

where for our purposes each A_j is a Hermitian sum of commuting Pauli matrices. The technique exploits the fact that if A_j has two eigenvalues $\pm r$, then $\frac{\partial f}{\partial \theta_j}(\boldsymbol{\theta}) = r(f(\boldsymbol{\theta}^+) - f(\boldsymbol{\theta}^-))$ where $\boldsymbol{\theta}^+$ is $\boldsymbol{\theta}$ but with the j-th coordinate equal to $\theta_j + \frac{\pi}{4r}$ and $\boldsymbol{\theta}^-$ is $\boldsymbol{\theta}$ but with the j-th coordinate equal to $\theta_j - \frac{\pi}{4r}$. If some parameters are constrained to be the same, then the derivative is obtained by summing the results of this expression for each parameter; the number of objective function queries needed is then two times the number of those parameters. If $A_j = \sum_k P_k$ for commuting Pauli operators P_k, then we decompose $\exp(-i\theta_j A_j) = \prod_k \exp(-i\theta_j P_k)$ and then apply the previous rule. Thus, the cost of evaluating the partial derivative is proportional to the number of terms in the sum, in a loose way. In practice, this sum is evaluated stochastically with a probability depending on the weight of the term in the sum [68].

2.3.2.3 Wall clock time

Ultimately, one is interested in minimizing the amount of time it takes to run a complete experiment to some fixed precision. The models we develop here are meant to capture this in a cost efficient way, without using a wildly inaccurate proxy like mere "number of optimizer iterations". To this end, we not only consider the sampling noise, but also constraints like latency concerns inherent to real experiments.

To estimate the running time of an experiment we develop a model based on superconducting qubits [69, 70]. We also assume the user is executing the experiment through a cloud computing service, potentially introducing network latency. We consider three scenarios regarding network latency: zero latency, corresponding to the optimizer running completely on the server side; circuit batching, in which the user is allowed to send multiple circuits to the service in one batch; and finally no circuit batching, where the user is only allowed to send one circuit at a time.

The total running time of an experiment is equal to the number of queries made times the amount of time it takes to satisfy a single query. The time needed to satisfy a single query can be split into the time T_{sample} used in sampling circuits on the quantum processor, the time T_{switch} representing the overhead in switching between different circuits, and T_{cloud} representing the latency in communicating over the Internet. We have $T_{\text{sample}} = M/s$ where M is the number of measurements made to satisfy the query and s is the sampling rate of

the processor; $T_{\text{switch}} = r \times c$ where r is the overhead in readying the quantum processor to execute a circuit and c is the number of different circuits executed; and $T_{\text{cloud}} = \ell \times c/b$ where ℓ is the network round-trip time for communicating with the cloud server and b is the number of circuits sent to the server in a single round of communication. We use the values $s = 10^5$ Hz and $r = 0.1$ s. This sampling rate has not yet been achieved experimentally but is plausible assuming an order of magnitude or two improvement in current capabilities is possible; a recent experiment achieved a sampling rate of about 5×10^3 Hz [8]. When including network latency, we set $\ell = 4.0$ s; this value is based on our own experience executing experiments through an internal cloud interface. The value of b depends on the details of the algorithm. We ignore as negligible the time taken by the classical optimization algorithm to select parameters for querying, as the optimizers here use relatively simple classical updates.

2.4 Optimization strategies

2.4.1 Choice of optimizers

A wide range of optimizers now exist for continuous, non-linear optimizations, with different strengths and weaknesses. One key element for consideration is the stochastic nature of our objective function and its relation to the number of measurements made for each function evaluation. Some optimizers were designed with noiseless (up to reasonable precision limits) function evaluations in mind, and are relatively unstable with respect to even small amounts of noise. While one could insist on a number of measurements that renders the function evaluations essentially exact, this incurs a huge overhead per iteration. We group algorithms into two categories, distinguished by whether they have inherent hyperparameters that allow them to adjust their resilience to noise. If an algorithm in practice requires that the input be given to a fixed precision in order to be stable, we term it deterministic. If it has a hyperparameter that naturally allows it to accept more or less noise, we call it stochastic.

The difference between the two classes can be subtle, and depend on the details of implementation. For example, a gradient descent implementation that makes use of an exact line search can accidentally rule out good regions of space from small wobbles in a query value, and is hence deterministic. However, if that sample implementation substitutes a fixed step with a learning rate, it is not only more robust to noise, but that learning rate can be adjusted to match noise levels in the objective queries. Hence we term that a stochastic optimizer. Considering the costs of each with external hyperparameters (e.g. number of

measurements) and internal hyperparameters (e.g. learning rate) tuned for optimal performance will show us these trade-offs.

Overall, we investigated six different optimizers. Four of these have been studied in past work, and the last two are surrogate model-based optimizers that we introduce here. Surrogate model-based optimizers construct a model of the objective function using previously evaluated points and use the model to determine what points to evaluate next. They are popular choices for the optimization of objective functions that are expensive to evaluate or noisy (or both) [71, 72].

Listed briefly, the optimizers we study here are:

- Deterministic algorithms:
  - The Nelder-Mead simplex method [22]. This method has been used in previous theoretical [38, 39] and experimental [20, 21] works on variational algorithms. We used the implementation from SciPy [73].
  - Bounded Optimization By Quadratic Approximation (BOBYQA) [49]. This is a surrogate model based algorithm that uses an interpolating quadratic model to approximate the objective function, and has been studied in a previous work on variational algorithms [42]. We used the implementation from the Python package Py-BOBYQA [72].
- Stochastic algorithms:
  - Simultaneous Perturbation Stochastic Approximation (SPSA) [28]. This method has also been used in previous theoretical [40] and experimental [23] works on variational algorithms. We used our own implementation.
  - Stochastic gradient descent using analytic gradient measurements obtained via the "parameter-shift rule" [65, 66, 67].
  - Model Gradient Descent (MGD). This is a surrogate model-based algorithm we introduce here that uses a least-squares quadratic model to estimate the gradient of the objective function. We give pseudocode in Appendix B.
  - Model Policy Gradient (MPG). Building on the vanilla policy gradient method [41], this method additionally introduces a least-squares quadratic model to reduce variance in the estimation of the policy gradient. We give pseudocode in Appendix B.

2.4.2 Model Gradient Descent and Model Policy Gradient

In this section we describe and motivate the design choices of our new algorithms, Model Gradient Descent and Model Policy Gradient, which are described in pseudocode in Algorithm B.1 and B.2. These are surrogate model-based methods which use least-squares regression to fit quadratic models of the objective function. A key expense in variational quantum algorithms is the evaluation of the function at different points, which is costly due to the underlying variance. Hence, it would be beneficial to reuse the history of point evaluations, rather than to discard them at each iteration. For local optimizations where iterates proceed gradually, it seems intuitive that this should be possible. Eventually, if one collected enough points in a small enough region, it should be possible to construct a surrogate model that is more accurate than raw function evaluations at a fixed number of measurements.

As a combination of this motivation and simplicity, we use a least-squares fit to a quadratic function. However, it is also clear that if the region of sampled points is too large, the function may not be well approximated by a quadratic, hence we use a trusted region of sample points, which may be new or reused from previous iterates.

In each iteration, the algorithms sample a number of points randomly from the vicinity of the current iterate. They fit quadratic models to these points and other previously evaluated points within the vicinity. Finally, MGD uses the gradient of this quadratic model as an approximation to the true gradient and performs gradient descent; MPG queries the model to evaluate a large batch of data points and performs policy gradient optimization. The reason we did not use standard trust-region solution techniques after building the quadratic model is that we found empirically that the quadratic model we built upon stochastic function evaluations had slightly negative Hessian eigenvalues, which dictates in a standard trust region solution method that the solution is on the exterior of the trust region. This constant jumping to the exterior of the trust region represented a sort of fundamental inefficiency under stochastic functions. In contrast, the gradient or policy gradient of the model, while stochastic, represented a reliable estimator that in conjunction with techniques like a fixed learning rate, combined the increased accuracy of additional samples with the robustness of a stochastic gradient descent.

To enhance the performance and stability of the methods, we introduced several hyperparameters to our algorithms. In particular, as algorithms approach an optimum, decreasing the radius of the neighborhood from which points are sampled is expected to give a more accurate estimate of the function value and its gradient. Thus, we introduce a hyperparameter ξ for MGD which controls the rate at which the radius decreases. As for MPG, we introduce the fixed sample radius ratio δ_r with respect to the maximal sample radius of

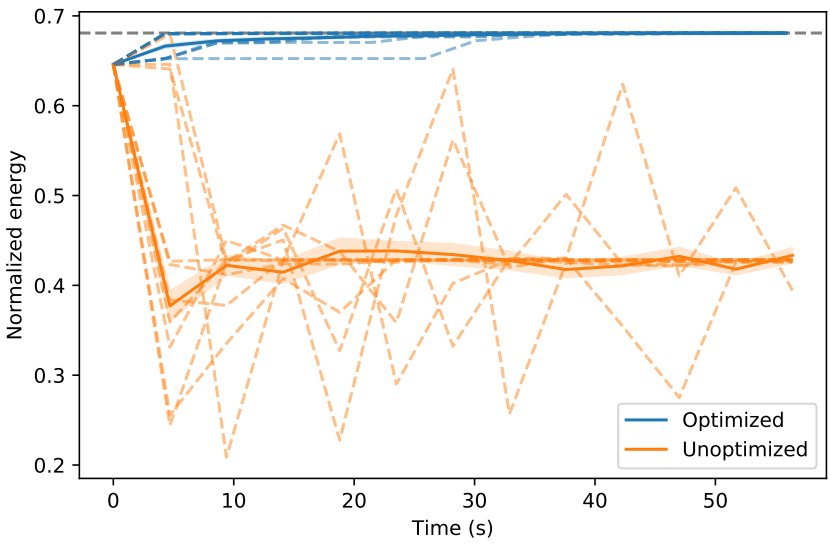


Figure 2.1: Optimization progress of SPSA in simulated experiments on a Sherrington-Kirkpatrick model Hamiltonian using two different hyperparameter settings: the ones used by default in the implementation from the software package Qiskit (Unoptimized), and ones that were found by searching for good settings (Optimized). The solid line represents the mean energy over 50 runs with different PRNG seeds, and the shaded region represents a width of one standard deviation of the mean. The dotted lines are 10 example trajectories. The dotted gray line corresponds to the ansatz optimum. SPSA fails to converge with the unoptimized hyperparameters.

the policy. The selected sample radius adaptively shrinks along with the maximal sample radius as the policy gradually becomes more confident. It may also be advantageous to decrease the learning rate of both algorithms. Thus, we introduce hyperparameters α and A which control the rate of this decrease. The details of how these parameters enter can be found in the pseudocode of the algorithms.

2.4.3 Hyperparameter selection

Each optimizer we considered here has a number of hyperparameters, and empirically we noted that the choice of these hyperparameters had a great impact on performance. Strikingly, some optimizers that failed completely with out of the box settings became competitive choices with even slight adjustments. Recalling that many of the optimizers we consider are inherently deterministic, one important hyperparameter external to all methods is the number of measurement shots per energy evaluation.

We tuned hyperparameters by grid search, and separately for each problem class and ansatz depth considered. For each combination of hyperparameters considered in the

search, we performed an optimization run using the wall clock time model that includes network latency and circuit batching. The optimal hyperparameters were those that minimized time to convergence with a precision target of 10^{-3}. To avoid effects of overfitting, we restricted consideration to single realizations, where other runs are not further optimized within a problem class. Note that the details of hyperparameter selection has a significant effect on the performance of the algorithms. For example, choosing a more lenient precision requirement while still minimizing time to solution leads to different performance characteristics on other problems. See Appendix C for more details, including descriptions of the hyperparameters.

As a simple demonstration of the importance of hyperparameter selection, we considered the performance on a simple test case with two different hyperparameter settings. Figure 2.1 shows the optimization progress of SPSA in simulated experiments on a Sherrington-Kirkpatrick model Hamiltonian with $n = 8$ and $p = 1$, using two different hyperparameter settings: the ones used by default in the implementation from the software package Qiskit [74], and ones that we optimized for minimal time to solution with a fixed precision cutoff. Depicted is the normalized energy versus wall clock time, using the wall clock time model that includes network latency and circuit batching. With tuned hyperparameters, SPSA converges to the solution rapidly, and without tuning it quite obviously does not. The erratic trajectory when using the unoptimized default parameters can be attributed to the fact that the initial learning rate of the algorithm is set to a value over 100 times larger than the optimized value. Hence, while SPSA is a powerful stochastic method capable of dealing with variable function noise, this flexibility must be actively used to make a proper comparison. Not taking advantage of this capability has led previous studies to conclude that SPSA is not effective for these problems. This demonstrates the importance of optimizing hyperparameters in making a fair comparison between optimization algorithms, and throughout this study we make an effort to tune all methods under consideration.

2.5 Results

To increase the applicability of our results to experiment, we consider both ideal and faulty operation of a quantum device. In the first case, in order to isolate challenges pertaining only to sampling noise, we assume an ideally functioning quantum computer, so that the only source of stochasticity in the objective function is finite sampling effects. In the other case, we modeled the effect of gate rotation error as follows: each time the optimizer queries the point $\boldsymbol{\theta}$, the objective function is evaluated at the point $\boldsymbol{\theta} + \boldsymbol{\varepsilon}$ instead, where each

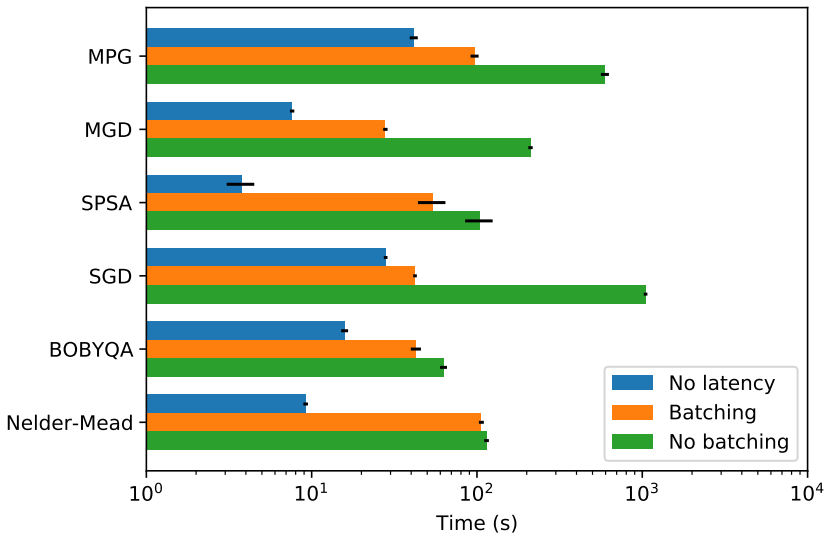


Figure 2.2: Wall clock time for optimization to achieve precision 1e-3 for the Sherrington-Kirkpatrick model at $p = 1$. Times are averaged over 50 experiments with different PRNG seeds. The black lines at the tips of the bars represent a width of one standard deviation. The best choice of optimizer can depend on the wall clock time model, with MGD, MPG, and SGD benefiting greatly from the ability to request execution of a batch of circuits.

component of $\boldsymbol{\epsilon}$ is chosen from the normal distribution with mean 0 and standard deviation ε (for some gate error level ε). Since this model does not straightforwardly translate to the calculation of gradients for SGD, we did not perform simulations of gate error with SGD.

Each simulation we perform is characterized by four attributes: the problem (3-regular Max-Cut, Sherrington-Kirkpatrick, or Hubbard), the ansatz depth p, the choice of optimizer, and gate error level ε (possibly 0). For each set of attributes considered, we performed 50 statistically independent simulations. For each numerical simulation we performed, we estimate the wall-clock time of actually performing the experiment on a quantum computer accessed through a cloud service using the various cost models described in Section 2.3.2, and set a limit to the total amount of time allowed. We are interested in how quickly a given optimization algorithm converges to the optimal energy to within a target precision. By "optimal energy" we mean the energy of the ansatz state at the nearest local optimum as determined from a classical optimization of the noiseless objective function.

2.5.1 The case of $p = 1$ and no gate errors

First, we present the results of simulations with $p = 1$ and no gate errors. Figure 2.2 shows the wall clock time for different optimizers to achieve precision 10^{-3} for the Sherrington-Kirkpatrick model at $n = 8$ and $p = 1$. We define this time to be the earliest time at

which the current and all future evaluated points have an approximation ratio or normalized energy close to the optimal value to within 10^{-3}. We show the results for the three different wall-clock models described in Section 2.3.2: no network latency, network latency present but with circuit batching, and network latency present with no circuit batching. Note that Nelder-Mead converged in only 44 out of 50 runs; the other algorithms converged in all of them.

These results show that the proper choice of optimizer depends on the situation. SPSA performed the best under the wall clock time model with no latency, but was outperformed by MGD, SGD, and BOBYQA under the model that included latency and circuit batching. Under the model that included latency but did not have circuit batching, BOBYQA performed the best.

The importance of the wall clock time model, and in particular the effect of network latency, is evident. In the presence of network latency, MPG, MGD and SGD benefit much more from circuit batching than the other algorithms do. Both algorithms work by obtaining an estimate of the objective function gradient in each iteration. Circuit batching provides a benefit because multiple different circuits are needed to estimate the gradient, and these circuits can be sent over the network in one batch, reducing total network latency costs. SPSA also estimates the gradient, but it only uses 2 different circuits for that purpose. In contrast, the hyperparameters of MGD and MPG were chosen so that they both used 10, while SGD used 72. Indeed, the plot shows SGD benefiting from batching to a greater degree than MGD and MPG.

As an illustration of the ability of the various optimizers to tolerate different amounts of variance in the objective function, we note that the optimal hyperparameters dictates that SGD uses 1,000 measurement shots per evaluations, MGD and SPSA use 5,000, MPG uses 20,000, Nelder-Mead uses 25,000, and BOBYQA uses 125,000. This makes clear our distinction between deterministic and stochastic optimizers. While one can find external hyperparameter settings that allow Nelder-Mead and BOBYQA to succeed, the lack of internal hyperparameters for noise tolerance means the number of measurements grows wildly. In contrast, stochastic methods like MPG, MGD and SPSA can find balanced settings using far fewer measurements per point while remaining stable. In larger systems beyond the scope of simulation, it may not be easy to a priori determine the required measurements to make a deterministic method stable, and hence the flexibility of naturally stochastic methods is likely to be preferred. For all cases, however, some amount of hyperparameter tuning is a necessity for good performance.

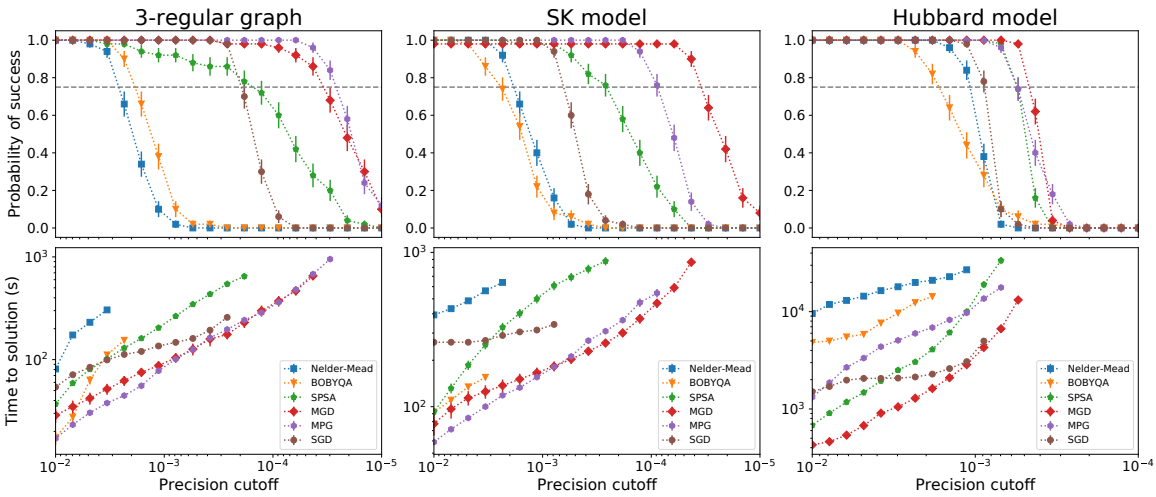


Figure 2.3: Success probability and time to solution for varying levels of required precision at $p = 5$. Top: The probability of converging (out of 50 trials) to the optimal value of the ansatz at the given precision. Bottom: The average wall clock time the optimizer took to reach the given precision. Error bars represent 1 standard deviation. Time to solution is only reported if the probability of convergence was at least 75% (dotted horizontal gray line). We see that Nelder-Mead and BOBYQA are the least likely to converge and often the slowest to converge when they do succeed. Meanwhile, MGD and MPG have the highest probability of converging as well as usually the fastest convergence times.

2.5.2 The case of $p = 5$ and no gate errors

At $p = 5$ there are a greater number of parameters to optimize. For the QAOA problems there are now 10 parameters, and for the Hubbard model there are 15. Here we fixed the wall clock time model to the one that includes network latency and circuit batching, and plot the performance of the optimizers as a function of the desired level of precision of convergence to the ansatz optimum. We present the results in Figure 2.3. The optimizers did not always converge within the time limit we allowed (1,500 seconds for the QAOA problems and 24 hours for the Hubbard model). The top row depicts the probability of convergence to the desired precision, out of 50 runs. The bottom row depicts the average wall clock time for convergence, with data plotted only if the probability of convergence was at least 75%.

These simulations show that not only were Nelder-Mead and BOBYQA the least likely to converge; they were also often the slowest to converge when they did succeed. Meanwhile, MGD, MPG, and SPSA converged even at high levels of precision, with MGD and MPG consistently converging the most quickly in this regime. This is again a symptom of the fragility of using deterministic optimizers in a stochastic setting. Outside the regime of precise tuning, methods like Nelder-Mead and BOBYQA become unstable, whereas even

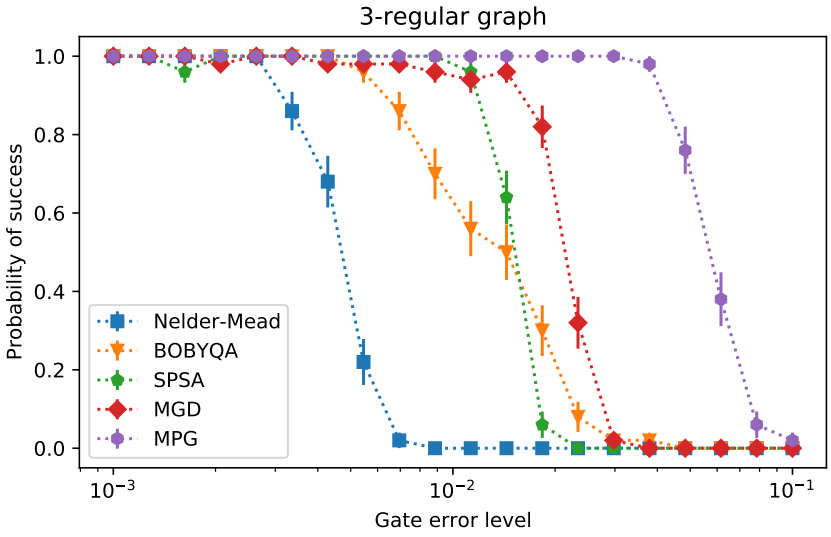


Figure 2.4: Probability of convergence as a function of gate error level under a model of rotation error for the 3-regular graph model. Shown is the probability, over 50 trials with different PRNG seeds, of converging to within a precision of 5e-3, as a function of gate error level. Error bars represent one standard deviation. In this scenario, Nelder-Mead is the least resilient to this noise, while MPG is the most, and MGD follows.

outside the regime of tuning, methods like MGD, MPG, and SPSA are able to succeed.

Note that the plots would look different if we had tuned the hyperparameters with a different strategy. For example, we tuned the hyperparameters to minimize the time to convergence to a precision of 10^{-3}. If we had instead used a less precise cutoff, such as 10^{-2}, then we would expect the optimizers to converge faster to less precise cutoffs, but perhaps more slowly or less robustly to higher precision cutoffs at smaller ones. At a glance in these figures, one can see remnants of the hyperparameter selection cutoff. In Appendix C we highlight this effect with an example.

2.5.3 The impact of rotation errors at $p = 5$

Finally, to understand the impact of gate error in addition to simple sampling noise, at $p = 5$ we consider gate rotation errors as well. As described above, the model of gate rotation error that we used does not simply translate to SGD, so we do not include results for it. Again, we fixed the wall clock time model to the one that includes network latency and circuit batching. In running the optimization algorithms, we used the hyperparameters that were optimized for the case of no gate errors.

Figure 2.4 shows the probability of convergence to a precision of 5×10^{-3} for the various optimizers as a function of the gate error level ε, for the 3-regular graph model.

The results show that in this scenario, Nelder-Mead is the least resilient to this type of noise, while MPG and MGD are the most. SPSA also showed good noise resilience in other scenarios; see Section D in the appendix for data for the other models.

Note that for a given gate error level, algorithmic improvements can increase the success probability with respect to the ideal solution only up to a certain point. That is, beyond a certain level of noise, the device cannot produce a more precise solution, and hence this is not a failing of the optimizer but rather represents a device limitation. We do not differentiate between these circumstances in the presented data, but merely note that it is a consideration when defining probability of success.

2.6 Conclusion

Variational quantum algorithms are a promising candidate for execution on near-term quantum computers, and a number of experimental demonstrations of these algorithms have already been performed. These algorithms rely on a classical optimization subroutine, and hence the efficiency of these algorithms can be limited by the performance of these optimizers. Here, we saw that to accurately assess the performance of these optimizers, it is crucial to develop a good cost model, and tune available hyperparameters to operational specifications.

Given the unique considerations of quantum systems, we developed two new surrogate model-based optimizers, MGD and MPG, to fill some of the gaps of previous methods. We numerically compared their performance with other popular alternatives, and found it advantageous in several realistic settings. We also probed how the cost model and presence of errors can significantly impact the choice of optimizer in a practical setting.

Now that quantum computers are coming online, accessing superconducting qubits through a cloud interface is an important scenario to consider. The latency of communicating over the Internet can cause large increases in running times, but this can be mitigated by circuit batching, though the cost savings depends on the optimizer.

We also observed that inherently stochastic optimizers, such as MPG, MGD and SPSA, were more robust to variations in problems or setting once properly tuned. This extended to situations where finite gate or circuit noise was present. In contrast, while it was sometimes possible to make deterministic optimizers competitive through careful tuning, these tunings were fragile with respect to small variations in the problem or the introduction of noise. Overall, MPG and MGD's tolerance of noise, ability to take advantage of circuit batching, and good overall performance make them good candidates for actual experiments, but the best optimizer can depend on the processor's wall-clock model, level of noise, number of

parameters, or the specific circuit ansatz.

In this work, we have shown how practical considerations can significantly affect the calculus of choosing an optimizer for running variational algorithms. Future work will develop more accurate noise and cost models, and further development of optimizers can take these unique considerations into account.

Code Availability

Implementations of Model Gradient Descent and Model Policy Gradient are available at https://github.com/quantumlib/ReCirq.

CHAPTER 3

Quantum Approximate Optimization on a Superconducting Qubit Processor

3.1 Introduction

The Google Sycamore superconducting qubit platform has been used to demonstrate computational capabilities surpassing those of classical supercomputers for certain sampling tasks [8]. However, it remains to be seen whether such processors will be able to achieve a similar computational advantage for problems of practical interest. Along with quantum chemistry [75, 76], machine learning [77], and simulation of physical systems [3], discrete optimization has been widely anticipated as a promising area of application for quantum computers.

Beginning with a focus on quantum annealing [78] and adiabatic quantum computing [79], the possibility of quantum enhanced optimization has driven much interest in quantum technologies over the years. This is because faster optimization could prove transformative for diverse areas such as logistics, finance, machine learning, and more. Such discrete optimization problems can be expressed as the minimization of a quadratic function of binary variables [80, 81], and one can visualize these cost functions as graphs with binary variables as nodes and (weighted) edges connecting bits whose (weighted) products sum to the total cost function value. For most industrially-relevant problems, these graphs are non-planar and many ancilla would be required to embed them in (quasi-)planar graphs matching the qubit connectivity of most hardware platforms [82]. This limits the applicability of scalable architectures for quantum annealing [83] and corresponds to increased circuit complexity in digital quantum algorithms for optimization such as QAOA.

The quantum approximate optimization algorithm (QAOA) is the most studied gate model approach for optimization using near-term devices [9]. While the prospects for achieving quantum advantage with QAOA remain unclear, QAOA prescribes a simple

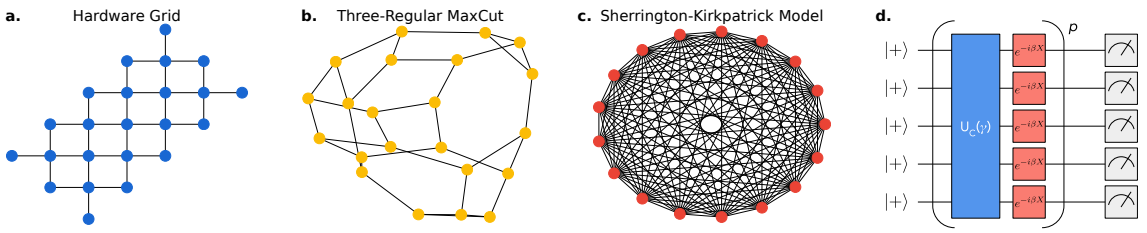


Figure 3.1: We studied three families of optimization problems: **a.** Hardware Grid problems with a graph matching the hardware connectivity of the 23 qubits used in this experiment. **b.** MaxCut on random 3-regular graphs, with the largest instance depicted (22 qubits). **c.** The fully-connected Sherrington-Kirkpatrick (SK) model shown at the largest size (17 qubits). **d.** QAOA uses p applications of problem and driver unitaries to approximate solutions to optimization problems. The parameters γ and β are shared among qubits in a layer but different for each of the p layers.

paradigm for optimization which makes it amenable to both analytical results and implementation on current processors [84, 85, 86, 87, 88, 89, 90, 91, 92]. For these reasons, QAOA has also become popular as a system-level benchmark of quantum hardware. This work builds on prior experimental demonstrations of QAOA on superconducting qubits [24, 93, 94, 95], ion traps [27], and photonics systems [96].

We are able to experimentally resolve, for the first time, increased performance with greater QAOA depth and apply QAOA to cost functions on graphs that deviate significantly from our hardware connectivity. Owing to the low error rates of the Sycamore platform, the trade-off between the theoretical increase in quality of solutions with increasing QAOA depth and additional noise is apparent for hardware-native problems. We also apply the algorithm to non-native graph problems with their necessary compilation overhead and study the scaling of solution quality and problem size. Our results reveal that the performance of QAOA is qualitatively different when applied to hardware native graphs versus more complex graphs, highlighting the challenge of scaling QAOA to problems of industrial importance.

For this study, we used a "Sycamore" quantum processor which consists of a two-dimensional array of 54 transmon qubits [8]. Each qubit is tunably coupled to four nearest neighbors in a rectangular lattice. In this case, all device calibration was fully automated and data was collected using a cloud interface to the platform programmed using Cirq [97]. Our experiment was restricted to 23 physical qubits of the larger Sycamore device, arranged in a topology depicted in Figure 3.1a.

3.2 The QAOA

The quantum approximate optimization algorithm (QAOA) is a quantum algorithm for finding approximate solutions to combinatorial optimization problems. It was introduced in [9]. In this section, we describe the algorithm.

A combinatorial optimization problem is specified by n binary variables $\{z_j\}_{j=1}^n$ and m clauses $\{C_\alpha\}_{\alpha=1}^m$. An assignment of each variable to either 0 or 1 produces a bitstring $z = (z_1, \ldots, z_n)$. A clause is a function that takes a bitstring and outputs either 0 or 1. If $C_\alpha(z) = 1$ then we say that z satisfies the clause C_α; otherwise, it does not satisfy the clause. The goal of the problem is to find a bitstring z that satisfies as many clauses as possible. That is, we want to maximize the cost function

$$C(z) = \sum_{\alpha=1}^{m} C_\alpha(z). \tag{3.1}$$

Each clause C_α can be identified with a quantum operator that is diagonal in the computational basis, with diagonal elements given by $\langle z|C_\alpha|z\rangle = C_\alpha(z)$. Here, we abuse notation and use C_α to refer to both the clause and the associated quantum operator. We can then associate the cost function with a quantum operator which we also refer to by C, given by

$$C = \sum_{\alpha} C_\alpha. \tag{3.2}$$

This operator is also diagonal in the computational basis and has diagonal elements given by $\langle z|C|z\rangle = C(z)$.

Because the quantum operator C_α is diagonal in the computational basis, it can be expanded as a sum of terms containing Pauli Z operators. This can be viewed as the Fourier expansion of the boolean function C_α [98]. As an example, consider the MaxCut problem. In this problem, we are given a graph with n vertices and would like to partition the vertices into two groups in order to maximize the number of cut edges, where we say an edge is cut if its vertices belong to different groups. If we assign a variable to each vertex and interpret the binary values 0 and 1 as representing the two groups, then each edge $\langle jk\rangle$ is associated with a clause given by

$$C_{\langle jk\rangle} = \frac{I - Z_j Z_k}{2}. \tag{3.3}$$

where Z_j is the Pauli Z operator acting on the j-th qubit. This works because

$$\langle z|C_{\langle jk\rangle}|z\rangle = \begin{cases} 0 & \text{if } z_j = z_k \\ 1 & \text{if } z_j \neq z_k \end{cases} . \tag{3.4}$$

In general, the quantum operator C_α will only act on bits that are involved in the corresponding clause.

In the QAOA, we define an operator that depends on a parameter γ,

$$U_C(\gamma) = e^{-i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}. \tag{3.5}$$

We call this the "problem unitary" because it depends on the cost function specified by the problem. This operator maps computational basis states as $|z\rangle \mapsto e^{-i\gamma C(z)}|z\rangle$. We also define a unitary that depends on an angle β,

$$U_B(\beta) = e^{-i\beta B} = \prod_j e^{-i\beta X_j}, \qquad B = \sum_j X_j \tag{3.6}$$

This unitary is a product of X rotations and does not depend on the problem. The initial state is the uniform superposition of all computational basis states:

$$|s\rangle = \frac{1}{\sqrt{2}} \sum_z |z\rangle. \tag{3.7}$$

To implement the QAOA, we pick a positive integer p and choose parameters $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$. These parameters determine a quantum state

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U_B(\beta_p)U_C(\gamma_p) \cdots U_B(\beta_1)U_C(\gamma_1)|s\rangle, \tag{3.8}$$

The QAOA works by preparing the state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ and measuring it to obtain a bitstring. With a good choice of the parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, we hope that the measured bitstring will provide a good approximate solution to the problem. In practice, we treat the QAOA as a variational quantum algorithm in which the goal is to find parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ in order to maximize the expected value of the cost function

$$\langle C \rangle = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}|C|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle. \tag{3.9}$$

It can be shown that as p goes to infinity, the maximum possible expectation value tends

towards the solution of the problem. That is,

$$\lim_{p\to\infty} \max_{\boldsymbol{\gamma},\boldsymbol{\beta}} \langle \boldsymbol{\gamma},\boldsymbol{\beta}|C|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle = \max_z C(z). \tag{3.10}$$

The QAOA works at a finite value of p, though, and the main question of theoretical interest is whether it can provide good approximate solutions as measured by the approximation ratio

$$\frac{\langle \boldsymbol{\gamma},\boldsymbol{\beta}|C|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle}{\max_z C(z)}. \tag{3.11}$$

The QAOA has been proven to achieve nontrivial approximation ratios for some problems, and it is an open question whether there is a problem for which the QAOA achieves a higher approximation ratio than the best known classical algorithm [9, 84, 99, 92].

3.3 Compilation and problem families

In this work, we consider problems given by cost Hamiltonians of the form

$$C = \sum_{j<k} w_{jk} Z_j Z_k \tag{3.12}$$

where $w_{jk} \in \{0, \pm 1\}$. Note that this can represent the cost function of the MaxCut problem up to a constant shift and rescaling factor. The cost Hamiltonian (3.12) can be associated with a graph on n vertices where there is an edge between vertices j and k if $w_{jk} \neq 0$. We will study three families of problem graphs depicted in Figure 3.1. We choose the convention that the goal is to *minimize* the expected value of the cost function, but for comparison among problem instances, we divide by $C_{\min} = \min_{\mathbf{z}} C(\mathbf{z})$, which is negative for all problems we study, so we are in fact *maximizing* $\langle C\rangle / C_{\min}$.

We approach compilation as two distinct steps: routing and gate synthesis. The need for routing arises when simulating U_C for a cost function C defined on a graph that is not a subgraph of our planar hardware connectivity. To simulate such U_C we perform layers of swap gates (forming a swap network) which permute qubits such that all edges in the problem graph correspond to an edge in the hardware graph at least once, at which point the corresponding cost function terms can be implemented. An example of such a swap network is depicted in Figure 3.2a.

The final compilation step, gate synthesis, involves decomposing arbitrary 1- and 2-qubit interactions into physical gates supported by the device (see, e.g. Figure 3.2b). The

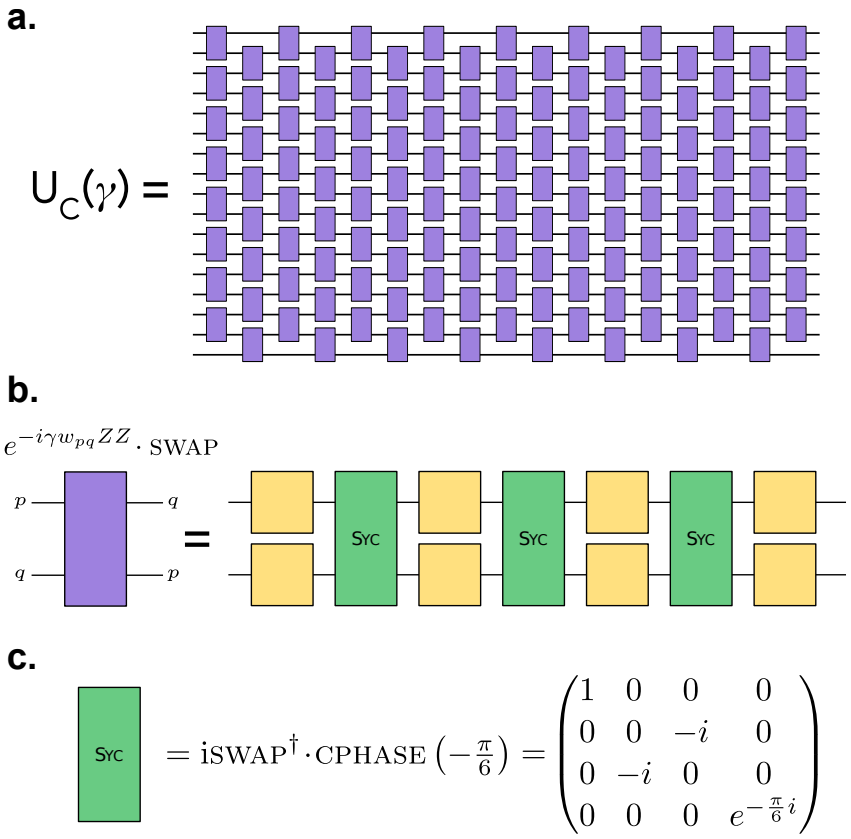


Figure 3.2: **a.** The linear swap network can route a 17-qubit SK model problem unitary to n layers of nearest-neighbor two qubit interactions. **b.** The $e^{-i\gamma w ZZ} \cdot$ SWAP interaction is a composite phasing and SWAP operation which can be synthesized from three applications of our hardware native entangling SYC and γw-dependent single-qubit rotations (yellow boxes). **c.** The definition of the SYC gate.

physical gates used in this experiment are arbitrary single-qubit rotations and a two-qubit entangling gate native to the Sycamore hardware which we refer to as the SYC gate and define in Figure 3.2c. Through multiple applications of this gate and single-qubit rotations, we are able to realize arbitrary entangling gates. Compilation details can be found in Appendix E. The average two-qubit gate fidelity on this device was 99.4% as measured by cross entropy benchmarking [8] and average readout fidelity was 95.9% per qubit. We now discuss compilation for the three families of optimization problems studied in this work.

Hardware Grid Problems. Swap networks are not required when the problem graph matches the connectivity of our hardware; this is the main reason for studying such problems despite results showing that problems on such graphs are efficient to solve on average [81, 100]. We generated random instances of hardware grid problems by sampling w_{ij} to be ± 1 for edges in the device topology (and zero otherwise). Gates are scheduled so that the degree-four interaction graph can be implemented in four rounds of two-qubit gates by cycling through the interactions to the left, right, top and bottom of each interior qubit. Each two-qubit ZZ interaction can be synthesized with two layers of hardware-native SYC gates interleaved with γ-dependent single-qubit rotations. In total, each application of the problem unitary is effected with eight total layers of SYC gates.

Sherrington-Kirkpatrick (SK) Model. A canonical example of a frustrated spin glass is the Sherrington-Kirkpatrick model [55]. It is defined on the complete graph with w_{ij} randomly chosen to be ± 1. For large n, optimal parameters are independent of the instance [92]. The SK model is the most challenging model to implement owing to its fully-connected interaction graph. Optimal routing can be performed using the linear swap networks discussed in Ref. [101] and depicted in Figure 3.2. This requires n layers of the composite $e^{-i\gamma w ZZ} \cdot$ SWAP interaction, each of which can be synthesized from three SYC gates with interleaved γ-dependent single-qubit rotations. Thus, one application of the problem unitary can be effected in $3n$ layers of SYC gates.

MaxCut on 3-Regular Graphs. MaxCut is a widely studied problem, and there is a polynomial-time algorithm due to Goemans and Williamson [102] which guarantees a certain approximation ratio for all graphs, and it is an open question whether QAOA can efficiently achieve this or beat it [103]. Unlike the previous two problem families, all edge weights are set to 1, and we sample random 3-regular graphs to generate various instances. The connectivity of the problem Hamiltonian's graph differs for each instance. While one could use the fully-connected swap network to route these circuits, this is wasteful. Instead, we used the routing functionality from the $\mathrm{t}|\mathrm{ket}\rangle$ compiler to heuristically insert SWAP operations which move logical assignments to be adjacent [104]. These compiled circuits are of roughly equal depth to those from a fully-connected swap network, but the number

of two qubit operations is roughly quadratically reduced.

3.4 Comparisons with prior work

Reference	Date	Problem topology	$\Delta(G)$	n	p	Optimization
[24]	2017-12	Hardware	3	19	1	Yes
[96]	2018-08	Hardware	1	2	1	No
[27]	2019-06	Hardware[1] (system 1)	n	12, 20	1	Yes
		Hardware[1] (system 2)	n	20–40	1–2[(2)]	No
[93]	2019-07	Hardware	3	8	1	No
[94]	2019-12	Ring	2	4	1	No
		Fully-connected	n			No
[95]	2019-12	Hardware	1	2	1, 2	Yes
This work		Hardware	4	2–23	1–5	Yes
		3-regular	3	4–22	1–3	Yes
		Fully-connected	n	3–17	1–3	Yes

Table 3.1: An overview of experimental demonstrations of QAOA. Although each work generally frames the algorithm in terms of a combinatorial optimization problem (2SAT, Exact Cover, etc.), we classify problems based on their topology, maximum degree of the problem graph $\Delta(G)$, the number of qubits n and the depth of the algorithm p. These attributes give a rough view of the difficulty of a particular instance. We indicate whether variational optimization of parameters was demonstrated. [1]In superconducting processors, "Hardware" topologies are 2-local planar lattices. In ion trap processors, hardware-native topologies are long range couplings of the form $J_{ij} \approx J_0/|i-j|^\alpha$. [2]$p = 2$ only for $n = 20$.

Prior work has included experimental demonstration of the QAOA. The referenced works often include additional results, but we focus specifically on the sections dealing with experimental implementation of the algorithm.

[24] demonstrated a Bayesian optimization of p = 1 parameters on a 19-bit hardware-native Ising graph using a Rigetti superconducting qubit processor. The authors compared the cumulative probability of finding the lowest energy bitstring over the course of the optimization to binomial coin flips and showed performance from the device exceeding random guessing. The problem topology involved a roughly hexagonal tessellation. The problems were related to a restricted form of two-class clustering.

[96] demonstrated a $n = 2, p = 1$ QAOA landscape on their photonic quantum processor. They presented three instances of the two-bit problem, which was framed as Max2Xor. The color scale for the landscapes was re-scaled for experimental values. They demonstrated a high probability of obtaining the correct bitstrings.

[27] demonstrated application of the QAOA with two ion trap quantum processors, called "system 1" and "system 2". The problems were of the form $J_{ij} \approx J_0/|i-j|^\alpha$ with α close to unity. This corresponds to an antiferromagnetic 1D chain. This problem is fully connected, but is spiritually similar to the hardware native planar graphs studied in superconducting architectures in the sense that the cost function cannot be programmed and is easily solvable at any system size. A landscape is shown for $n = 20$ from system 1. Optimization traces are shown for $n = 12$ and $n = 20$ on system 1. On system 2, performance was demonstrated at optimal parameters for $n = \{20, 25, 30, 35, 40\}$. Additionally, a partial $p = 2$ grid search was performed on system 2. Nine discrete choices for $(\gamma_1, \beta_1, \beta_2)$ were selected and then a scan over γ_2 was reported for each choice. Finally, on system 2, performance was compared between $p = 1$ and $p = 2$ at $n = 20$, giving a ratio of $(93.8 \pm 0.4)\%$ versus $(93.9 \pm 0.3)\%$, respectively.

[93] demonstrated an application of the QAOA via IBM's Quantum Experience cloud service on the 16Q Melbourne device. The 8-bit problem studied was framed as 2SAT and had a topology matching the device with maximum node degree of 3. A landscape with re-scaled color map was compared to the theoretical landscape.

[94] implemented QAOA on two types of problems; each with two compilation strategies. The 4-bit problems had a ring topology and a fully-connected topology. While a 4-qubit ring would fit on the Rigetti superconducting device, they implemented both problems using only linear connectivity with the introduction of SWAPs. In one compilation strategy, they used CZ as the gate-synthesis target. In the other, they used both CZ and iSWAP. The color bars were re-scaled for the experimental data.

[95] ran 2-bit QAOA instances on their superconducting architecture at $p = 1$ and $p = 2$. They show four $p = 1$ landscapes and demonstrate optimization for $n = 2, p = 2$. They observed that increasing circuit depth to $p = 2$ increases the probability of observing the correct bitstring.

3.5 Energy landscapes and optimization

QAOA is a variational quantum algorithm where circuit parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ are optimized using a classical optimizer, but function evaluations are executed on a quantum processor [86, 105, 106]. First, one repeatedly constructs the state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ with fixed parameters and samples bitstrings to estimate $\langle C \rangle \equiv \langle \boldsymbol{\gamma}, \boldsymbol{\beta} | C | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle$. On our superconducting qubit platform we can sample roughly five thousand bitstrings per second. A classical "outer-loop" optimizer can then suggest new parameters to decrease the observed expectation value. Note that we normalize by the cost function's true minimum, so we are in fact maximiz-

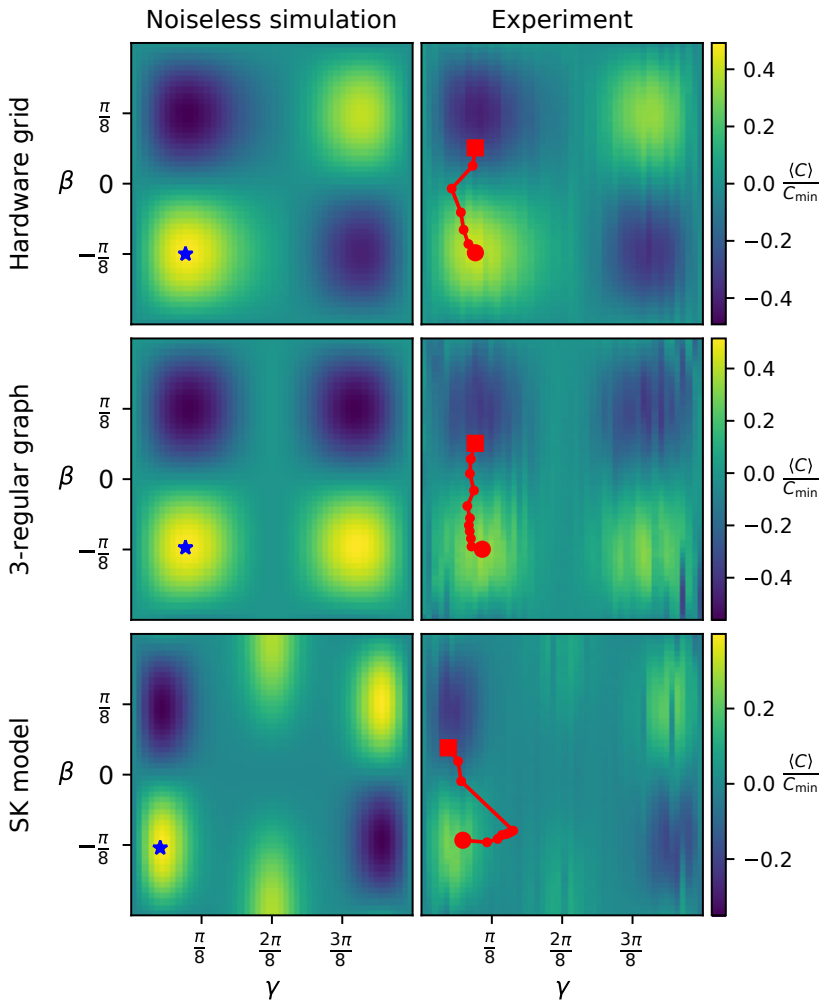


Figure 3.3: Comparison of simulated (left) and experimental (right) $p = 1$ landscapes, with a clear correspondence of landscape features. An overlaid optimization trace (red, initialized from square marker) demonstrates the ability of a classical optimizer to find optimal parameters. The blue star in each noiseless plot indicates the theoretical local optimum. Problem sizes are $n = 23$, $n = 14$, and $n = 11$ for Hardware Grid, 3-regular MaxCut, and SK model, respectively.

ing $\langle C \rangle / C_{\min}$ ($C_{\min}$ is negative and hence, minimizing $\langle C \rangle$ corresponds to maximizing $\langle C \rangle / C_{\min}$).

For $p = 1$, we can visualize the cost function landscape as a function of the parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta}) = (\gamma_1, \beta_1)$ in a three-dimensional plot (where we drop the subscripts and label the axes γ and β). The presence of features like hills and valleys in the landscape gives confidence that a classical optimization can be effective. Comparison of simulated and empirical $p = 1$ landscapes is a common qualitative diagnostic for the performance of experiments [96, 27, 93, 94, 95]. For classical optimization to be successful the quantum computer must provide accurate estimates of $\langle C \rangle$. Otherwise, noise can overwhelm any signal making it difficult for a classical optimizer to improve the parameter estimates. Issues such as decoherence, crosstalk, and systematic errors manifest as differences (e.g., damping or warping) from the ideal landscape.

Figure 3.3 contains simulated theoretical and experimental landscapes for selected instances of the three problem families evaluated on a grid of $\beta \in [-\pi/4, \pi/4]$ and $\gamma \in [0, \pi/2]$ parameters with a resolution of 50 points along each linear axis. Each expectation value was estimated using 50,000 circuit repetitions with efficient post-processing to compensate for readout bias (see Appendix F). The hardware grid problem shows clear features at the maximum size of our study, $n = 23$. For the other two problems performance degrades with increasing n and so we show data at $n = 14$ for the 3-regular graph problem and $n = 11$ for the SK model. We highlight the correspondence between experimental and theoretical landscapes for problems of large size and complexity. Prior experimental demonstrations have presented landscapes for a maximum of $n = 20$ on a hardware-native interaction graph [27] and a maximum of $n = 4$ for fully-connected problems like the SK model [94].

In Figure 3.3, we also overlay a trace of the classical optimizer's path through parameter space as a red line. We used the optimizer Model Gradient Descent (MGD) described in Chapter 2. In this example, we initialized the parameter optimization from an intentionally bad parameter setting and observed that MGD was able to enter the vicinity of the optimum in 10 iterations or fewer, with each iteration consisting of six energy evaluations of 25,000 shots each.

3.6 Hardware performance of QAOA

As the name implies, noisy intermediate-scale quantum (NISQ) processors are noisy devices with high error rates and a variety of error channels. Thus, NISQ circuits are expected to degrade in performance as the number of gates is increased. Here, we study the perfor-

mance of QAOA as implemented on our quantum processor at different n and p using an application-specific metric: the normalized observed cost function $\langle C \rangle / C_{\min}$. A value of 1 is perfect and 0 corresponds to the performance we would expect from random guessing. In order to distinguish the effects of noise from the robustness provided by using a classical outer-loop optimizer, here we report results obtained from running circuits at the theoretically optimal $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ values.

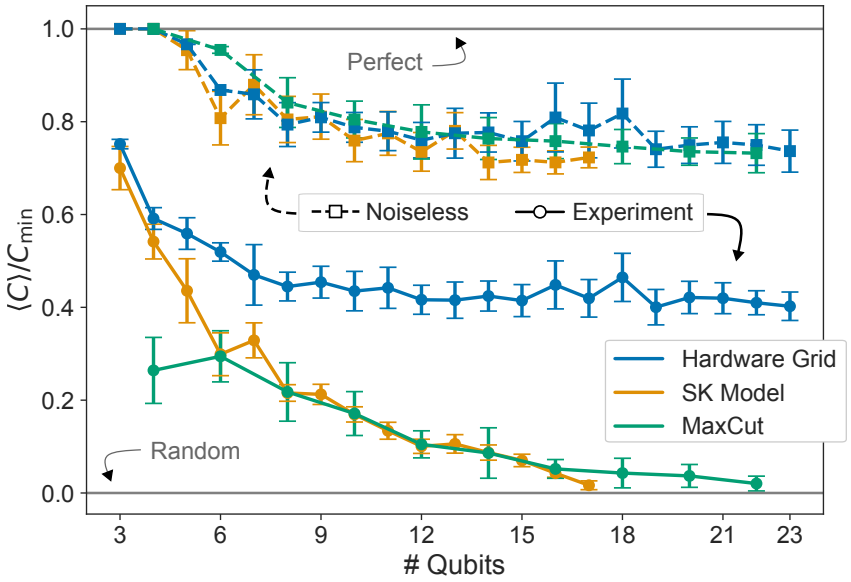


Figure 3.4: QAOA performance as a function of problem size, n. Each size is the average over ten random instances (std. deviation given by error bars). While Hardware Grid problems show n-independent noise, we observe that experimental SK model and MaxCut solutions approach those found by random guessing as n is increased.

In Figure 3.4, we observe that $\langle C \rangle / C_{\min}$ achieved for the hardware graph seems to saturate to a value that is independent of of n. This occurs despite the fact that circuit fidelity is decreasing with increasing n. In fact, this is theoretically anticipated behavior that can be understood by moving to the Heisenberg operator formalism and considering an observable $Z_i Z_j$. The expectation value for this operator is conjugated by the circuit unitary involving p applications of the instance graph. This gives an expression for the expectation value of $Z_i Z_j$ which only involves qubits that are at most p edges away from i and j. Thus for fixed p, the error for a given term is asymptotically unaffected as we grow n. Recall that C is a sum of these terms, so the total error scales linearly with n; but $C_{\min} \propto n$ so $\langle C \rangle / C_{\min}$ is constant with respect to n. Note that non-local error channels or crosstalk could potentially remove this property.

Compiled problems—namely SK model and 3-regular MaxCut problems—result in deeper circuits extensive in the number of qubits. As the depth grows, there is a higher

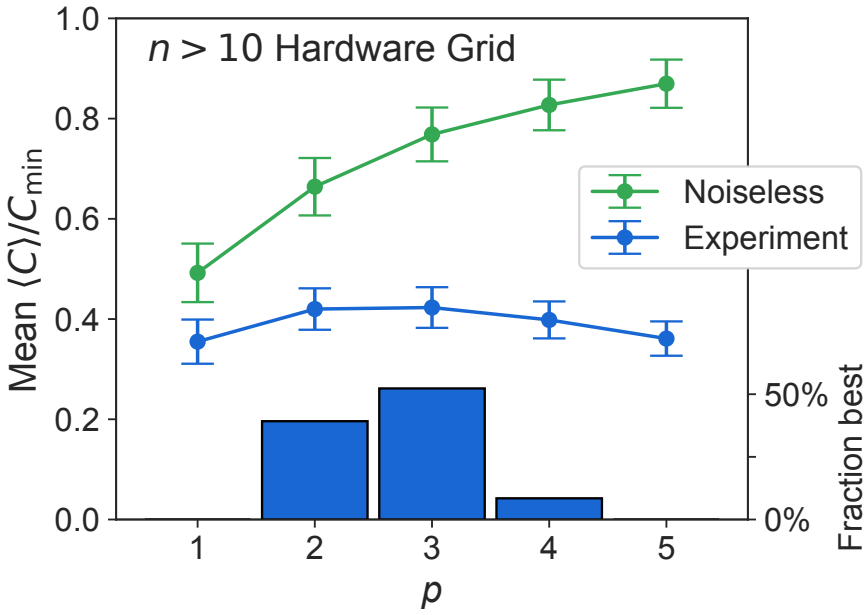


Figure 3.5: QAOA performance as a function of depth, p. In ideal simulation, increasing p increases the quality of solutions. For experimental Hardware Grid results, we observe increased performance for $p > 1$ both as measured by the mean over all instances (lines) and statistics of which p maximizes performance on a per-instance basis (histogram). At larger p, errors overwhelm the theoretical performance increase.

chance of an error occurring. The high degree of the SK model graph and the high effective degree of the MaxCut circuits after compilation means that these errors quickly propagate among all qubits and the quality of solutions can be approximately modeled as the result of a depolarizing channel, with further analysis in Appendix G. Even on these challenging problems, we observe performance exceeding random guessing for problem sizes up to 17 bits, even with circuits of depth $p = 3$. Note finally that despite circuits with significantly fewer gates (although similar depth), performance on the MaxCut instances tracks performance on the SK model instances rather closely, further substantiating the circuit depth as a useful proxy for the performance of QAOA.

In a noiseless case, the quality of a QAOA solution can be improved by increasing the depth parameter p. However, the additional depth also increases the probability of error. We study this interplay between noise and algorithmic power in Figure 3.5. Previously, improved performance with $p > 1$ had only been experimentally demonstrated for an $n = 2$ problem [95]. For larger problems ($n = 20$), performance for $p = 2$ was shown to be within error bars of the $p = 1$ performance [27]. Figure 3.5 shows the p-dependence averaged across all 130 instances where $n > 10$. The mean finds its maximum at $p = 3$, although there are variations among the instances comparable in scale to the experimental

p-dependence. The relatively flat dependence of performance on depth suggests that the experimental noise seems to nearly balance the increase in theoretical performance for this problem family. For a more meaningful aggregation of the many random instances across problem sizes, we consider each instance individually and identify which value of the hyperparameter p maximizes performance for that particular instance. A histogram of these per-instance maximal values is inset in Figure 3.5, showing that performance is maximized at $p = 3$ for over half of instances larger than ten qubits. Note finally that our full dataset (see Appendix H) includes per-instance data at all settings of p.

3.7 Conclusion

Discrete optimization is an enticing application for near-term devices owing to both the potential value of solutions as well as the viability of heuristic low-depth algorithms such as the QAOA. While no existing quantum processors can outperform classical optimization heuristics, the application of popular methods such as the QAOA to prototypical problems can be used as a benchmark for comparing various hardware platforms.

Previous demonstrations of the QAOA have primarily optimized problems tailored to the hardware architecture at minimal depth. Using the Google "Sycamore" platform, we explored these types of problems, which we termed Hardware Grid problems, and demonstrated robust performance at large numbers of qubits. We showed that the locations of maxima and minima in the $p = 1$ diagnostic landscape match those from the theoretically computed surface, and that variational optimization can still find the optimum with noisy quantum objective function evaluation. We also applied the QAOA to various problem sizes using pre-computed parameters from noiseless simulation, and observed an n-independent noise effect on the approximation ratios for Hardware Grid problems. This is consistent with our theoretical understanding that the noise-induced degradation of each term in the objective function remains constant in the shallow-depth regime where correlations remain local. Furthermore, we report the first clear cases of performance maximization at $p = 3$ for the QAOA owing to the low error rate of our hardware.

Most real world instances of combinatorial optimization problems cannot be mapped to hardware-native topologies without significant additional resources. Instead, problems must be compiled by routing qubits with swap networks. This additional overhead can have a significant impact on the algorithm's performance. We studied random instances of the fully-connected SK model. Although we report non-negligible performance for large ($n = 17$), deep ($p = 3$), and complex (fully-connected) problems, we see that performance degrades with problem size for such instances.

The promise of quantum enhanced optimization will continue to motivate the development of new quantum technology and algorithms. Nevertheless, for quantum optimization to compete with classical methods for real-world problems, it is necessary to push beyond contrived problems at low circuit depth. Our work demonstrates important progress in the implementation and performance of quantum optimization algorithms on a real device, and underscores the challenges in applying these algorithms beyond those natively realized by hardware interaction graphs.

Code and Data Availability

The code used in this experiment is available at https://github.com/quantumlib/ReCirq. The experimental data for this experiment is available at http://dx.doi.org/10.6084/m9.figshare.12597590.v2.

CHAPTER 4

Preparing Slater Determinants and Fermionic Gaussian States

4.1 Introduction

An important scientific challenge is to determine the properties of a material specified by its atomic configuration or crystal structure. For instance, is the material hard or soft? Does it conduct or insulate? Does it superconduct? Is it good at converting solar radiation into more useful forms of energy? One way to perform this task is to synthesize the material in a laboratory, but this is often a very costly process. An alternative is to simulate the material on a computer. The properties of a material are determined by the behavior of the electrons within it, so the challenge is to compute the properties of a system of many interacting electrons. Materials scientists and chemists have been tackling this challenge for decades, and advances in computational methods and computing power have revolutionized their fields [107]. Nevertheless, the simulation of quantum systems remains a challenging problem for classical computers. The essential difficulty seems to be the fact that storing the full representation of a quantum system requires resources that scale exponentially in the size of the system. While many classical approximation methods exist, methods that guarantee calculations up to "chemical accuracy," about 1.5 milliHartrees, are feasible only for very small system sizes [107]. Quantum computers hold the promise of performing calculations for system sizes and accuracies inaccessible with a classical computer [5], and experimental proofs-of-concept have already been demonstrated [76, 23].

One of the advantages of performing a simulation on a quantum computer over a laboratory experiment with real materials is the possibility of preparing the computer precisely in a known initial state. While arbitrary states cannot be prepared efficiently, many useful classes of states can be. In this work, we give quantum algorithms to prepare arbitrary eigenstates of quadratic Hamiltonians, also known as fermionic Gaussian states, a useful class of initial states for electronic structure simulations. For the special case of preparing

Slater determinants, our algorithm improves on the algorithm given in [108] by exploiting a unitary symmetry. Our algorithms use only nearest-neighbor interactions on a linear array, a restriction motivated by current superconducting architectures.

4.2 Background

4.2.1 Second quantization and the canonical anticommutation relations

Second quantization is a mathematical formalism used to describe systems of fermions. The idea is that there are a number of *modes* which can be occupied by a fermion or not. A system of N fermionic modes is described by a set of fermionic *annihilation operators* $\{a_p\}_{p=1}^N$ satisfying the *canonical anticommutation relations*

$$a_p a_q + a_q a_p = 0, \tag{4.1}$$

$$a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}. \tag{4.2}$$

The adjoint $a_p^\dagger$ of an annihilation operator a_p is called a *creation operator*, and we refer to creation and annihilation operators as fermionic *ladder operators*.

We will always work in a finite-dimensional vector space, and in this setting, the anticommutation relations (4.1) and (4.2) have the following consequences [109]:

- The operators $\{a_p^\dagger a_p\}_{p=1}^N$ commute with each other and have eigenvalues 0 and 1. These are called the *occupation number operators*.
- There is a normalized vector $|\text{vac}\rangle$, called the *vacuum state*, which is a mutual 0-eigenvector of all the $\{a_p^\dagger a_p\}$.
- If $|\psi\rangle$ is a 0-eigenvector of $a_p^\dagger a_p$, then $a_p^\dagger|\psi\rangle$ is a 1-eigenvector of $a_p^\dagger a_p$. This explains why we say that $a_p^\dagger$ creates a fermion in mode p.
- If $|\psi\rangle$ is a 1-eigenvector of $a_p^\dagger a_p$, then $a_p|\psi\rangle$ is a 0-eigenvector of $a_p^\dagger a_p$. This explains why we say that a_p annihilates a fermion in mode p.
- $a_p^2 = 0$ for all p. One cannot create or annihilate a fermion in the same mode twice.
- The set of 2^N vectors

$$\left(a_1^\dagger\right)^{i_1} \cdots \left(a_N^\dagger\right)^{i_N} |\text{vac}\rangle, \qquad i_1, \ldots, i_N \in \{0, 1\} \tag{4.3}$$

are orthonormal. Without loss of generality, we assume that they form a basis for the entire vector space.

- The annihilation operators $\{a_p\}$ act on this basis as follows:

$$\langle \text{vac} | \left[a_N^{j_N} \cdots a_1^{j_1} \right] a_p \left[\left(a_1^\dagger \right)^{i_1} \cdots \left(a_N^\dagger \right)^{i_N} \right] | \text{vac} \rangle = \delta_{j_p,0} \delta_{i_p,1} \prod_{q<p} \delta_{j_q,i_q} (-1)^{i_q}. \quad (4.4)$$

See [109] for a derivation and discussion of these consequences.

4.2.2 Mapping fermions to qubits

To simulate a system of fermions on a quantum computer, we must choose a concrete set of qubit operators which satisfy the canonical anticommutation relations (4.1) and (4.2), i.e., we need to map the fermionic operators to qubit operators. There are several ways of achieving this; in this work we mainly use the Jordan-Wigner Transformation (JWT):

$$a_p \mapsto \frac{1}{2}(X_p + iY_p) Z_1 \dots Z_{p-1} = (|0\rangle\langle 1|)_p Z_1 \dots Z_{p-1}, \quad (4.5)$$

where X, Y, and Z are the Pauli matrices. Under the JWT, computational basis states correspond exactly to the basis (4.3):

$$\left(a_1^\dagger \right)^{i_1} \cdots \left(a_N^\dagger \right)^{i_N} | \text{vac} \rangle \mapsto | i_1, \dots, i_N \rangle . \quad (4.6)$$

4.3 Algorithms for state preparation

The first step in performing a quantum simulation is initializing the computer in a known state. In this section, we give algorithms to prepare important classes of starting states. These algorithms can be used, for instance, to prepare the ground state of the BCS mean-field Hamiltonian used in [108] to study superconductivity in the Fermi-Hubbard model.

4.3.1 Preparing Slater determinants

A Slater determinant is a state of the form

$$|\Psi_S\rangle = b_1^\dagger \cdots b_{N_f}^\dagger |\text{vac}\rangle, \qquad b_p^\dagger = \sum_{q=1}^{N} Q_{pq} a_q^\dagger, \quad (4.7)$$

where Q is an $N_f \times N$ matrix with orthonormal rows. The $\{b_p\}$ are a new set of fermionic annihilation operators that also satisfy the anticommutation relations (4.1) and (4.2).

The standard algorithm for preparing Slater determinants was described in [110] and improved upon in [108] using elementary operations called Givens rotations. Here, we present an algorithm that reduces the number of Givens rotations required by exploiting a symmetry in the definition of a Slater determinant.

We can prepare a Slater determinant by first preparing a computational basis vector (4.6) and then applying the basis transformation $\mathcal{U}$:

$$|\Psi_S\rangle = \mathcal{U} a_1^\dagger \cdots a_{N_f}^\dagger |\text{vac}\rangle, \qquad \mathcal{U} a_p^\dagger \mathcal{U}^\dagger = b_p^\dagger \quad \text{for } p = 1, \ldots, N_f. \tag{4.8}$$

The unitary $\mathcal{U}$ can be decomposed as a sequence of elementary operations called Givens rotations [108]:

$$\mathcal{U} = \mathcal{G}_1 \cdots \mathcal{G}_{N_G} \tag{4.9}$$

A Givens rotation $\mathcal{G}_{pq}(\theta, \varphi)$ has the following effect:

$$\begin{pmatrix} \mathcal{G} a_p^\dagger \mathcal{G}^\dagger \\ \mathcal{G} a_q^\dagger \mathcal{G}^\dagger \end{pmatrix} = G(\theta, \varphi) \begin{pmatrix} a_p^\dagger \\ a_q^\dagger \end{pmatrix}, \tag{4.10}$$

where

$$G(\theta, \varphi) = \begin{pmatrix} \cos\theta & -e^{i\varphi}\sin\theta \\ \sin\theta & e^{i\varphi}\cos\theta \end{pmatrix}. \tag{4.11}$$

The operation $\mathcal{G}_{pq}(\theta, \varphi)$ can be written in terms of fermionic operators (independent of the fermion-to-qubit mapping) as

$$\mathcal{G}_{pq}(\theta, \varphi) = \exp\left[i\varphi a_q^\dagger a_q\right] \exp\left[\theta(a_p^\dagger a_q - a_q^\dagger a_p)\right] \tag{4.12}$$

Under the JWT, it can be implemented with the circuit in Fig. 4.1

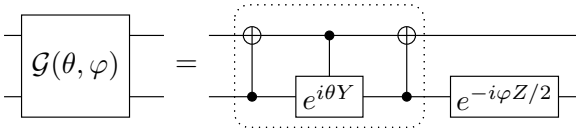


Figure 4.1: Quantum circuit for a Givens rotation on neighboring qubits: the part in the dotted box represents a rotation between the two states $|01\rangle$ and $|10\rangle$.

The decomposition (4.9) corresponds to a decomposition

$$U = G_{N_G} \cdots G_1, \tag{4.13}$$

where U is a unitary matrix that satisfies

$$QU^\dagger = (I \quad 0), \tag{4.14}$$

and the G_k are $N \times N$ matrices which act as (4.11) on two rows. The matrix on the right-hand side of (4.14) corresponds to a Slater determinant in the computational basis, and the Givens rotations actd on it to turn it into Q. We choose the G_k to always act on neighboring rows so that the corresponding gates act on neighboring qubits. The order of the matrices in (4.13) is reversed from that in (4.9) because the matrices G act on vectors of creation operators, while the operators $\mathcal{G}$ act directly on the original creation operators as in (4.10).

Now, we explain how to reduce the number N_G of rotations required by exploiting a symmetry; namely, the Slater determinant (4.7) remains unchanged up to a global phase under the transformation $Q \mapsto VQ$ for any $N_f \times N_f$ unitary V:

$$\left(\prod_{p=1}^{N_f} \sum_{q=1}^{N_f} V_{pq} b_q^\dagger \right) |\text{vac}\rangle = \det(V) |\Psi_S\rangle. \tag{4.15}$$

We can use V to zero out the elements of Q in its upper right corner; the resulting matrix can be decomposed with fewer Givens rotations.

Here is an example with $N = 6$ and $N_f = 3$:

$$\begin{aligned} Q \;\to\; & \begin{pmatrix} * & * & * & * & * & 0 \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix} \;\to\; \begin{pmatrix} * & * & * & * & * & 0 \\ * & * & * & * & * & 0 \\ * & * & * & * & * & * \end{pmatrix} \\ \to\; & \begin{pmatrix} * & * & * & * & 0 & 0 \\ * & * & * & * & * & 0 \\ * & * & * & * & * & * \end{pmatrix} \;=\; VQ\,, \end{aligned} \tag{4.16}$$

where $*$ represents an arbitrary matrix element, and the red-colored elements are zeroed out by rotating two neighboring rows together. This procedure does not change the Slater determinant and does not require any physical operation. The decomposition (4.13) can

then be found by zeroing out elements of VQ with rotations of adjacent columns:

$$VQ \to \begin{pmatrix} * & * & * & 0 & 0 & 0 \\ * & * & * & * & * & 0 \\ * & * & * & * & * & * \end{pmatrix} \to \begin{pmatrix} * & * & 0 & 0 & 0 & 0 \\ * & * & * & * & 0 & 0 \\ * & * & * & * & * & * \end{pmatrix} \to \begin{pmatrix} \lambda_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 \\ 0 & * & * & * & * & 0 \end{pmatrix}$$
$$\to \begin{pmatrix} \lambda_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 & 0 \end{pmatrix} \to \begin{pmatrix} \lambda_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & 0 & 0 \end{pmatrix} \to VQU^\dagger, \tag{4.17}$$

where the λ_j's are phase factors, i.e., $|\lambda_j| = 1$. They correspond to a global phase in the quantum state, and hence it is not necessary to bring them to 1. The blue-colored elements become phase factors or zero due to the orthonormality of the rows. Givens rotations on non-overlapping pairs of columns can be performed in parallel. The total number of Givens rotations needed is

$$N_G = NN_f - N_f(N_f - 1)/2 - N_f(N_f + 1)/2 = N_f(N - N_f), \tag{4.18}$$

and the circuit depth is $N - 1$. In the worst case, $N_f = N/2$, we require $N^2/4$ rotations, which for large N is about a $1/3$ reduction in the number of rotations we would need if we did not exploit the unitary symmetry.

In summary, we have described a method to prepare a Slater determinant (4.7) using two-qubit gates that act only on neighboring qubits. Our method can be broken into three steps:

1. Zero out the upper right matrix elements of Q using the freedom $Q \mapsto VQ$.
2. Diagonalize VQ using a sequence of Givens rotations as column transformations.
3. Find the quantum gates that correspond to the Givens rotations in Step 2 using the circuit in Fig. 4.1.

4.3.2 Preparing fermionic Gaussian states

Slater determinants can be viewed as special cases of a more general class of states, namely, eigenstates of quadratic Hamiltonians, also known as fermionic Gaussian states. In this section, we introduce quadratic Hamiltonians and explain how to prepare their eigenstates on a quantum computer.

4.3.2.1 Quadratic Hamiltonians

A *quadratic Hamiltonian* is a Hermitian operator containing only terms which are quadratic in the fermionic ladder operators:

$$H = \sum_{p,q=1}^{N} M_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q=1}^{N} (\Delta_{pq} a_p^\dagger a_q^\dagger - \Delta_{pq}^* a_p a_q), \tag{4.19}$$

where M and Δ are $N \times N$ matrices. Since H is Hermitian, we must have $M = M^\dagger$ and $\Delta = -\Delta^T$. Any quadratic Hamiltonian may be rewritten in the following form:

$$H = \sum_{p=1}^{N} \varepsilon_p b_p^\dagger b_p + \text{constant}, \tag{4.20}$$

where the ε_p are real numbers and the constant represents a shift in eigenvalues, i.e., it is a real or complex multiple of the identity operator (in this case, real). The $\{b_p\}$ are a new set of fermionic annihilation operators that satisfy the anticommutation relations (4.1) and (4.2). Since the fermionic number operators $\{b_p^\dagger b_p\}$ have eigenvalues 0 and 1, we can deduce the eigenvalues of H from the form (4.20). Up to the additive constant, they are sums of subsets of $\{\varepsilon_p\}_{p=1}^N$. The Hamiltonian H is diagonal in the basis determined by the $\{b_p\}$.

If $\Delta = 0$ in (4.19), then H conserves particle number. In this case, we can find the $\{b_p\}$ by diagonalizing the Hermitian matrix M. That is, we compute the matrix U such that

$$U M^T U^\dagger = \text{diag}(\varepsilon_1, \ldots, \varepsilon_N), \tag{4.21}$$

and then we have

$$\begin{pmatrix} b_1^\dagger \\ \vdots \\ b_N^\dagger \end{pmatrix} = U \begin{pmatrix} a_1^\dagger \\ \vdots \\ a_N^\dagger \end{pmatrix}. \tag{4.22}$$

Then, any N_f rows of U can be taken to be the matrix Q in (4.7), which determines a Slater determinant.

In the general case $\Delta \neq 0$, H does not conserve particle number. In this case the new creation operators $\{b_p^\dagger\}$ will need to be linear combinations of both the original creation

operators *and* the original annihilation operators:

$$\begin{pmatrix} b_1^\dagger \\ \vdots \\ b_N^\dagger \\ b_1 \\ \vdots \\ b_N \end{pmatrix} = W \begin{pmatrix} a_1^\dagger \\ \vdots \\ a_N^\dagger \\ a_1 \\ \vdots \\ a_N \end{pmatrix}, \tag{4.23}$$

where W is a $2N \times 2N$ matrix. In order for the $\{b_p\}$ to satisfy the anticommutation relations, W must be unitary. Furthermore, W must have the block form

$$W = \begin{pmatrix} W_1^* & W_2^* \\ W_2 & W_1 \end{pmatrix}, \tag{4.24}$$

where the fact that the $\{b_p\}$ satisfy the relations (4.1) and (4.2) imply that the $N \times N$ submatrices W_1 and W_2 satisfy

$$W_1 W_2^T + W_2 W_1^T = 0, \tag{4.25}$$

$$W_1 W_1^\dagger + W_2 W_2^\dagger = I. \tag{4.26}$$

We can always choose W so that $0 \leq \varepsilon_1 \leq \cdots \leq \varepsilon_N$.

4.3.2.2 Calculating W

In this section, we show how to calculate the matrix W in (4.23), which puts the Hamiltonian (4.19) into the diagonal form (4.20). First, we rewrite the Hamiltonian (4.19) in matrix form:

$$H = \frac{1}{2}\begin{pmatrix} a_1^\dagger & \cdots & a_N^\dagger & a_1 & \cdots & a_N \end{pmatrix} \begin{pmatrix} \Delta & M \\ -M^* & -\Delta^* \end{pmatrix} \begin{pmatrix} a_1^\dagger \\ \vdots \\ a_N^\dagger \\ a_1 \\ \vdots \\ a_N \end{pmatrix} + \text{constant}, \tag{4.27}$$

where the extra constant comes from the different ordering of the operators, i.e., the fact that

$$a_p a_p^\dagger = I - a_p^\dagger a_p, \tag{4.28}$$

from (4.2). Now, we introduce the Majorana fermion operators:

$$f_p = \frac{1}{\sqrt{2}}\big(a_p^\dagger + a_p\big), \quad f_{p+N} = \frac{i}{\sqrt{2}}\big(a_p^\dagger - a_p\big), \tag{4.29}$$

which satisfy the anticommutation relations

$$\{f_p, f_q\} = \delta_{pq}, \quad \text{for } p, q = 1, \ldots, 2N. \tag{4.30}$$

We can write the transformation (4.29) in matrix form:

$$\begin{pmatrix} f_1 \\ \vdots \\ f_{2N} \end{pmatrix} = \Omega \begin{pmatrix} a_1^\dagger \\ \vdots \\ a_N^\dagger \\ a_1 \\ \vdots \\ a_N \end{pmatrix}, \qquad \Omega = \frac{1}{\sqrt{2}} \begin{pmatrix} I & I \\ iI & -iI \end{pmatrix}. \tag{4.31}$$

In terms of the Majorana operators, the Hamiltonian (4.27) takes the form

$$H = \frac{i}{2}\mathbf{f}^T A \mathbf{f} + \text{constant}, \tag{4.32}$$

where $\mathbf{f} = (f_1 \ \cdots \ f_{2N})^T$, and A is the $2N \times 2N$ real antisymmetric matrix given by

$$A = -i\Omega^* \begin{pmatrix} \Delta & M \\ -M^* & -\Delta^* \end{pmatrix} \Omega^\dagger. \tag{4.33}$$

Since A is real and antisymmetric, it can be brought by an orthogonal matrix R into the following canonical form, which is equivalent to the Schur form up to a permutation of rows and columns:

$$RAR^T = \begin{pmatrix} 0 & \mathcal{E} \\ -\mathcal{E} & 0 \end{pmatrix}, \qquad \mathcal{E} = \operatorname{diag}(\varepsilon_1, \ldots, \varepsilon_N). \tag{4.34}$$

This procedure corresponds to a transformation of the Majorana operators:

$$\mathbf{f}' = R\mathbf{f}, \tag{4.35}$$

where the f'_p are a new set of Majorana operators which also satisfy the anticommutation relations (4.30). The corresponding transformation of the creation and annihilation operators is given by the matrix W that we are trying to calculate, which is related to R by

$$W = \Omega^\dagger R \Omega. \tag{4.36}$$

In summary, W can be calculated in four steps:

1. Write the quadratic Hamiltonian (4.19) into the matrix form (4.27).
2. Find the real antisymmetric matrix A using (4.33).
3. Calculate the orthogonal matrix R which brings A into the canonical form (4.34).
4. Calculate the matrix W using (4.36).

4.3.2.3 Preparing eigenstates

We can prepare eigenstates of the Hamiltonian (4.20) using the basis transformation $\mathcal{W}$ such that

$$\mathcal{W} a_p \mathcal{W}^\dagger = b_p, \qquad p = 1, \ldots, N, \tag{4.37}$$

where the $\{b_p\}$ are defined in (4.23). Since we choose W so that the $\{\varepsilon_p\}$ are all nonnegative, the ground state of H is

$$|\Psi_0\rangle = \mathcal{W}|\text{vac}\rangle \tag{4.38}$$

and the eigenstates of H take the form

$$\left(b_1^\dagger\right)^{i_1} \cdots \left(b_N^\dagger\right)^{i_N} |\Psi_0\rangle, \tag{4.39}$$

where $i_p \in \{0, 1\}$ for $p = 1, \ldots, N$. Therefore, we can prepare an eigenstate of H by applying $\mathcal{W}$ to a computational basis state (4.6).

Up to an overall phase, the unitary $\mathcal{W}$ is uniquely determined by the matrix W in (4.24);

actually, due to redundancy in W we only need its lower half,

$$W_L = (W_2 \quad W_1). \tag{4.40}$$

W_L describes how to write the new annihilation operators as linear combinations of the original ladder operators; W_2 gives the coefficients corresponding to creation operators while W_1 gives the coefficients corresponding to annihilation operators.

We show that using elementary operations on W_L, the unitary $\mathcal{W}$ can be decomposed as

$$\mathcal{W} = \mathcal{B}\mathcal{G}_1\mathcal{B}\mathcal{G}_2\mathcal{G}_3\mathcal{B}\cdots\mathcal{G}_{N_G}\mathcal{B}\cdot\mathcal{V} = \mathcal{U}\cdot\mathcal{V}, \tag{4.41}$$

where the $\mathcal{G}_j$ are Givens rotations, $\mathcal{V}$ is a product of Givens rotations, and $\mathcal{B} = a_N + a_N^\dagger$ is the particle-hole transformation on the last fermionic mode:

$$\mathcal{B}a_N\mathcal{B}^\dagger = a_N^\dagger, \tag{4.42}$$

$$\mathcal{B}a_p\mathcal{B}^\dagger = a_p \qquad \text{for } p = 1, \ldots, N-1. \tag{4.43}$$

Under the JWT, $\mathcal{B}$ is implemented easily by applying the Pauli X operator on the last qubit, since ladder operators corresponding to modes other than the last one do not act on the last qubit. Now, a Givens rotation takes the form

$$G = \left(\begin{array}{cc|cc} \cos\theta & -e^{i\varphi}\sin\theta & 0 & 0 \\ \sin\theta & e^{i\varphi}\cos\theta & 0 & 0 \\ \hline 0 & 0 & \cos\theta & -e^{-i\varphi}\sin\theta \\ 0 & 0 & \sin\theta & e^{-i\varphi}\cos\theta \end{array}\right), \tag{4.44}$$

because if we perform a rotation on creation operators, we need to perform the conjugated rotation on annihilation operators. The matrix representation of the particle-hole transformation is

$$B = B^\dagger = \begin{pmatrix} I - e_N e_N^T & e_N e_N^T \\ e_N e_N^T & I - e_N e_N^T \end{pmatrix}, \tag{4.45}$$

where $e_N = (0, \ldots, 0, 1)^T$ is a vector of length N. Multiplying a matrix on the right by $B^\dagger$ corresponds to swapping the N-th and $(2N)$-th columns.

Our goal now is to find a decomposition of a unitary U,

$$U = BG_{N_G} \cdots BG_3G_2BG_1B \tag{4.46}$$

such that

$$VW_LU^\dagger = (0 \quad I), \tag{4.47}$$

where V is an arbitrary $N \times N$ unitary matrix. The matrix on the right-hand side of (4.47) represents the original annihilation operators, and the matrices V and U act on it to turn it into W_L. We explain how to do this using an example where $N = 4$. First, we take V to be a product of Givens rotations to zero out some matrix elements on the left side of W_L:

$$VW_L = \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & * & * & * & * & 0 \\ 0 & 0 & * & * & * & * & * & * \\ 0 & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{array}\right), \tag{4.48}$$

where the blue zero in the upper right corner is automatically zeroed out due to condition (4.25). Then, we can use Givens rotations on neighboring columns as well as the transformation B to swap the N-th and last columns:

$$\begin{aligned} VW_L \to & \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & * & * & * & * \\ 0 & 0 & * & * & * & * & * & * \\ 0 & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{array}\right) \to \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & * & * & * & 0 \\ 0 & 0 & 0 & * & * & * & * & 0 \\ 0 & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{array}\right) \to \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & * & * \\ 0 & 0 & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{array}\right) \\ \to & \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & 0 & * & * & 0 \\ 0 & * & * & * & 0 & * & * & * \end{array}\right) \to \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & * & * & 0 & 0 & * & * \end{array}\right) \to \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 & 0 & 0 & 0 & 0 \end{array}\right) \\ \to & \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_4 \end{array}\right) \to VW_LU^\dagger\,. \end{aligned} \tag{4.49}$$

The red-colored matrix elements in the fourth column are always zeroed out by the particle-hole transformation B, and the other red-colored matrix elements on the left side are zeroed

out by the Givens rotations G. The red-colored elements on the right side become nonzero due to the particle-hole transformation B, and the blue-colored matrix elements are brought to zeros or phase factors automatically by the condition (4.25) or (4.26). The phase factors are brought to ones in the last step by single-qubit rotations. The total numbers of Givens rotations and particle-hole transformations in this step are

$$N_G = N(N-1)/2, \qquad N_B = N, \tag{4.50}$$

and the circuit depth is $2N - 1$.

We can rearrange (4.47) so that V acts as column transformations that can be implemented using the circuit in Fig. 4.1:

$$W_L = V^\dagger(0 \quad I)U = (0 \quad I)\begin{pmatrix} V^T & 0 \\ 0 & V^\dagger \end{pmatrix} U. \tag{4.51}$$

The matrix $\mathrm{diag}\big(V^T, V^\dagger\big)$ can be decomposed as Givens rotations and can be implemented with a circuit of depth $N - 1$ as described in Section 4.3.1.

In summary, we have described a method to prepare an arbitrary eigenstate of a quadratic Hamiltonian (4.19) using two-qubit gates that act only on neighboring qubits. Our method can be achieved in four steps:

1. Calculate the matrix W using the procedure described in Section 4.3.2.2.
2. Zero out the upper-left matrix elements of W_L with Givens rotations using the transformation $W_L \to VW_L$.
3. Zero out the remaining matrix elements on the left side of VW_L using the sequence (4.49),
4. Find the quantum gates corresponding to the sequences in Steps 2 and 3.

4.4 Conclusion

The simulation of electronic structure is one of the most anticipated applications of quantum computers. Improved simulation methods would allow us to design better materials, and we expect that quantum computers will simulate systems far out of the reach of classical computers. The first step of many quantum simulation algorithms is to initialize the computer in a precisely known quantum state. Here, we gave algorithms to prepare useful classes of starting states for electronic structure simulations, namely, Slater determinants

and fermionic Gaussian states. Our algorithms work for a linearly-connected architecture using only nearest-neighbor interactions. We implemented these algorithms in the free software package OpenFermion.

Code Availability

The algorithms described in this chapter are implemented in the free software package OpenFermion [17].

CHAPTER 5

Hartree-Fock on a Superconducting Qubit Processor

5.1 Introduction

The Hartree-Fock method is a variational method for approximately solving the electronic structure problem. Importantly, it can be performed efficiently on a classical computer. It is central to classical and quantum electronic structure calculations and often serves as a starting point for more sophisticated methods.

Here, we implemented the Hartree-Fock method as a variational quantum eigensolver (VQE) [20] on a quantum computer. We executed this VQE on Google's Sycamore processor [8] for linear hydrogen chains of length 6, 8, 10, and 12, and two pathways for diazene bond isomerization. Our largest simulations used a dozen qubits – twice as many as the largest prior quantum simulations of chemistry [23] – and required only nearest-neighbor coupling (depicted in Figure 5.1). Prior simulations of chemistry on superconducting qubit devices and trapped ion systems demonstrated the possibility of error mitigation through VQE [76, 23, 21, 111, 25, 112, 113], albeit on a small scale. We demonstrated that, within the model, achieving chemical accuracy through VQE is possible for larger problems when combined with effective error mitigation strategies. Furthermore, we argue that the circuit ansatz we used is especially appealing as a benchmark for chemistry.

The hydrogen chains are a common benchmark in electronic structure [114, 115, 116] and the diazene bond isomerization provides a system where the required accuracy is more representative of typical electronic structure problems and has been used as a benchmark for coupled cluster methods [117]. For the diazene isomerization our goal was to resolve the energetic difference between the transition states of two competing mechanisms, requiring accuracy of about 40 milliHartree. This objective differs from prior quantum simulations of chemistry which have focused on bond dissociation curves [76, 23, 21, 111].

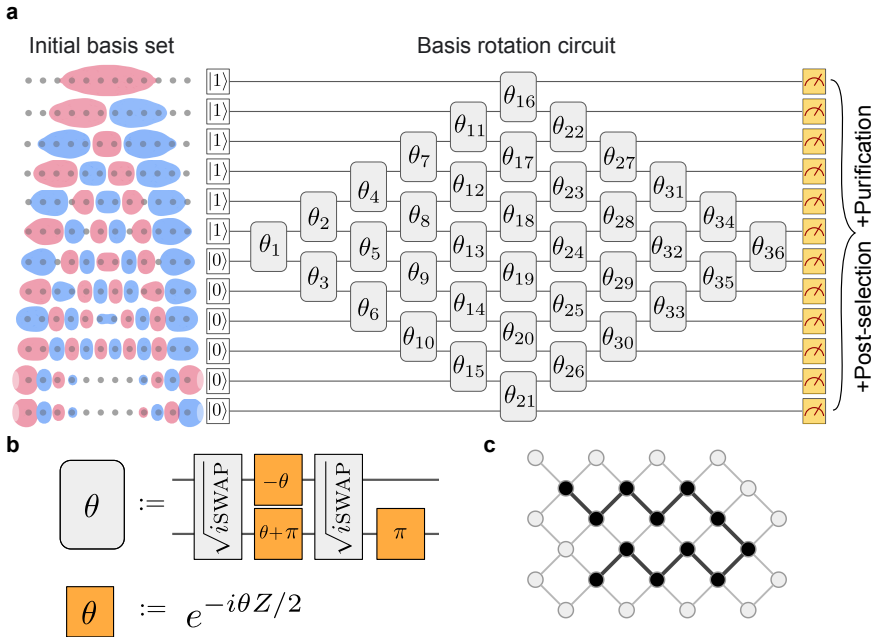


Figure 5.1: **Basis rotation circuit and compilation.** a) To the left of the circuit diagram are the initial orbitals for the H_{12} chain with atom spacings of 1.3 Å, obtained by diagonalizing the Hamiltonian ignoring electron-electron interactions. The circuit diagram depicts the basis rotation ansatz for a linear chain of twelve hydrogen atoms. Each grey box with a rotation angle θ represents a Givens rotation gate. b) Compilation of the Givens rotation gate to $\sqrt{i\text{SWAP}}$ gates and single-qubit gates that can be realized directly in hardware. The H_{12} circuit involves 72 $\sqrt{i\text{SWAP}}$ gates and 108 single-qubit Z rotation gates with a total of 36 variational parameters. c) Depiction of a twelve qubit line on a subgrid of the entire 54-qubit Sycamore device. All circuits only require gates between pairs of qubits which are adjacent in a linear topology.

One motivation for this work was to calibrate and validate the performance of the Sycamore processor in realizing an important algorithmic primitive for quantum chemistry and lattice model simulations. Our experiment was also appealing for benchmarking purposes since the circuits we explored generated highly entangled states but with special structure that enabled the efficient measurement of fidelity and the determination of systematic errors. Further motivation was to implement the largest variational quantum simulation of chemistry so that it is possible to better quantify the current gap between the capabilities of NISQ devices and real applications. Even though the Hartree-Fock ansatz can be efficiently simulated classically, the circuits in our experiment are more complex than prior experimental quantum simulations of chemistry. Finally, the structure of the Hartree-Fock state enabled us to sample the energy and gradients of the variational ansatz with fewer measurements than would typically be required, allowing us to focus on other aspects of quantum simulating chemistry at scale, such as the effectiveness of various types of error

mitigation. Thus, our choice to focus on Hartree-Fock for this experiment embraces the notion that we should work towards valuable quantum simulations of chemistry by first scaling up important components of the exact solution (e.g., error-mitigation strategies and basis rotations) in a fashion that enables us to completely understand and perfect those primitives.

5.2 Background

This section provides background knowledge in quantum chemistry. It draws heavily from Chapter 2 of [118].

5.2.1 The electronic structure problem

The central problem in electronic structure is to solve the non-relativistic time-independent Schrödinger equation

$$H|\Psi\rangle = E|\Psi\rangle \tag{5.1}$$

where H is the Hamiltonian for a system of nuclei and electrons. Since nuclei are much heavier than electrons, it is a good approximation to assume that they are fixed. This is called the Born-Oppenheimer approximation, and it is central to quantum chemistry. In the Born-Oppenheimer approximation, the Hamiltonian takes the form

$$H = -\sum_i \frac{1}{2}\nabla_i^2 - \sum_{i,A} \frac{Z_A}{|\mathbf{r}_i - \mathbf{R}_A|} + \sum_{i<j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}, \tag{5.2}$$

where $\mathbf{r}_i$ is the position of the i-th electron, $\mathbf{R}_A$ is the position of the A-th nucleus, Z_A is the charge of the A-th nucleus, and the Laplacian operator ∇_i^2 involves differentiation with respect to the coordinates $\mathbf{r}_i$. A solution to the Schrödinger equation is an eigenvector $|\Psi\rangle$ of the linear operator H with eigenvalue E. This eigenvector is a wavefunction

$$|\Psi\rangle = |\Psi(\{\mathbf{r}_i\})\rangle \tag{5.3}$$

that depends on the electronic coordinates. The eigenvector with lowest energy is called the ground state.

5.2.2 The Pauli exclusion principle

While the Hamiltonian (5.2) depends only on the spatial coordinates $\{\mathbf{r}_i\}$, the complete description of an electron must also include its *spin*. This is an additional degree of freedom that is discrete and takes the values $\pm\frac{1}{2}$. We denote the spin of the i-th electron by ω_i, and denote the collection of the three spatial coordinates and one spin coordinate by $\mathbf{x}_i$:

$$\mathbf{x}_i = (\mathbf{r}_i, \omega_i). \tag{5.4}$$

Thus, the full wavefunction for a system of electrons depends on the coordinates $\mathbf{x}_i$:

$$|\Psi\rangle = |\Psi(\{\mathbf{x}_i\})\rangle. \tag{5.5}$$

The Pauli exclusion principle states that the wavefunction for a system of electrons must be antisymmetric with respect to the exchange of the coordinates of two electrons:

$$\Psi(\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_\eta) = -\Psi(\mathbf{x}_1, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_\eta) \tag{5.6}$$

This is an independent postulate of quantum mechanics.

5.2.3 Spatial orbitals and spin orbitals

The electronic wavefunction is usually written in terms of Slater determinants. To define Slater determinants, we need to first define orbitals.

An *orbital* is a wavefunction for a single electron. A *spatial orbital* $\psi_p(\mathbf{r})$ describes the spatial distribution of an electron. Spatial orbitals usually form an orthonormal set:

$$\int \psi_p^*(\mathbf{r})\psi_q(\mathbf{r})d\mathbf{r} = \delta_{pq}. \tag{5.7}$$

If a set of spatial orbitals $\{\psi_p\}$ were complete, then any arbitrary function could be expanded as a linear combination of them. In general, the set would have to be infinite in order to be complete. In practice, we choose a finite set and work within the space spanned by that set.

A *spin orbital* $\chi(\mathbf{x})$ describes both the spatial distribution and the spin of an electron. The spin of an electron can be completely described using the two orthonormal spin functions $\alpha(\omega)$ and $\beta(\omega)$, representing spin up and spin down. A spin orbital is constructed by

multiplying a spatial orbital by one of the spin functions:

$$\chi(\mathbf{x}) = \psi(\mathbf{r})\alpha(\omega) \qquad \text{or} \qquad \chi(\mathbf{x}) = \psi(\mathbf{r})\beta(\omega). \tag{5.8}$$

If the spatial orbitals are orthonormal, then so are the spin orbitals.

5.2.4 Slater determinants

A spin orbital is a wavefunction for a single electron. Wavefunctions for multiple electrons are constructed from spin orbitals by multiplying them and taking appropriate linear combinations to ensure that the result is antisymmetric. For the case of two electrons in two spin-orbitals χ_p and χ_q, the appropriate linear combination is

$$\Psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2}}(\chi_p(\mathbf{x}_1)\chi_q(\mathbf{x}_2) - \chi_q(\mathbf{x}_1)\chi_p(\mathbf{x}_2)). \tag{5.9}$$

This function is antisymmetric with respect to the exchange of $\mathbf{x}_1$ and $\mathbf{x}_2$, so it is a proper wavefunction for two electrons. This wavefunction can be rewritten as a determinant

$$\Psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2}} \begin{vmatrix} \chi_p(\mathbf{x}_1) & \chi_q(\mathbf{x}_1) \\ \chi_p(\mathbf{x}_2) & \chi_q(\mathbf{x}_2) \end{vmatrix}. \tag{5.10}$$

The generalization for a system of η electrons is

$$\Psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\eta) = \frac{1}{\sqrt{\eta!}} \begin{vmatrix} \chi_p(\mathbf{x}_1) & \chi_q(\mathbf{x}_1) & \cdots & \chi_r(\mathbf{x}_1) \\ \chi_p(\mathbf{x}_2) & \chi_q(\mathbf{x}_2) & \cdots & \chi_r(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_p(\mathbf{x}_\eta) & \chi_q(\mathbf{x}_\eta) & \cdots & \chi_r(\mathbf{x}_\eta) \end{vmatrix}. \tag{5.11}$$

This wavefunction is antisymmetric and is called a Slater determinant. Since the form of the determinant is determined by its diagonal elements, it is common to introduce a shorthand notation for it, which includes the normalization factor:

$$\Psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\eta) = |\chi_p(\mathbf{x}_1)\chi_q(\mathbf{x}_2)\cdots\chi_r(\mathbf{x}_\eta)\rangle. \tag{5.12}$$

If we assume that the coordinate labels are in the order $\mathbf{x}_1, \ldots \mathbf{x}_\eta$, then this can be further shortened to

$$\Psi(\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_\eta) = |\chi_p\chi_q\cdots\chi_r\rangle. \tag{5.13}$$

This wavefunction has η electrons occupying the η spin-orbitals $\chi_p, \chi_q, \ldots, \chi_r$.

5.2.5 Second quantization

Let $\{\chi_p\}_{p=1}^N$ be a set of spin-orbitals. We can identify each Slater determinant constructed from these spin-orbitals with an *occupation-number vector* $|\mathbf{k}\rangle$,

$$|\mathbf{k}\rangle = |k_1, k_2, \ldots, k_N\rangle, \quad k_p = \begin{cases} 1 & \text{if } \chi_p \text{ is occupied} \\ 0 & \text{if } \chi_p \text{ is unoccupied} \end{cases}. \tag{5.14}$$

By χ_p being occupied we mean that it appears in the Slater determinant (5.13). The occupation number vectors form an orthonormal basis for a 2^N-dimensional abstract Hilbert space called *Fock space*. The state with no occupied orbitals has no electrons and is called the *vacuum state*:

$$|\text{vac}\rangle = |\mathbf{0}\rangle. \tag{5.15}$$

Second quantization is a formalism in which all operators and states are expressed in terms of creation and annihilation operators. The annihilation operators $\{a_p\}_{p=1}^N$ are defined by the relations

$$a_p|k_1, \ldots, k_{p-1}, 1, k_{p+1}, \ldots, k_N\rangle = (-1)^{\sum_{s=0}^{p-1} k_s}|k_1, \ldots, k_{p-1}, 0, k_{p+1}, \ldots, k_N\rangle, \tag{5.16}$$

$$a_p|k_1, \ldots, k_{p-1}, 0, k_{p+1}, \ldots, k_N\rangle = 0. \tag{5.17}$$

The creation operators $\{a_p^\dagger\}$ are the Hermitian conjugates. The creation and annihilation operators satisfy the anticommutation relations

$$a_p a_q + a_q a_p = 0, \tag{5.18}$$

$$a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}. \tag{5.19}$$

Additional properties were given in Section 4.2. The occupation-number vectors can be obtained by applying the creation operators to the vacuum state:

$$|\mathbf{k}\rangle = \left(a_1^\dagger\right)^{k_1} \cdots \left(a_N^\dagger\right)^{k_N}|\text{vac}\rangle. \tag{5.20}$$

In equations (5.11-5.13) the Slater determinant is written in terms of the spin orbitals $\{\chi_p\}_{p=1}^N$. We obtain new sets of Slater determinants by considering different sets of spin orbitals. A new set of spin orbitals $\{\tilde{\chi}_p\}_{p=1}^N$ is formed by performing a unitary transformation

on the original set of orbitals,

$$\tilde{\chi}_p = \sum_{q=1}^{N} Q_{pq}\chi_q, \tag{5.21}$$

where Q is a unitary matrix. In Fock space, this corresponds to a transformation of the creation operators $a_p^\dagger$ into a new set of creation operators $b_p^\dagger$,

$$b_p^\dagger = \sum_{q=1}^{N} Q_{pq}a_q^\dagger. \tag{5.22}$$

The new Slater determinants correspond to vectors of the form

$$\left(b_1^\dagger\right)^{k_1} \cdots \left(b_N^\dagger\right)^{k_N} |\text{vac}\rangle. \tag{5.23}$$

This explains the definition of Slater determinants given in Section 4.3.1.

In second quantization, the electronic Hamiltonian takes the form

$$H = \sum_{pq} h_{pq}a_p^\dagger a_q + \frac{1}{2}\sum_{pqrs} g_{pqrs}a_p^\dagger a_q^\dagger a_r a_s \tag{5.24}$$

where

$$h_{pq} = \int \chi_p^\star(\mathbf{x})\left(-\frac{1}{2}\nabla^2 - \sum_A \frac{Z_A}{|\mathbf{r}-\mathbf{R}_A|}\right)\chi_q(\mathbf{x})d\mathbf{x} \tag{5.25}$$

$$g_{pqrs} = \iint \frac{\chi_p^*(\mathbf{x}_1)\chi_q^*(\mathbf{x}_2)\chi_s(\mathbf{x}_1)\chi_r(\mathbf{x}_2)}{|\mathbf{r}_1-\mathbf{r}_2|}d\mathbf{x}_1 d\mathbf{x}_2. \tag{5.26}$$

These expressions for the coefficients h_{pq} and g_{pqrs} are determined by the requirement that the matrix elements of the second-quantized Hamiltonian (5.24) between occupation number vectors be equal to the matrix elements of the original Hamiltonian (5.2) between the corresponding Slater determinants.

5.2.6 The Hartree-Fock method

The Hartree-Fock method approximates the ground state energy of the electronic Hamitonian as the lowest energy among all Slater determinants. In second quantization, a Slater

determinant $|\Psi_S\rangle$ has the form (5.23), so it can be written as

$$|\Psi_S\rangle = b_1^\dagger \cdots b_\eta^\dagger |\text{vac}\rangle, \qquad b_p^\dagger = \sum_{q=1}^{N} Q_{pq} a_q^\dagger, \tag{5.27}$$

where Q is a unitary matrix and N is the total number of spin orbitals. This state can be obtained by starting with an occupation number vector and changing basis:

$$|\Psi_S\rangle = \mathcal{U} a_1^\dagger \cdots a_\eta^\dagger |\text{vac}\rangle, \qquad \mathcal{U} a_p^\dagger \mathcal{U}^\dagger = b_p^\dagger \quad \text{for } p = 1, \ldots, \eta. \tag{5.28}$$

A result due to Thouless [119] shows that the basis transformation $\mathcal{U}$ takes the form $\mathcal{U}(Q)$, where we define

$$\mathcal{U}(e^\kappa) = \exp\left(\sum_{p,q=1}^{N} \kappa_{pq} a_p^\dagger a_q \right). \tag{5.29}$$

The Hartree-Fock method works by treating the entries of κ as variational parameters to obtain a parameterization of the space of Slater determinants given by

$$|\kappa\rangle = \mathcal{U}(e^\kappa) a_1^\dagger \cdots a_\eta^\dagger |\text{vac}\rangle. \tag{5.30}$$

The Hartree-Fock state $|\psi_{\text{HF}}\rangle$ is the lowest energy Slater determinant, so it can be expressed as

$$|\psi_{\text{HF}}\rangle = |\kappa^\star\rangle, \qquad \kappa^\star = \text{argmin}_\kappa \langle \kappa | H | \kappa \rangle. \tag{5.31}$$

The Hartree-Fock method can be performed efficiently on a classical computer using an iterative method.

An important property of the basis change unitary (5.29) is that consecutive basis changes can be concatenated into a single basis change. Specifically, for antihermitian matrices α and β,we have

$$\mathcal{U}(e^\alpha)\mathcal{U}(e^\beta) = \mathcal{U}(e^\alpha e^\beta). \tag{5.32}$$

5.3 Methods

5.3.1 Representation

We represented the electronic Hamiltonian in second quantization. For our initial orbitals, we used what are commonly referred to as "core orbitals," depicted for H_{12} on the left side of Figure 5.1a. The Slater determinants formed from these orbitals are eigenfunctions of the electronic Hamiltonian omitting the electron-electron interaction term, the last term in (5.2). These orbitals are easy to compute because omitting the electron-electron interaction term results in a quadratic Hamiltonian in second quantization.

We modeled hydrogen chains of length N with N qubits. Our simulations required N qubits to simulate $2N$ spin-orbitals due to the constraint that the spin-up orbitals have the same spatial wavefunction as the spin-down orbitals. For diazene we required 10 qubits after pre-processing. An initial guess for the Hartree-Fock state, from which we can optimize the orbitals, was obtained by filling the lowest energy η orbitals, where η is the number of electrons.

5.3.2 Circuit ansatz

The Hartree-Fock ansatz is given by (5.30). Under the Jordan-Wigner Transformation (see Section 4.2.2), the occupation number vector $a_1^\dagger \cdots a_\eta^\dagger |\mathrm{vac}\rangle$ corresponds to a computation basis vector. We implement the unitary $\mathcal{U}(e^\kappa)$ using the decomposition into Givens rotations described in Section 4.3.1. The Givens rotation gates were implemented by decomposition into two $\sqrt{i\mathrm{SWAP}}$ gates and three Rz gates. In Figure 5.1, we depict the basis change circuit for the H_{12} chain, which has a diamond-shaped structure.

5.3.3 Energy measurement

The average energy of any molecular system can be evaluated with knowledge of the one-particle reduced density matrix (1-RDM), $\langle a_p^\dagger a_q \rangle$, and the two-particle reduced density matrix (2-RDM), $\langle a_p^\dagger a_q^\dagger a_r a_s \rangle$. In general, it is not possible to exactly reconstruct the 2-RDM from knowledge of just the 1-RDM. However, for single-Slater determinants (as in our Hartree-Fock experiment), the 2-RDM is completely determined by the 1-RDM [120]:

$$\langle a_p^\dagger a_q^\dagger a_r a_s \rangle = \langle a_p^\dagger a_s \rangle \langle a_q^\dagger a_r \rangle - \langle a_q^\dagger a_s \rangle \langle a_p^\dagger a_r \rangle. \tag{5.33}$$

Thus, in our experiment we only needed to sample the 1-RDM to estimate the energy. As the 2-RDM has quadratically more elements than the 1-RDM, this approach is a significant

simplification. In Appendix I we describe the protocol we used to measure the 1-RDM using $N + 1$ distinct circuits. For each circuit, we performed 250,000 measurements.

5.3.4 Error mitigation

The unitary (5.29) conserves particle number, which in the qubit picture corresponds to Hamming weight. Since our initial state has a well-defined Hamming weight equal to $N/2$, the ansatz state $|\kappa\rangle$ should also the same well-defined Hamming weight. The measurement protocol described in Appendix I allows the Hamming weight to be measured simultaneously with the 1-RDM elements. As our first error mitigation strategy, we discarded bitstrings that did not have the correct Hamming weight.

In addition to post-selection on Hamming weight, we performed another error mitigation technique, pure-state projection using a method known as McWeeny purification [120]. This procedure uses the fact that the 1-RDM for any single-Slater determinant wavefunction $|\psi_\kappa\rangle$ is idempotent, that is, it has eigenvalues that are either 0 or 1 [121]. Due to errors, the measured 1-RDM will not be exactly idempotent. McWeeney purification is an iterative procedure that projects the measured (noisy) 1-RDM onto the set of idempotent matrices. Letting D_n denote the matrix in the n-th iteration, the procedure is defined by

$$D_{n+1} = 3D_n^2 - 2D_n^3. \tag{5.34}$$

Each iteration brings the eigenvalues closer to 0 and 1, and the procedure converges quadratically.

5.3.5 Optimization

As discussed in Chapter 2, the choice of algorithm for optimizing the variational parameters in VQE can have a great effect on overall performance. In this work, we used a second-order method of Newton-Raphson type similar to previous methods discussed in the literature for orbital optimization [122, 123]. The special structure of Slater determinants enabled the gradient and Hessian to be easily measured, thus making this method especially suitable.

In each iteration of the optimizer, the update to our current quantum state $|\psi\rangle$ takes the form

$$|\psi\rangle \to \mathcal{U}(e^R)|\psi\rangle = e^{\hat{R}}|\psi\rangle, \qquad \hat{R} = \sum_{p,q=1}^{N} R_{pq} a_p^\dagger a_q. \tag{5.35}$$

The energy of the updated state is a function of the variables R_{pq}:

$$E = \langle\psi|e^{-\hat{R}}He^{\hat{R}}|\psi\rangle = \langle\psi|\left(H + [H,\hat{R}] + \frac{1}{2}[[H,\hat{R}],\hat{R}] + \cdots\right)|\psi\rangle. \tag{5.36}$$

At a local minimum, we would have

$$\frac{\partial E}{\partial R_{pq}} = 0. \tag{5.37}$$

The update vector $\mathbf{r}$ containing the values of R_{pq} to use is calculated by solving the augmented Hessian matrix equation

$$\begin{pmatrix} 0 & \mathbf{g}^\dagger \\ \mathbf{g} & \mathbf{B} \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{r} \end{pmatrix} = \epsilon \begin{pmatrix} 1 \\ \mathbf{r} \end{pmatrix}, \tag{5.38}$$

where

$$\mathbf{g}_{pq} = \left.\frac{\partial E}{\partial R_{pq}}\right|_{R=0} \tag{5.39}$$

$$\mathbf{B}_{pq,rs} = \left.\frac{\partial^2 E}{\partial R_{pq}\partial R_{rs}}\right|_{R=0}. \tag{5.40}$$

The resulting update vector is a solution to the Newton-Raphson equation with a level shift,

$$\mathbf{g} + (\mathbf{B} - \epsilon)\mathbf{r} = 0. \tag{5.41}$$

From (5.36) we see that

$$\mathbf{g}_{pq} = \langle\psi|[H, a_p^\dagger a_q]|\psi\rangle, \tag{5.42}$$

$$\mathbf{B}_{pq,rs} = \langle\psi|[[H, a_p^\dagger a_q], a_r^\dagger a_s]|\psi\rangle. \tag{5.43}$$

These quantities can be computed directly from the 1-RDM. To see this, recall from Section 5.3.3 that any two-body term of the form $\langle a_p^\dagger a_q^\dagger a_r a_s\rangle$ can be computed from the 1-RDM. Therefore, the only terms in $[H, a_p^\dagger a_q]$ that could potentially present a problem are three-body terms of the form

$$[a_p^\dagger a_q^\dagger a_r a_s, a_i^\dagger a_j] = a_p^\dagger a_q^\dagger a_r a_s a_i^\dagger a_j - a_i^\dagger a_j a_p^\dagger a_q^\dagger a_r a_s. \tag{5.44}$$

However, this is really a two-body term in disguise, because the term on left can be made into the term on the right by pulling $a_i^\dagger$ and a_j through the other operators, with two-body

terms potentially materializing according to the fermionic anticommutation relations. The result is that the three-body terms cancel out (or are annihilated) and only two-body terms remain. The same reasoning shows that the double commutator is also actually a two-body operator.

Once the update vector $\mathbf{r}$ is computed, the quantum state is updated according to (5.35). Note that since consecutive basis change circuits can be concatenated according to (5.32), the circuit depth stays constant.

5.4 Results

As a benchmark, we studied symmetrically stretched hydrogen chains of length 6, 8, 10, and 12 atoms. The results are shown in Figure 5.2. The initial parameters were set to the parameters obtained by solving the Hartree-Fock equations on a classical computer. The data from the quantum computer is plotted along with classical Hartree-Fock results, showing successively improving agreement as we added post-selection, then purification, and finally variational relaxation. The 6- and 8-qubit data achieved chemical accuracy after VQE, and even the 12-qubit data followed the expected energy closely. The error data in Figure 5.2b and the other inserts show a consistent decrease in error by a factor of about 100 when using these protocols. Figure 5.2c details the significant decrease in error using a modest number of VQE iterations.

A fidelity witness can be efficiently computed from the experimental data [124]; see Appendix J. This value is a lower bound to the true fidelity, and thus potentially loose when the fidelity is small. However, Figure 5.2b demonstrates that this value generally tracks the measured errors. Table 5.1 shows how fidelity increased as we added various forms of error mitigation, starting on the left column where the optimal angles were computed classically. Uncertainties in the last digit, indicated in the parenthesis, were calculated by the procedure described in Appendix I.5. The first column of Table 5.1 is an estimate of the fidelity based on multiplying the fidelity for all the gates and readout assuming 99.5% fidelity for single qubit gates, 99% fidelity for two-qubit gates, and 97% fidelity for readout. We see that this estimate qualitatively follows the "raw" fidelity witness estimates except when the witness value is very small. For all hydrogen systems studied, we observed drastic fidelity improvements with combined error mitigation.

Diazene isomerization. We simulated two isomerization pathways for diazene, marking the first time that a chemical reaction mechanism has been modelled using a quantum computer. It is known that Hartree-Fock theory reverses the order of the transition states; however, here we focused on the accuracy of the computation with respect to the simu-

system	estimate	raw	+ps	+pure	+VQE
H_6	0.571	0.674(2)	0.906(2)	0.9969(1)	0.99910(9)
H_8	0.412	0.464(2)	0.827(2)	0.9879(3)	0.99911(8)
H_{10}	0.277	0.316(2)	0.784(3)	0.9704(5)	0.9834(4)
H_{12}	0.174	0.010(2)	0.654(3)	0.9424(9)	0.9913(3)

Table 5.1: ***Average fidelity lower bounds for hydrogen chain calculations***. We report values of the fidelity witness, averaged across H-H separations of $\{0.5, 0.9, 1.3, 1.7, 2.1, 2.5\}$ Å, starting from circuits with the theoretically optimal variational parameters (κ). "estimate" corresponds to an estimate of the fidelity derived by multiplying gate errors assuming 0.5 percent single-qubit gate error, 1 percent two-qubit gate error and 3 percent readout error. "Raw" corresponds to fidelities from constructing the 1-RDM without any error mitigation. "+ps" corresponds to fidelities from constructing the 1-RDM with post-selection on particle number. "+pure" corresponds to fidelities from constructing the 1-RDM with post-selection and applying purification as post-processing. Finally, "+VQE" corresponds to fidelities from using all previously mentioned error mitigation techniques in conjunction with variational relaxation. Note that for small values (such as the "raw" value for H_{12}) we expect the fidelity lower-bound is more likely to be loose.

lated model. Correctly identifying this pathway requires resolving the energy gap of 40 milliHartree between the two transition states. The pathways correspond to the motion of the hydrogen in the process of converting *cis*-diazene to *trans*-diazene. One mechanism is in-plane rotation of a hydrogen and the other is an out-of-plane rotation corresponding to rotation of the HNNH dihedral angle. Figure 5.3 contains VQE optimized data simulating nine points along the reaction coordinates for in-plane and out-of-plane rotation of hydrogen. For all points along the reaction coordinate the initial parameter setting was the solution to the Hartree-Fock equations. VQE produced 1-RDMs with average fidelity greater than 0.98 after error-mitigation. Once again, we see that our full error mitigation procedure significantly improves the accuracy of our calculation. Our VQE calculations on diazene predicted the correct ordering of the transition states within the chemical model with an energy gap of 41 ± 6 milliHartree; the true gap is 40.2 milliHartree.

5.5 Conclusion

In this work we took a step towards answering the question of whether NISQ computers can offer quantum advantage for chemical simulation by studying VQE performance on basis rotation circuits that are widely used in quantum algorithms for fermionic simulation. The considered ansatz afforded ways to minimize the resource requirements for VQE and study device performance for circuits that are similar to those needed for full Hamiltonian

simulation. These basis rotation circuits also made an attractive benchmark due to their prevalence, optimal known compilation, the ability to extract fidelity and fidelity witness values and the fact that they parameterize a continuous family of analytically solvable circuits demonstrating a high degree of entanglement. The circuits also serve as a natural progression towards more correlated ansatze such as a generalized swap network [58] or a non-particle conserving circuit ansatz followed by particle number projection.

We demonstrated the performance of two error mitigation techniques on basis rotation circuit fidelity. The first is post-selection on total occupation number when measuring all elements of the 1-RDM. This step was accomplished by permuting the basis rotation circuit such that all measurements involved estimating nearest-neighbor observables and measuring each pair of observables such that the total occupation number is preserved. The second is the application of McWeeny purification as a post-processing step. The energy improvements from projecting back to the pure-state N-representable manifold was evidence that generalized pure-state N-representability conditions would be instrumental in making NISQ chemistry computations feasible. This fact underscores the importance of developing procedures for applying pure-state N-representability conditions in a more general context. The post-selection and RDM measurement techniques can be generalized to measuring all 1-RDM and 2-RDM elements when considering a less restrictive circuit ansatz by permuting the labels of the fermionic modes. For ansatzes such as the generalized swap network [58], the circuit structure would not change, only the rotation angles. Thus, the measurement schemes presented here are applicable in the more general case. Furthermore, it is important to understand the performance of these error mitigation techniques when combined with alternatives such as noise extrapolation [125].

Finally, we were able to show further evidence that variational relaxation effectively mitigates coherent errors arising in implementation of physical gates. The performance of our problem specific optimization strategy motivates the study of iterative wavefunction constructions [126] in a more general setting. The combination of these error mitigation techniques with VQE unambiguously resolved a chemical mechanism within the model chemistry using a quantum computation. It is still an open question whether NISQ devices will be able to simulate challenging quantum chemistry systems and it is likely that major innovations would be required. However, we find the accuracy of these experiments and the effectiveness of these error-mitigation procedures to be an encouraging signal of progress in that direction.

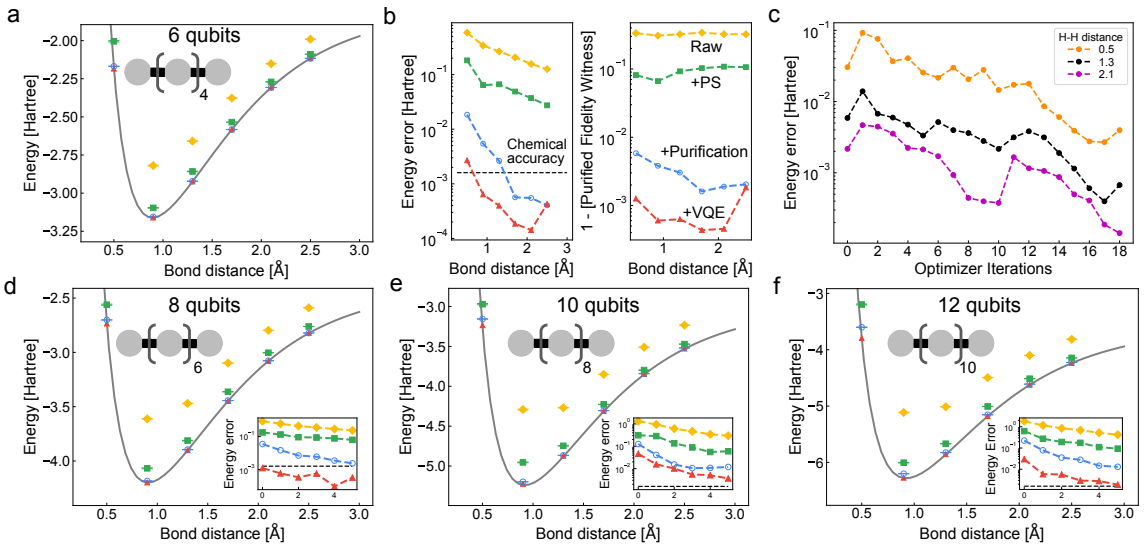


Figure 5.2: **Static and VQE performance on hydrogen chains.** Binding curve simulations for H_6, H_8, H_{10}, and H_{12} with various forms of error mitigation. Subfigures (a, d, e, f) compare Sycamore's raw performance (yellow diamonds) with post-selection (green squares), purification (blue circles), and error mitigated combined with variational relaxation (red triangles). For all hydrogen systems the raw data at 0.5 Å bond length is off the top of the plot. The yellow, green, and blue points were calculated using the optimal basis rotation angles computed from a classical simulation; thus, the variational optimization shown here is only used to correct systematic errors in the circuit realization. Subfigure (b) contains the absolute error and infidelity for the H_6 system. For all points we calculated a fidelity witness as described in Appendix J. The error bars for all points were computed by estimating the covariance between simultaneously measured sets of 1-RDM elements and resampling those elements under a multivariate Gaussian model. Energies from each sample were tabulated and the standard deviation is used as the error bar. The "+PS" means applying post-selection to the raw data, "+Purification" means applying post-selection and McWeeny purification, and "+VQE" means post-selection, McWeeny purification, and variational relaxation. Subfigure (c) contains optimization traces for three H_6 geometries (bond distances of 0.5 Å, 1.3 Å, and 2.1 Å). All optimization runs used between 18 and 30 iterations. The lowest energy solution from the optimization trace was reported.

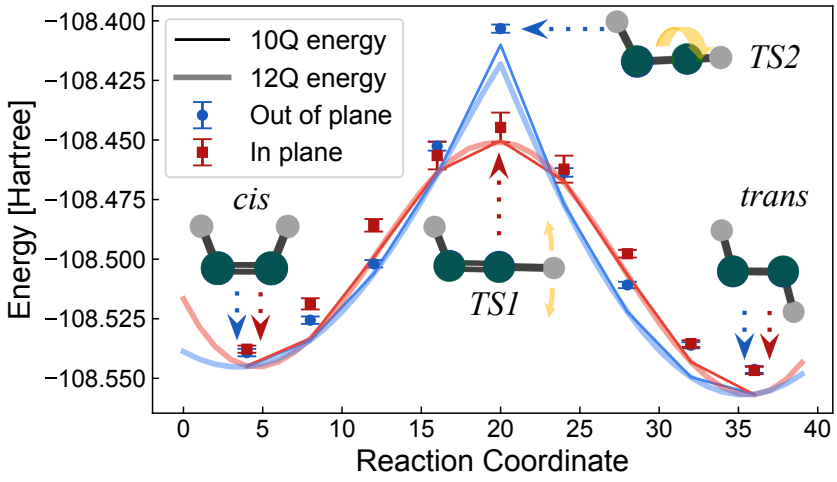


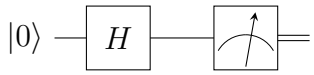
Figure 5.3: **VQE performance on distinguishing the mechanism of diazene isomerization.** Hartree-Fock curves for diazene isomerization between *cis* and *trans* configurations. $TS1$ and $TS2$ are the transition states for the in-plane and out-of-plane rotation of the hydrogen, respectively. The yellow arrows on $TS1$ and $TS2$ indicate the corresponding reaction coordinate. The solid curve is the energy obtained from optimizing a 10-qubit problem generated by freezing the core orbitals generated from two self-consistent-field cycles. The transparent lines of the same color are the full 12 qubit system indicating that freezing the lowest two levels does not change the characteristics of the model chemistry. Nine points along the reaction paths are simulated on Sycamore using VQE. We allowed the optimizer 30 iterations for all points except the fifth and sixth points from the left of the in-plane rotation curve, for which we allowed 60 iterations. The error bars for all points were computed by estimating the covariance between simultaneously measured sets of 1-RDM elements and resampling those elements under a multivariate Gaussian model. Energies from each sample were tabulated and the standard deviation is used as the error bar. No purification was applied for the computation of the error bars. If purification is applied the error bars become smaller than the markers. Each basis rotation for diazene contains 50 $\sqrt{i\text{SWAP}}$ gates and 80 Rz gates.

CHAPTER 6

Generating Certified Random Numbers on a Superconducting Qubit Processor

6.1 Introduction

Randomness is a valuable resource with many applications, including randomized algorithms, statistical sampling, sortition, and the generation of cryptographic keys. For most purposes requiring random numbers, it suffices to generate them using a standard pseudo-random number generator like the GNU/Linux special device file `/dev/urandom`. However, for some applications, such as the generation of cryptographic keys and the implementation of lotteries, true randomness is desirable. In classical physics, there is no such thing as true randomness: The outcome of any experiment can, in principle, be determined from the initial conditions. On the other hand, quantum physical systems exhibit true randomness. For example, the following quantum circuit, if executed correctly, produces an unbiased coin flip:



So an operator of a quantum device can plausibly claim the ability to generate truly random numbers. However, this leaves open the question of whether a skeptical client can *certify* that the claimed output of a random quantum process was actually produced through that process. This issue of certification has been widely studied in the general setting of device-independent random number generation. In this setting, the user of a quantum device purporting to produce random numbers makes no assumptions about the inner workings of the device, but is able to gain confidence in the randomness of its output by performing only classical communication and computation. The first protocol for this task was proposed by Colbeck in 2006 [11]. It was followed up by papers that provided security proofs of

steadily increasing degrees of security and efficiency [127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137]. In the past few years, such protocols have been realized in experiment [138, 139, 140, 141, 142]. These protocols are more properly termed randomness *expansion*, since they require a small initial random seed.

The protocols for device-independent random number generation require the client to interact with two separate quantum devices in order to produce Bell inequality violations [143]. The client must be able to verify through precise distance and timing measurements that the devices could not have communicated in order to produce the violations. Such verification is impossible if the client is receiving the random numbers through communication with a distant server through the Internet. This constitutes a major practical limitation of these protocols. However, it is still possible to devise protocols that work in this situation, if one is willing to accept computational hardness assumptions and impose computational limits on the server. The first such protocol was proposed by Brakerski et al[144], based on the quantum hardness of the learning with errors (LWE) problem [145]. This protocol requires the server to compute a lattice-based one-way function in superposition, as well as to store a quantum state in memory while exchanging messages with the client, two feats that are not possible with NISQ devices accessed through the Internet. An alternative protocol has been given by Scott Aaronson [146], based on the hardness of sampling from the output distribution of random quantum circuits. This proposal suffers from the major drawback that the verification procedure requires significant classical computational resources on the part of the client. However, it is amenable to implementation on NISQ devices, and therefore may be one of the first applications of quantum computers.

In this work, we take a first step towards the experimental implementation of a protocol for generating certified random numbers on a quantum computer accessed through the Internet. We mostly follow the proposal of [146]. We execute a prototype of the protocol on a Sycamore superconducting quantum processor [8]. In our implementation, we focused on exercising the software infrastructure needed to run this protocol through the Internet; in particular, we used automatic calibration of the quantum processor, a procedure with temporary limitations that prevented us from achieving experimental parameters that would allow true certification of randomness. In particular, we used only 23 qubits.

The protocol that we use works as follows. The client generates challenge circuits which it sends to the server, who is operating a quantum computer. The server responds to each challenge by sending a requested number of bitstrings within a given short time limit T. When the server is honest, it produces the bitstrings by sampling from the circuit using the quantum computer, so the bitstrings contain entropy due to the inherent randomness of quantum measurements. The client performs statistical tests on the returned bitstrings in

order to verify that the server must have produced them honestly. Finally, the client passes the raw bitstrings through a randomness extractor to obtain random bits of higher quality, in the sense of being closer to the uniform distribution.

One of the statistical tests performed by the client involves simulating the circuits to compute the probabilities of the bitstrings in the circuit output states; this is the reason that the verification is expensive, since simulating quantum circuits takes time exponential in the number of qubits. Indeed, the circuits must be difficult enough to simulate that an adversary cannot plausibly cheat the protocol by performing classical simulations instead of quantum sampling. The expense of verification is the major point in which the protocol can potentially be improved upon.

6.2 Preliminaries

6.2.1 Random quantum circuit sampling

A quantum circuit C on n qubits determines an ideal probability distribution $p_C(s)$ over the 2^n possible bitstrings of length n, with the probability of a bitstring s being equal to the magnitude squared of the amplitude of the bitstring in the output state of the circuit. We call p_C the output distribution of the circuit. Sampling from the output distribution of a random circuit drawn from appropriate ensembles is believed to be a difficult task for classical computers [146, 147, 148]. For deep enough quantum circuits, the running time of a classical algorithm for this task is exponential in the number of qubits. By contrast, the task is easy to perform on an ideal quantum computer. Currently existing quantum computers lack error-correction, so they can sample from the output distribution only with an imperfect circuit fidelity $F < 1$. Sampling with imperfect fidelity is still difficult for classical computers.

In this work, we use a family of circuits similar to the one used in the demonstration of quantum supremacy [8]. Each circuit consists of a number of "cycles," where each cycle consists of two layers of gates: a layer of single-qubit gates followed by a layer of two-qubit gates. The single-qubit gates are chosen randomly from the set $\{\sqrt{X}, \sqrt{Y}, \sqrt{W}\}$, where X and Y are the Pauli matrices and $W = (X + Y)/\sqrt{2}$, subject to the constraint that no qubit is acted upon by the same single-qubit gate in consecutive cycles. In the layer of two-qubit gates, the pattern of qubit interactions is specially chosen to be difficult to simulate [8]. The two-qubit gate we use is native to the Sycamore hardware and has the

matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & -i & 0 & 0 \\ 0 & 0 & 0 & e^{-i\frac{\pi}{6}} \end{pmatrix}. \tag{6.1}$$

At the end of the circuit, an additional layer of single-qubit gates is appended, subject to the same constraint regarding consecutive cycles described above.

6.2.2 Cross-entropy benchmarking (XEB)

For a random quantum circuit drawn from an appropriate ensemble, such as the one described above, the distribution of the ideal output probabilities is well-approximated by an exponential, or Porter-Thomas, distribution with probability density function given by

$$f(x) = e^{-x}, \tag{6.2}$$

where $x = Dp$ is the ideal bitstring probability $p = p_C(s)$ scaled by the Hilbert space dimension $D = 2^n$. When a bitstring s is sampled with circuit fidelity F, the probability density of $x = Dp$ is well approximated by

$$f(x|F) = [Fx + (1 - F)]e^{-x} \ . \tag{6.3}$$

Furthermore, if we assume these approximations are exact, then we have [8]

$$F = D \cdot \langle p_C(s) \rangle - 1, \tag{6.4}$$

where $\langle p_C(s) \rangle$ denotes the expected value of the ideal probabilities $p_C(s)$ for bitstring s sampled with fidelity F. Therefore, the experimental value of F can be estimated from a set of sampled bitstrings $s_1, \ldots, s_k$ by computing the linear cross-entropy fidelity

$$\mathcal{F}_{\text{XEB}} = D \cdot \frac{1}{k} \sum_{j=1}^{k} p_C(s_j) - 1 \ . \tag{6.5}$$

If multiple circuits $C_1, \ldots, C_R$ are sampled, with $s_{i,j}$ corresponding to the j-th bitstring sampled from the i-th circuit, then we can calculate the linear cross-entropy fidelity using

all the bitstrings as

$$\mathcal{F}_{\text{XEB}} = D \cdot \frac{1}{Rk} \sum_{i=1}^{R} \sum_{j=1}^{k} p_{C_i}(s_{i,j}) - 1 \, . \tag{6.6}$$

6.2.3 Randomness extractors

Randomness extractors are functions that convert bits from a weak source of randomness into near-uniform random bits [149]. For general sources of randomness, this is only possible if the function is also given a small uniformly random input seed as a catalyst. The amount of randomness in a random variable X is measured by its min-entropy, which is given by $-\max_x \log(\Pr[X = x])$.

Formally, a function $\text{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ε)-extractor if for every random variable X on $\{0,1\}^n$ with min-entropy at least k, $\text{Ext}(X, Y)$ is ε-close to uniform when the seed Y is uniformly distributed on $\{0,1\}^d$. In our protocol we will apply a randomness extractor to the output of a quantum computer, which contains intrinsic randomness but is not uniformly distributed.

6.3 The computational hardness assumption

The validity of our protocol for generating certified randomness depends on a computational hardness assumption regarding the following problem:

Problem 1. *Given a quantum circuit C, positive integer k, and fidelity threshold F, produce unique bitstrings $s_1, \ldots, s_k$ such that $\mathcal{F}_{XEB} \geq F$, where $\mathcal{F}_{XEB}$ is defined in (6.5).*

This problem is easy to perform with access to a quantum computer that can execute the given circuit with fidelity F: Simply execute and sample the circuit on the quantum computer to obtain the requested number of bitstrings; at relevant experimental parameters where $n > 50$ and k is at most a few million, the sampled bitstrings will be unique with very high probability. By contrast, this problem is difficult for classical computers for appropriate ensembles of circuits. The best known classical algorithms for this problem involve simulating the circuit, which has running time that scales exponentially with the number of qubits [150, 151].

The assumption we make is the following:

Assumption 1. *For a "large enough" circuit drawn from certain ensembles, including the one given in Section 6.2.1, solving Problem 1 with high probability in a "sufficiently*

short" amount of time is only possible by executing and sampling the circuit on a quantum computer with fidelity at least F.

Crucially, our assumption is that the problem must be solved through sampling, an inherently random process. The determination of what constitutes a "large enough" circuit (in terms of qubit number or depth) or a "sufficiently short" amount of time are to be made at the time of executing the protocol, taking into account practical considerations such as what the best available classical simulation algorithms can achieve.

6.4 The protocol

The protocol for generating certified randomness involves interaction between a client, who desires to generate random bits, and a server, who operates a quantum computer. The protocol consists of three stages: sample generation, verification, and randomness extraction. Only the sample generation stage requires interaction between the client and the server. The verification and randomness extraction stages can be performed by the client alone.

The protocol is parameterized by the following numbers:

- Operational parameters:
  - R: The number of rounds of interaction.
  - k: The number of unique bitstrings to sample per circuit.
  - T: The time limit for the server to respond in each round.
- Verification parameters:
  - ℓ: The number of rounds to verify.
  - F: The minimum value of the linear cross entropy fidelity required to pass the verification test.

The sample generation stage, described as an algorithm executed by the client, is as follows:

Algorithm 1 (Sample generation)**.**

1. *Begin with a seed of uniformly random bits y_1.*
2. *Using y_1, seed a pseudorandom number generator g.*

3. *For $i = 1$ to R:*

   (a) *Use g to generate a circuit C_i and send it to the server.*

   (b) *Wait to receive unique samples $\langle s_{i,1}, \ldots, s_{i,k} \rangle := S_i$.*

   (c) *If the server takes longer than time T to respond,* abort *the protocol.*

The verification stage is as follows:

Algorithm 2 (Verification)**.**

1. *Begin with a seed of uniformly random bits y_2.*

2. *Using y_2, select a subset $L \subseteq R$ of size ℓ uniformly at random.*

3. *For each $i \in L$, compute the ideal bitstring probabilities $p_{C_i}(s_{i,j})$ for the bitstrings $s_{i,1}, \ldots, s_{i,k}$.*

4. *Using all the computed bitstring probabilities, calculate the linear cross-entropy fidelity $\mathcal{F}_{\text{XEB}}$. If $\mathcal{F}_{\text{XEB}} < F$,* reject. *Otherwise, perform other statistical tests on the bitstrings (see below). If the tests pass,* accept. *Otherwise,* reject.

In Step 4 of the verification, the client may perform additional statistical tests on the bitstrings to counter adversarial strategies that can lower the cost of producing bitstrings through classical simulation. One such strategy is to use a tensor network simulator to partially contract the tensor network of the circuit by fixing some of the output bits, allowing for the sampling of bitstrings that share those output bits at a relatively lower cost. To counter this strategy, the client should check that the bitstrings do not form clusters of small Hamming distance.

If the verification passes, then the client proceeds to the randomness extraction stage, which is simple to describe, and uses a randomness extractor Ext:

Algorithm 3 (Randomness extraction)**.**

1. *Begin with a seed of uniformly random bits y_3.*

2. *Concatenate all sample sets S_i into a single bitstring S.*

3. *Output* $\mathrm{Ext}(S, y_3)$.

The output of the randomness extraction stage is the output of the protocol. Assuming that the verification stage ended with acceptance, the output is certified to consist of near-uniform random bits.

6.5 Entropy estimation

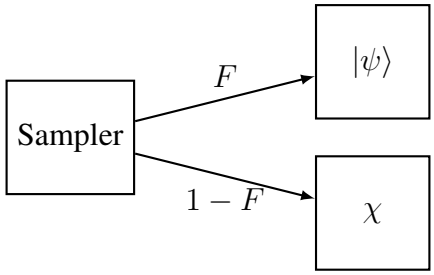
In this section, we estimate the amount of min-entropy contained in the output of the protocol.

6.5.1 The case of an honest server

The experimental output of a noisy quantum random circuit can be described as

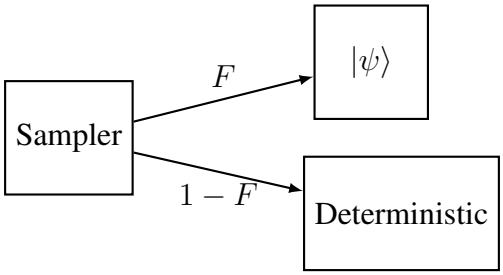
$$F|\psi\rangle\langle\psi| + (1-F)\chi \ , \tag{6.7}$$

where F is the experimental fidelity (probability of no error), $|\psi\rangle$ is the ideal output of the quantum circuit, and χ is the result of errors. Measurement of this state can be interpreted as measuring the ideal quantum state $|\psi\rangle$ with probability F, and measuring the operator χ with probability $1-F$. This is depicted in the following diagram:



(6.8)

For the purposes of quantum randomness generation, we take an adversarial approach with respect to the noise operator χ and consider it to be deterministic. The reason is that we are only interested in the quantum entropy generated experimentally, and not in using potential "noise" or "errors" in the experiment as a source of entropy. Arguably, if we were willing to accept an entropy source based on "noise", there are simpler setups than the one proposed in this paper, and they would depend on specific models for the noise. Furthermore, the potential entropy coming from the noise cannot be certified. Therefore, we model the sampling as depicted in the next diagram:



(6.9)

In this model, the bitstring with the highest probability is the deterministic noise with probability $1 - F$. Therefore, for a sample of k bitstrings, the min-entropy is

$$\text{min-entropy} = -\log\big((1-F)^k\big) \approx kF \tag{6.10}$$

It is possible to obtain a tighter bound by ignoring the very unlikely event that all outputs of the experiment correspond to the "noise" term. In the simplified model of (6.9), we can chose a constant c_1 such that with a given high probability there are effectively at least $kF - c_1\sqrt{kF(1-F)}$ samples from the ideal quantum distribution, where k is the number of bitstrings sampled. For instance, $c_1 = 5$ results in a probability of $1 - 1.5 \times 10^{-12}$.

Let us now assume that the distribution of the ideal bitstring probabilities $p(s) = |\langle s|\psi\rangle|^2$ follows the Porter-Thomas distribution with probability density function

$$f(x) = e^{-x}, \tag{6.11}$$

where $x = Dp(s)$ is the ideal bitstring probability scaled by the Hilbert space dimension $D = 2^n$. In this case, the minus log probability of a sample has normal distribution with average $q(\log(D) - 1 + \gamma)$, where $q = kF - c_1\sqrt{kF(1-F)}$ and γ is the Euler constant. The variance is $q\pi^2/6$. We can chose a constant c_2 such that with a given high probability the min-entropy is $q(\log(D) - 1 + \gamma) - c_2\sqrt{q\pi^2/6}$. Putting it all together, we obtain the following lower bound for the min-entropy:

$$q(\log(D) - 1 + \gamma) - c_2\sqrt{\frac{q\pi^2}{6}}. \tag{6.12}$$

6.5.2 Adjustments to the min-entropy bound

Depending on what adversarial assumptions are made on the server, adjustments should be made to the bound (6.12). One such adjustment arises from considering that the server can reorder bitstrings in the sample before returning it to the client. The min-entropy above is given by the maximum probability of a sample of $q = kF - c_1\sqrt{kF(1-F)}$ bitstrings. If $q \sim kF \ll \sqrt{D}$, all these bitstrings are most likely distinct. If the server can reorder bitstrings in the sample, then the maximum probability of a sequence of bitstrings is now $q!$ times higher, so the min-entropy must be corrected to

$$q(\log(D) - 1 + \gamma) - c_2\sqrt{\frac{q\pi^2}{6}} - \log(q!)\ . \tag{6.13}$$

6.6 Security and verification time

Our proposal for certifiable randomness generation presents two competing requirements:

1. The random circuit sampling task must be difficult enough for classical simulations so that no practical adversary can simulate it in the allocated sampling time for the random number generator server.

2. The random circuit sampling task must be tractable enough for classical simulations so that the verification can be carried out in practice.

Algorithms for the sampling task are continually improving, and it is unlikely that a theoretically optimal algorithm will be mathematically proven. Therefore, we must rely on numerical benchmarking to argue for some degree of practical security for this protocol. Furthermore, we must rely on practical assumptions on the amount of computing power available to a potential adversary.

On the positive side, there are several techniques that allow us to tailor the computational cost of available algorithms and to trade the computational cost between different algorithms. For a given number of qubits we can choose the circuit depth and number of elided gates to adjust the required verification time [8]. It's probably preferable to have a large number of qubits to rule Schrodinger type simulations on a supercomputer with a fast interconnect; 53 qubits is enough. It's also preferable to use the minimum depth required to obtain a Porter-Thomas distribution, in order to maximize the fidelity.

We can start by fixing the maximum sampling time T imposed on the server. This depends on the sample size, sampling rate, and communication overhead. The sample size is proportional to $1/F^2$ for fidelity F. Let's assume that we can achieve a T of 1 minute (this would realistically be several minutes at the moment, but it will improve). We should choose a circuit ensemble difficult enough for the adversary to take a time longer than 1 minute using a realistic number of cores c_a. A reasonable choice is $c_a = 10^5$ CPU cores. The corresponding verification time using c_v cores would be $p(c_a/c_v)(1/F)$, where p is some padding factor accounting for possible unknown speedups of the adversary over the verifier. A reasonable choice would be $c_v = 10^4$ CPU cores and $p = 10$. We can aim for an optimistic fidelity of $F = 0.01$. In this case the verification time would be 1 week using 10^4 CPU cores in Google Cloud. The cost of this simulation would be about $16,000 using preemptible n1-standard VMs.

In general, the user can select the level of security of the certified random numbers according to the price he is willing to pay for the certification simulation.

6.7 Experimental implementation

In our implementation of the protocol, we focused on the exercise of the software infrastructure needed to run this protocol as a service accessed through the Internet. In particular, we used fully automatic calibration of our quantum processor, a procedure that suffers from some temporary limitations which prevent us from achieving experimental parameters that would provide true certification of randomness. In particular, we were limited to accessing 23 qubits. Our experiment was performed on a Sycamore processor.

We executed the protocol with $R = 50$ rounds. We used circuits consisting of 14 cycles. In each round we sampled $k = 10^6$ bitstrings. Since this is a prototype experiment on a relatively small number of qubits, we ignored the constraint that the sampled bitstrings had to be unique. The longest response time of the server was 404 seconds, but the average response time was 284 seconds, giving an effective sampling rate of about 3.5 kHz. For the linear cross entropy fidelity $\mathcal{F}_{XEB}$ we obtained a value of 0.079. To lower bound the min-entropy, we used Eq. (6.12) with $c_1 = c_2 = 5.0$, separately for each round, and added the results, giving an estimate of at least 8.7×10^7 bits of min-entropy being generated.

We implemented a randomness extractor following the construction of Raz, Reingold, and Vadhan [152], which is in turn based on Trevisan's construction [153]. For a fixed ε, the distance of the output from the uniform distribution, this extractor has an input seed size of $O(\log^2 n)$.

We used the GNU/Linux special device file `/dev/urandom` to produce the random seeds needed for the protocol. The PRNG used to generate the circuits was seeded with 624 bits; the PRNG was seeded once and then used to generate all the circuits. The randomness extractor with $\varepsilon = 0.01$ used a seed of 65,536 bits; due to the slowness of the implementation, we only applied it to the output of one of the rounds of the protocol.

Figure 6.1 shows the values of $\mathcal{F}_{\text{XEB}}$ calculated individually for the 500 protocol rounds. The highest value achieved was 0.088. Figure 6.2 compares the distribution of bitstring probabilities (the values $p_C(s)$) for the bitstrings obtained in this experimental round with bitstrings numerically sampled from the uniform distribution and from the ideal distribution p_C. The figure shows that the distribution closely matches the theoretical distribution (6.3).

6.8 Conclusion

In this work, we have implemented a prototype of a protocol for generating certified random numbers on a near-term quantum computer. Our protocol depends on a computational hardness assumption and its verification step is very expensive. In our protocol, the high

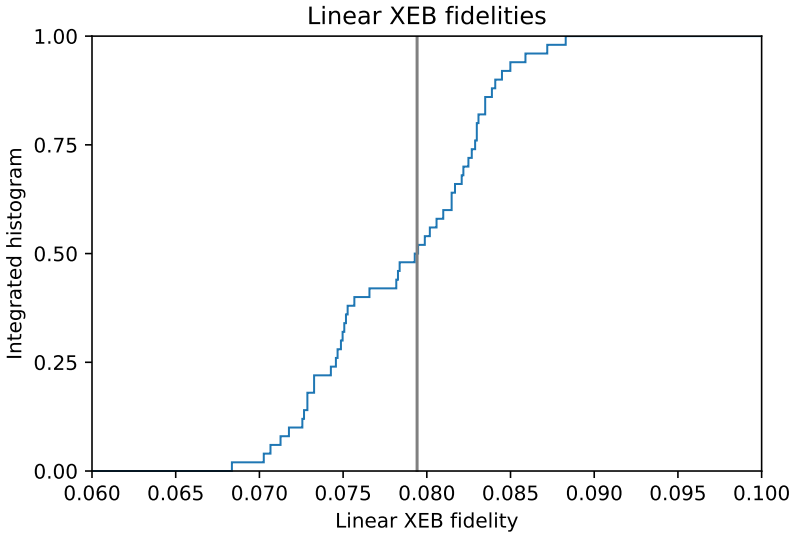


Figure 6.1: Integrated histogram of linear XEB fidelities achieved in the 50 protocol rounds executed. The circuits executed had 23 qubits and consisted of 14 cycles. The vertical line indicates the median value.

cost of the verification step is unavoidable because there is a direct relationship between the cost of the verification and the cost needed for an adversary to cheat the protocol. The most important open problem we leave here is to devise a protocol that is suitable for near-term quantum computers which does not require an expensive verification step.

In our implementation, we focused on exercising the software infrastructure needed to run this protocol through the Internet; in particular, we used automatic calibration of the quantum processor, a procedure with temporary limitations that prevented us from achieving experimental parameters that would allow true certification of randomness. Nevertheless, our results are a proof of concept that shows that certified random number generation may be the first application of NISQ computers.

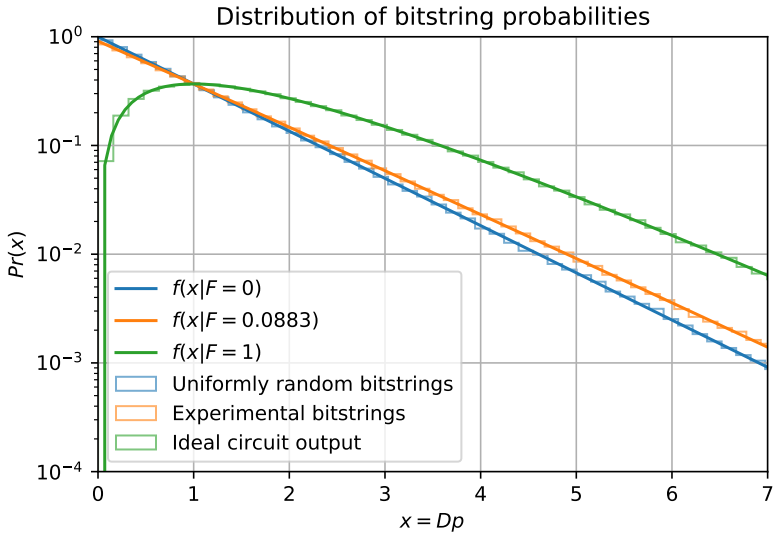


Figure 6.2: Histogram of scaled ideal probabilities for the protocol round with the highest fidelity. The ideal probability p of a bitstring is calculated from the final state amplitudes of the circuit, then scaled by the Hilbert space dimension $D = 2^n$, where n is the number of qubits (here, $n = 23$). The orange histogram represents the bitstrings obtained experimentally. The blue and green histograms are for bitstrings sampled uniformly at random, and from the ideal circuit output assuming a fidelity of 1, respectively. In each case, 10^6 bitstrings were sampled. The solid lines are theoretical probability distributions for the given fidelities.

APPENDIX A

Initial State for the Hubbard Model

In this appendix, we describe the initial state used for the numerics in Chapter 2 involving the Hubbard model.

The 2×2 Hubbard model has sites labeled as in Figure A.1.

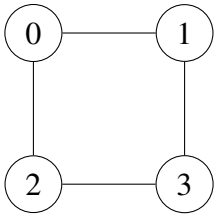


Figure A.1: Labeling of sites for 2×2 model.

For a single spin, the single-particle energies of the hopping term are {-2, 0, 0, 2}, with corresponding creation operators

$$
\begin{aligned}
b_0^\dagger &= \frac{1}{2}(a_0^\dagger + a_1^\dagger + a_2^\dagger + a_3^\dagger) \\
b_1^\dagger &= \frac{1}{\sqrt{2}}(a_0^\dagger - a_3^\dagger) \\
b_2^\dagger &= \frac{1}{\sqrt{2}}(a_1^\dagger - a_2^\dagger) \\
b_3^\dagger &= \frac{1}{2}(a_0^\dagger - a_1^\dagger - a_2^\dagger + a_3^\dagger)
\end{aligned}
$$

The ground eigenspace is degenerate, and a ground state has the form

$$
\left(\sum_{i,j=1}^{2} \alpha_{ij} b_{i,\uparrow} b_{j,\downarrow} \right) b_{0,\uparrow}^\dagger b_{0,\downarrow}^\dagger |\text{vac}\rangle
$$

Table A.1 lists the choices for the coefficients α_{ij} that give states with the correct total spin (singlet).

Choice	$\alpha_{1,1}$	$\alpha_{1,2}$	$\alpha_{2,1}$	$\alpha_{2,2}$
1	1	0	0	0
2	0	0	0	1
3	0	$1/\sqrt{2}$	$1/\sqrt{2}$	0
4	$1/\sqrt{2}$	0	0	$1/\sqrt{2}$
5	$1/\sqrt{2}$	0	0	$-1/\sqrt{2}$

Table A.1: Coefficient choices for the 2×2 Hubbard model ground state that give the correct total spin.

Of these, only choices 3 and 4 led to optimized energies that matched the true ground energy. We used choice 3 to construct our initial state.

APPENDIX B

Pseudocode for MGD and MPG

In this appendix, we give pseudocode for the algorithms MGD and MPG described in Chapter 2. The pseudocode for MGD is given in Algorithm B.1, and the pseudocode for MPG is given in Algorithm B.2.

Algorithm B.1 Model Gradient Descent

Input: Initial point x_0, learning rate γ, sample radius δ, sample number k, rate decay exponent α, stability constant A, sample radius decay exponent ξ, tolerance ε, maximum evaluations n

Initialize a list L
Let $x \leftarrow x_0$
Let $m \leftarrow 0$
while (#function evaluations so far) + k does not exceed n **do**
    Add the tuple $(x, f(x))$ to the list L
    Let $\delta' \leftarrow \delta/(m+1)^{\xi}$
    Sample k points uniformly at random from the δ'-neighborhood of x; Call the resulting set S
    for each x' in S **do**
        Add $(x', f(x'))$ to L
    end for
    Initialize a list L'
    for each tuple (x', y') in L **do**
        if $|x' - x| < \delta'$ **then**
            Add (x', y') to L'
        end if
    end for
    Fit a quadratic model to the points in L' using least squares linear regression with polynomial features
    Let g be the gradient of the quadratic model evaluated at x
    Let $\gamma' = \gamma/(m+1+A)^{\alpha}$
    if $\gamma' \cdot |g| < \varepsilon$ **then**
        return x
    end if

Algorithm B.1 Model Gradient Descent (continued)

Let $x \leftarrow x - \gamma' \cdot g$
Let $m \leftarrow m + 1$
end while
return x

Algorithm B.2 Model Policy Gradient

Input: learning rate γ, sample radius ratio δ_r, sample number k, model sample number M, learning rate decay exponent α, mean initialization $\boldsymbol{\mu}_0$, standard deviation initialization σ_0, decay steps t_{decay}, warm up steps t_{warm}, maximum evaluations n

Initialize a list L
Initialize the policy: $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}_0, \boldsymbol{\sigma} \leftarrow [\sigma_0, \cdots, \sigma_0]^T$.
Let $m \leftarrow 0$
while (#function evaluations so far) + k does not exceed n **do**
    Greedy estimation by the current policy: $x \leftarrow \arg\max_{\tilde{x}} \pi_{\boldsymbol{\varphi}}(\tilde{x})$.
    Add the tuple $(x, f(x))$ to the list L
    Sample k points according to the current policy $\pi_{\boldsymbol{\varphi}}$; Call the resulting set S
    for each x' in S **do**
        Add $(x', f(x'))$ to L
    end for
    Estimate the maximal radius within the set S, i.e. $r_{\max} \leftarrow \max_{x' \in S} |x' - x|$.
    if $m < t_{\text{warm}}$ **then** ▷ Compute the policy gradient directly
        Compute the baseline $\bar{f} \leftarrow \frac{1}{k} \sum_{x' \in S} f(x')$.
        Compute the policy gradient using the sampled data points

$$\nabla_{\boldsymbol{\varphi}} J(\boldsymbol{\varphi}) \leftarrow \frac{1}{k} \sum_{x' \in S} \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(x') \cdot (f(x') - \bar{f}).$$

    else ▷ Fit the model to compute the policy gradient
        Initialize a list L'
        for each tuple (x', y') in L **do**
            if $|x' - x| < \delta_r r_{\max}$ **then**
                Add (x', y') to L'
            end if
        end for
        Fit a quadratic model $F(\cdot)$ to the points in L' using least squares linear regression with polynomial features
        Sample M points according to the current policy $\pi_{\boldsymbol{\varphi}}$; Call the resulting set S'
        for each x' in S' **do**
            Evaluate with the model $F(x')$.
        end for
        Compute the baseline $\bar{F} \leftarrow \frac{1}{M} \sum_{x' \in S'} F(x')$.

Algorithm B.2 Model Policy Gradient (continued)

Compute the policy gradient

$$\nabla_{\boldsymbol{\varphi}} J(\boldsymbol{\varphi}) \leftarrow \frac{1}{M} \sum_{x' \in S'} \nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(x') \cdot (F(x') - \bar{F}).$$

end if
Decay the learning rate $\gamma' \leftarrow \gamma \cdot \alpha^{m/t_{\text{decay}}}$
Update the weights $\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} - \gamma' \cdot \nabla_{\boldsymbol{\varphi}} J(\boldsymbol{\varphi})$
Let $m \leftarrow m + 1$
end while
Greedy estimation by the current policy: $x \leftarrow \arg\max_{\tilde{x}} \pi_{\boldsymbol{\varphi}}(\tilde{x})$.
return x

APPENDIX C

Hyperparameter Selection for Optimization Numerics

Each algorithm we studied in Chapter 2 had hyperparameters and the choice of these hyperparameters had a great impact on performance. We tuned hyperparameters by performing a grid search. For each combination of hyperparameters considered in the search, we performed an optimization run using the wall clock time model that includes network latency and circuit batching. The optimal hyperparameters were those that minimized time to convergence with a precision target of 10^{-3}. Note that this choice does have an effect on the performance of the algorithms; choosing a more lenient precision target would give different results. To demonstrate this effect, we optimized hyperparameters of SPSA for the Hubbard model for a precision target of 10^{-2} instead of 10^{-3}. The results are shown in Figure C.1. As expected, the algorithm optimized for 10^{-2} performs better at larger precision cutoffs and worse at smaller ones.

Below, we describe the hyperparameters of these algorithms and the values that we searched through. For each algorithm, we considered the number of measurement shots per energy evaluation to be a hyperparameter, and for each algorithm we considered different sets of possible values between the QAOA and Hubbard model problems. In the tables below, there is one line for the values considered for the QAOA problems, and one line for the values considered for the Hubbard model. We include tables of the hyperparameters chosen by our grid search.

C.1 Nelder-Mead

The Nelder-Mead simplex method has a single additional hyperparameter which we call δ. This hyperparameter affects the size of the initial simplex. Given an initial guess $\boldsymbol{\theta}_0$, the algorithm constructs its initial simplex $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m)$, where m is the dimension of $\boldsymbol{\theta}_0$,

by defining $\boldsymbol{\theta}_i$ to be equal to $\boldsymbol{\theta}_0$ but with its i-th coordinate multiplied by $1+\delta$. In Table C.1 we show the hyperparameter values that we searched through. In Table C.2 we show the hyperparameters that were chosen by the search.

Hyperparameter	Possible values
number of shots (QAOA)	5,000, 25,000, 125,000, 625,000
number of shots (Hubbard)	10,000, 100,000, 1,000,000, 10,000,000
δ (determines initial simplex size)	0.001, 0.002, 0.004, 0.008, 0.016, 0.032, 0.064, 0.128, 0.256, 0.512

Table C.1: Hyperparameter selection for Nelder-Mead

Hyperparameter	3-reg (p=1)	3-reg (p=5)	SK (p=1)	SK (p=5)	Hubbard (p=5)
number of shots	25,000	25,000	25,000	125,000	10,000,000
δ	0.128	0.064	0.064	0.256	0.256

Table C.2: Optimized hyperparameters for Nelder-Mead

C.2 Bounded Optimization By Quadratic Approximation (BOBYQA)

The BOBYQA algorithm maintains a set of points $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k)$ through which it fits an interpolating quadratic model. In each iteration, it uses the model to predict a good point to go next, and incorporates that point into the model by replacing another point. The model is only assumed to be accurate within a "trust region radius" ρ of the most recently added point.

The value of k is a hyperparameter that can take values from $\{m+1, \ldots, (m+1)(m+2)/2\}$. In an m-dimensional optimization problem, it takes $(m+1)(m+2)/2$ points to fully determine a quadratic function. Thus, if k is smaller than this value, there is some freedom in choosing the particular quadratic function. BOBYQA takes up this freedom by minimizing the Frobenius norm of the difference between the Hessians of successive quadratic models. Instead of using k directly as a hyperparameter, we defined a transformed hyperparameter α taking values from $[0,1]$ and derived k from it using the formula $k = \lfloor (m+1) + \alpha[(m+1)(m+2)/2 - (m+1)] \rfloor$.

BOBYQA also has a hyperparameter we call ρ_0 which is the trust region radius at the beginning of the algorithm. In Table C.3 we show the hyperparameter values that we

searched through. In Table C.4 we show the hyperparameters that were chosen by the search.

Hyperparameter	Possible values
number of shots (QAOA)	5,000, 25,000, 125,000, 625,000
number of shots (Hubbard)	10,000, 100,000, 1,000,000, 10,000,000
α (determines number of points to interpolate)	0.0, 0.2, 0.4, 0.6, 1.0
ρ_0 (initial trust region radius)	0.01, 0.02, 0.04, 0.08, 0.16

Table C.3: Hyperparameter selection for BOBYQA

Hyperparameter	3-reg (p=1)	3-reg (p=5)	SK (p=1)	SK (p=5)	Hubbard (p=5)
number of shots	25,000	25,000	125,000	25,000	10,000,000
α	0.6	0.2	1.0	0.2	0.2
ρ_0	0.04	0.16	0.08	0.16	0.04

Table C.4: Optimized hyperparameters for BOBYQA

C.3 Stochastic Gradient Descent (SGD)

SGD has two additional parameters, the learning rate γ and the decay rate β. These determine the update rule that uses the current gradient $\mathbf{g}_j$ to update the current point $\boldsymbol{\theta}_j$ to the next point $\boldsymbol{\theta}_{j+1}$ as follows:

$$\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j - \gamma e^{-\beta j} \mathbf{g}_j.$$

In Table C.5 we show the hyperparameter values that we searched through. In Table C.6 we show the hyperparameters that were chosen by the search.

Hyperparameter	Possible values
number of shots	1000, 5000, 10000, 20000, 40000
number of shots (Hubbard)	10,000, 100,000, 1,000,000, 10,000,000
γ (learning rate)	0.001, 0.002, 0.004, 0.008, 0.016, 0.032, 0.064, 0.128, 0.256
β (decay rate)	0.01, 0.02, 0.04, 0.08, 0.16, 0.32

Table C.5: Hyperparameter selection for SGD

Hyperparameter	3-reg (p=1)	3-reg (p=5)	SK (p=1)	SK (p=5)	Hubbard (p=5)
number of shots	1,000	1,000	1,000	1,000	10,000
γ	0.016	0.008	0.008	0.004	0.004
β	0.32	0.02	0.16	0.08	0.32

Table C.6: Optimized hyperparameters for SGD

C.4 Simultaneous Perturbation Stochastic Approximation (SPSA)

SPSA estimates the gradient $\mathbf{g}_j$ at point $\boldsymbol{\theta}_j$ using the expression

$$\mathbf{g}_{j,k} = \frac{f(\boldsymbol{\theta}_j + c_j \boldsymbol{\Delta}_j) - f(\boldsymbol{\theta} - c_j \boldsymbol{\Delta}_j)}{2c_j} \cdot \boldsymbol{\Delta}_{j,k}^{-1}$$

where $\boldsymbol{\Delta}_j$ is chosen in each iteration to be a vector whose entries are chosen to be plus or minus 1 with equal probability and $c_j = c/j^\gamma$ where c and γ are hyperparameters called the perturbation size and perturbation decay exponent, respectively. The new point $\boldsymbol{\theta}_{j+1}$ is calculated according to the update rule

$$\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j - a_j \mathbf{g}_j$$

where $a_j = a/(j + A)^\alpha$ where a, α, and A are hyperparameters called the rate, rate decay exponent, and stability constant, respectively.

In Table C.7 we show the hyperparameter values that we searched through. Instead of trying every possible combination, we randomly picked 1000 combinations. In Table C.8 we show the hyperparameters that were chosen by the search.

Hyperparameter	Possible values
number of shots (QAOA)	5,000, 25,000, 125,000, 625,000
number of shots (Hubbard)	10,000, 100,000, 1,000,000, 10,000,000
a (rate)	0.005, 0.01, 0.02, 0.04, 0.08
c (perturbation size)	0.01, 0.02, 0.04, 0.08, 0.16
α (rate decay exponent)	0.1, 0.2, 0.4, 0.8
A (stability constant)	0, 50, 100, 200, 400
γ (perturbation decay exponent)	0.01, 0.02, 0.04, 0.08, 0.16

Table C.7: Hyperparameter selection for SPSA

Hyperparameter	3-reg (p=1)	3-reg (p=5)	SK (p=1)	SK (p=5)	Hubbard (p=5)
number of shots	1,000	25,000	25,000	25,000	1,000,000
a	0.08	0.04	0.005	0.01	0.01
c	0.16	0.01	0.02	0.02	0.02
α	0.4	0.8	0.2	0.8	0.8
A	200	50	50	100	100
γ	0.04	0.01	0.04	0.02	0.16

Table C.8: Optimized hyperparameters for SPSA

C.5 Model Gradient Descent (MGD)

MGD and its hyperparameters are described in Algorithm B.1. In our study we re-parameterized the hyperparameter k, the sample number, in a similar way to how we re-parameterized the number of interpolation points in BOBYQA. Instead of using k directly as a hyperparameter, we defined a transformed hyperparameter η being a positive real number and derived k from it using the formula $k = \eta \cdot (m+1)(m+2)/2$, where m is the dimension of the optimization problem.

In Table C.9 we show the hyperparameter values that we searched through. Instead of trying every possible combination, we randomly picked 1000 combinations. In Table C.10 we show the hyperparameters that were chosen by the search.

Hyperparameter	Possible values
number of shots (QAOA)	5,000, 20,000, 80,000
number of shots (Hubbard)	10,000, 100,000, 1,000,000, 10,000,000
γ (rate)	0.01, 0.02, 0.04, 0.08, 0.16
δ (sample radius)	0.01, 0.02, 0.04, 0.08, 0.16
η (determines sample number)	0.3, 0.6, 0.9, 1.2
α (rate decay exponent)	0.1, 0.2, 0.4, 0.8
A (stability constant)	0, 50, 100, 200, 400
ξ (sample radius decay exponent)	0.01, 0.02, 0.04, 0.08, 0.16

Table C.9: Hyperparameter selection for MGD

C.6 Model Policy Gradient (MPG)

MPG (Algorithm B.2) parameterizes a Gaussian sampling policy and optimizes in its parameter space. The learnable parameters introduced here are the mean and standard deviation of the policy, i.e. $\boldsymbol{\varphi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ have the same dimension as the point

Hyperparameter	3-reg (p=1)	3-reg (p=5)	SK (p=1)	SK (p=5)	Hubbard (p=5)
number of shots	1,000	5,000	1,000	5,000	100,000
γ	0.08	0.01	0.16	0.16	0.01
δ	0.08	0.08	0.04	0.04	0.08
η	0.9	0.3	1.2	0.3	0.6
α	0.4	0.4	0.8	0.8	0.4
A	100	0	100	400	100
ξ	0.08	0.08	0.02	0.01	0.04

Table C.10: Optimized hyperparameters for MGD

$\boldsymbol{\theta}$. Every iteration it samples a batch of data points to estimate the direction (Eqn. C.1) which maximizes the expected total reward (in our case, the reward is the negative ground state energy). The drawback of the vanilla policy gradient (VPG) algorithm [41] is that it requires a large batch size to control the variance of estimation. In order to enhance the sample efficiency, we integrate the idea of surrogate model-based optimization with the VPG algorithm. A quadratic model is trained by reusing the history data within some trust region of the current estimation $\boldsymbol{\theta}$. Once we have the model, we can query it to output estimations for any data point within the region. Note that the estimations of these data points have little cost compared with the samples in the beginning. Finally, the policy gradient is applied to improve the policy at the end of each iteration

$$\nabla_{\boldsymbol{\varphi}} J(\boldsymbol{\varphi}) = \mathop{\mathbb{E}}_{\boldsymbol{\theta}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma})} \left[\nabla_{\boldsymbol{\varphi}} \log \pi_{\boldsymbol{\varphi}}(\boldsymbol{\theta}) \cdot f(\boldsymbol{\theta})\right]. \tag{C.1}$$

The hyperparameters of the MPG algorithm are described as follows. The optimizer is chosen to be Adam [154], with β_1, β_2 being 0.9 and 0.999. The learning rate hyperparameter is γ with an exponential decay schedule of rate α for every step t_{decay}. The hyperparameter σ_0 specifies the initialization for the standard deviation of the Gaussian policy. The hyperparameter k is the sample batch size at each iteration. The sample radius ratio δ_r with respect to the maximal radius of the samples determines the trust region in which to fit the model. The hyperparameter t_{warm} is introduced because in the beginning, the number of data points collected is not adequate enough to fit a good model. Thus we adopt the vanilla policy gradient for the first several iterations before we accumulate enough data points. The hyperparameter M is the model sample number to estimate the policy gradient. It needs to be big enough so that the variance of the estimation is low. In our experiments, we used a constant $M = 65536$. Since here we use the quadratic model and Gaussian policy, one can also compute the policy gradient analytically, but implementing it this way would allow plugging in different models.

In Table C.11 we show the hyperparameter values that we searched through. Instead of trying every possible combination, we randomly picked 1000 combinations. In Table C.12 we show the hyperparameters that were chosen by the search.

Hyperparameter	Possible values
number of shots (QAOA)	1,000, 5,000, 20,000
number of shots (Hubbard)	10,000, 100,000, 1,000,000, 10,000,000
γ (learning rate)	0.001, 0.005, 0.008, 0.01, 0.02
α (learning rate decay exponent)	0.99, 0.96, 0.93, 0.90
σ_0 (standard deviation initialization)	$\exp(-4.0)$, $\exp(-5.0)$, $\exp(-6.0)$
k (sample number)	10, 20, 40
δ_r (sample radius ratio)	1.0, 2.0, 3.0
t_{warm} (warm up steps)	0, 5, 10

Table C.11: Hyperparameter selection for MPG

Hyper-parameter	3-reg (p=1)	3-reg (p=5)	SK (p=1)	SK (p=5)	Hubbard (p=5)
number of shots	5,000	20,000	20,000	5,000	1,000,000
γ	0.02	0.005	0.02	0.005	0.01
α	0.99	0.96	0.99	0.93	0.90
σ_0	$\exp(-4.0)$	$\exp(-4.0)$	$\exp(-4.0)$	$\exp(-4.0)$	$\exp(-5.0)$
k	10	10	10	20	20
δ_r	3	3	2	2	3
t_{warm}	10	5	5	0	10

Table C.12: Optimized hyperparameters for MPG

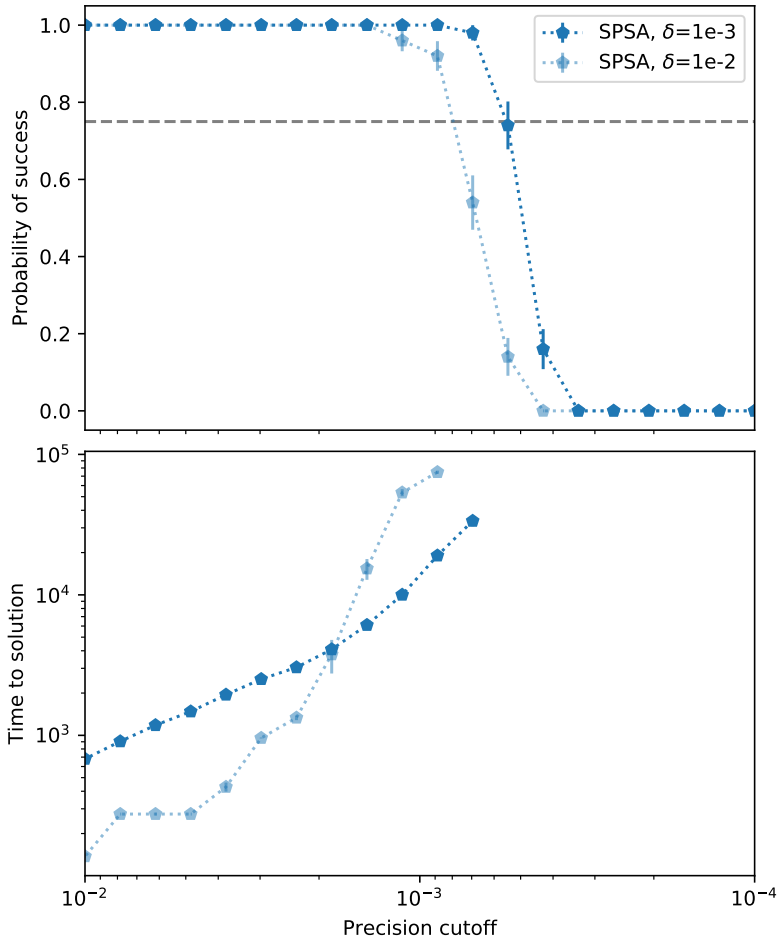


Figure C.1: Success probability and time to solution for varying levels of required precision at $p = 5$, for SPSA on the Hubbard model. Results are shown for two hyperparameter settings, optimized for two different precision cutoffs δ: 10^{-3} (dark colored) and 10^{-2} (light colored). Top: The probability of converging (out of 50 trials) to the optimal value of the ansatz at the given precision. Bottom: The average wall clock time the optimizer took to reach the given precision. Error bars represent 1 standard deviation. Time to solution is only reported if the probability of convergence was at least 75% (dotted horizontal gray line).

APPENDIX D

Additional Data from Optimization Numerics

In this appendix, we give additional data from the numerical experiments of Chapter 2.

In Figure D.1 we show a version of Figure 2.2 that also includes a plot for the 3-regular graph model. For the 3-regular graph model, BOBYQA only converged in 34 out of 50 runs, so we exclude its data (there other algorithms converged in at least 49 runs).

Figure D.2 shows a version of Figure 2.4 that also includes plots for the Sherrington-Kirkpatrick and Hubbard models. Figure D.3 plots the final energy error of the optimizers as a function of the amount of gate rotation error present, at $p = 5$. It shows that for the QAOA problems, MGD and SPSA clearly outperform the others when the final energy error is required to be less than about 1e-2. For the Hubbard model, the optimizers do not differentiate as clearly.

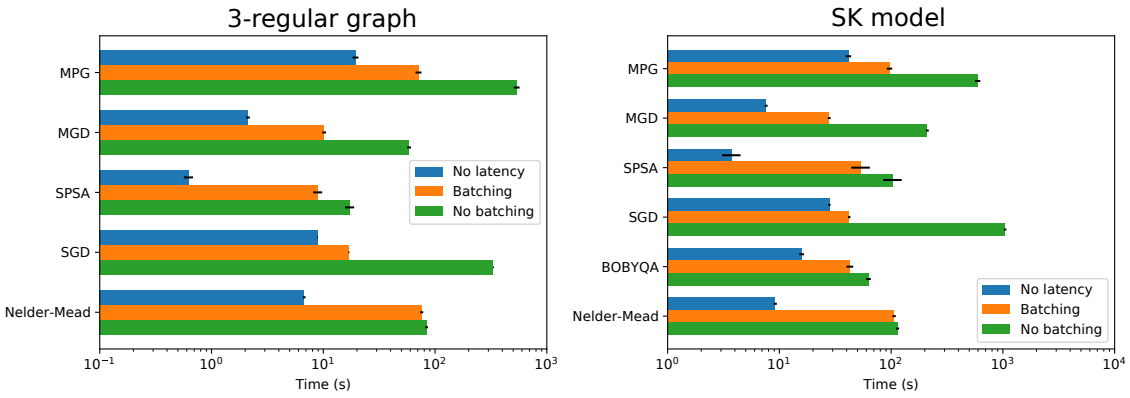


Figure D.1: Version of Figure 2.2 that also includes a plot for the 3-regular graph model.

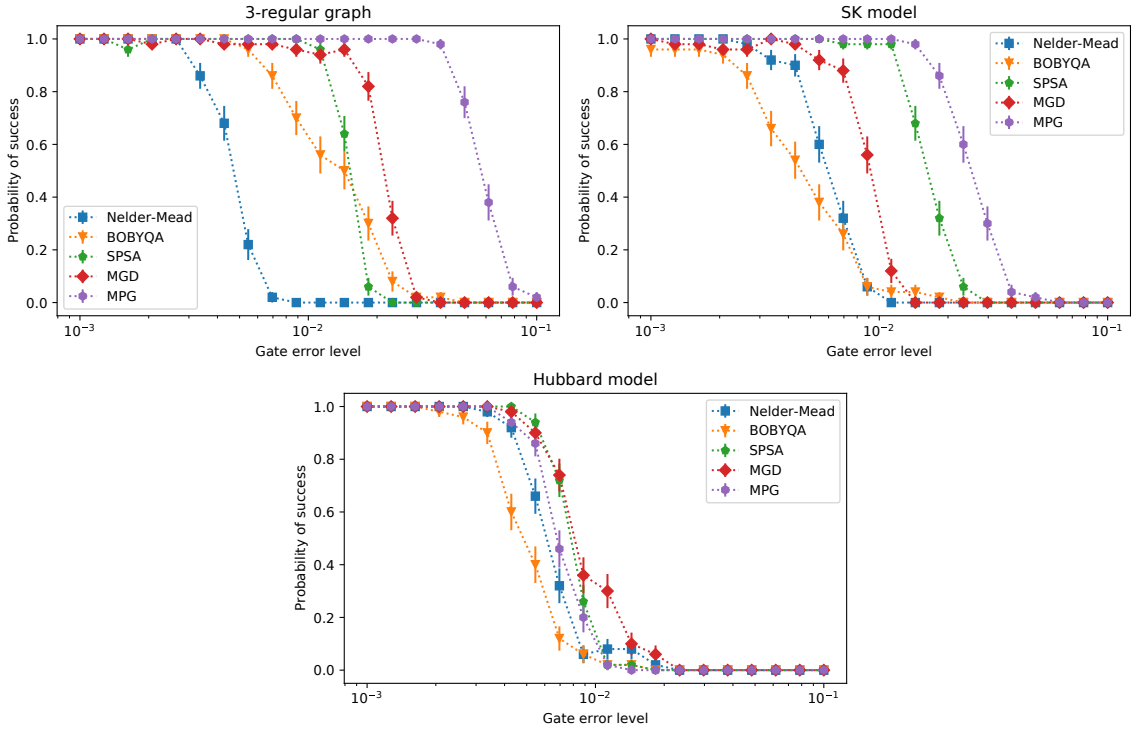


Figure D.2: Version of Figure 2.4 that also includes plots for the Sherrington-Kirkpatrick and Hubbard models.

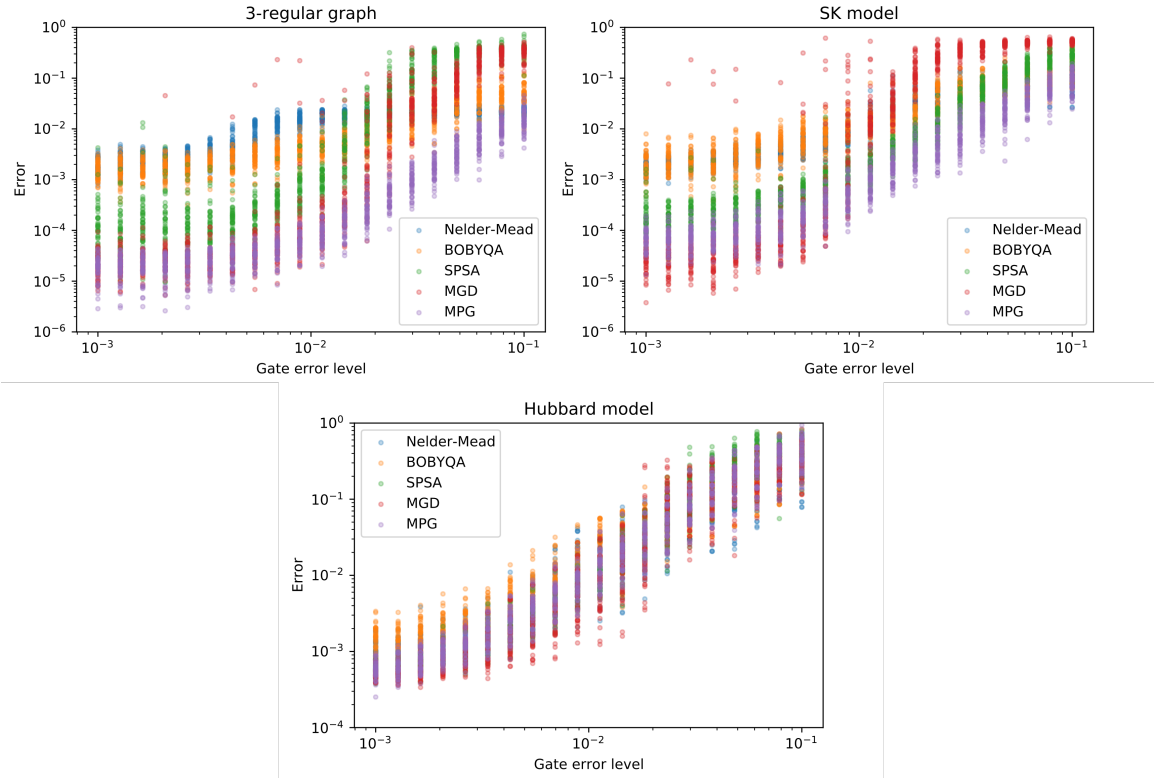


Figure D.3: Final energy error as a function of gate error level (amount of gate rotation error), for $p = 5$. For each gate error level and algorithm, the final error for the 50 runs with different PRNG seeds are plotted.

APPENDIX E

Hardware and Compilation Details for the QAOA Experiment

In this appendix, we discuss detailed compilation of the unitaries used in the experiment of Chapter 3 into the hardware native gateset, particularly the SYC gate defined in Figure 3.2c.

The SYC gate is similar to the gate used in [8] but with the conditional phase tuned to be precisely $\pi/6$. A $\sqrt{\text{iSWAP}}$ gate is simultaneously calibrated and available but has a longer gate duration and requires additional (physical) Z rotations to match phases. The required interactions for this study are compiled to an equivalent number of SYC and $\sqrt{\text{iSWAP}}$, so SYC was used in all circuits. Single-qubit microwave pulses enact "Phased X" gates PhX(θ, ϕ) (alternatively called XY rotations or the W gate) with $\phi = 0$ corresponding to $R_X(\theta)$ and $\phi = \frac{\pi}{2}$ corresponding to $R_Y(\theta)$ (up to global phase). Intermediate values of ϕ control the axis of rotation in the X-Y plane of the Bloch sphere.

Arbitrary single-qubit rotations can be applied by a PhX(θ, ϕ) gate followed by a $R_Z(\vartheta)$ gate. As a compilation step, we merge adjacent single-qubit operations to be of this form. Therefore, our circuit is structured as a repeating sequence of: a layer of PhX gates; a layer of Z gates; and a layer of SYC gates. All Z rotations of the form $\exp[-i\theta Z]$ can be efficiently commuted through SYC and PhX to the end of the circuit and discarded. This leaves alternating layers of PhX and SYC gates. The overheads of compilation are summarized in Table E.1.

Compilation of $ZZ(\gamma)$. These interactions (used for Hardware Grid and MaxCut problems) can be compiled with 2 layers of SYC gates and 2+1 associated layers of single qubit PhX gates. We report the required number of single-qubit layers as 2+1 because the initial (or final) layer from one set of interactions can be merged into the final (initial) single qubit gate layer of the preceding (following) set of interactions. In general, the number of single qubit layers will be equivalent to the number of two-qubit gate layers with one additional single-qubit layer at the beginning of the circuit and one additional single-qubit layer at the end of the circuit. The explicit compilation of ZZ to SYC is available in Cirq and a proof

Problem	Routing	Interaction	Synthesis
Hardware Grid	WESN	$e^{-i\gamma ZZ}$	2
MaxCut	Greedy	$e^{-i\gamma ZZ}$	2
MaxCut	Greedy	SWAP	3
SK Model	Swap Network	$e^{-i\gamma ZZ} \cdot$ SWAP	3

Table E.1: Compilation details for the problems studied. "Routing" gives the strategy used for routing, "Interaction" gives the type of two-qubit gates which need to be compiled, and "Synthesis" gives the number of hardware native 2-qubit SYC gates required to realize the target interaction. "WESN" routing refers to planar activation of West, East, etc. links.

can be found in the supplemental material of Ref. [8]. Here we reproduce the derivation in slightly different notation but following a similar motivation.

The SYC gate is an fSim($\pi/2$,$\pi/6$) which can be broken down into a CPHASE($\pi/6$), CZ, SWAP, and two S gates according to Figure E.1. We analyze the KAK coefficients for

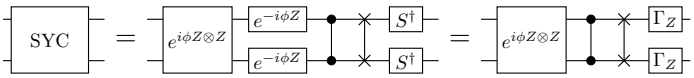


Figure E.1: Circuit decomposition of the SYC gate: $\Gamma_Z = S^\dagger e^{-i\phi Z} = e^{-i\phi Z} S^\dagger$ and $\phi = -\pi/24$, where two solid dots linked by a line represent the CZ gate and two crosses linked by a line represent the SWAP gate.

a composite gate of two SYC gates sandwiching arbitrary single qubit rotations, depicted in Figure E.2, to determine the space of gates accessible with two SYC gates.

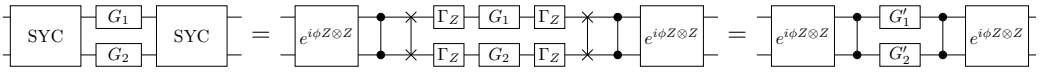


Figure E.2: Single-qubit gates sandwiched by two SYC gates: The Γ_Z gates map single-qubit operations to single-qubit operations

Any two qubit gate is locally equivalent to standard KAK form [155]. The coefficients in the KAK form is equivalent to the operator Schmidt coefficients of the 2-qubit unitary. To find the Schmidt coefficients, we introduce the matrix representation of 2-qubit gates in terms of Pauli operators, i.e., the jk-th matrix element equals to the corresponding coefficient of the Pauli operator $P_j \otimes P_k$, where $P_{0,1,2,3} = I, X, Y, Z$,

$$O_M = \sum_{j,k=0}^{3} M_{jk} P_j \otimes P_k \,. \tag{E.1}$$

The Schmidt coefficients of O_M equal to the singular values of M. Any single-qubit gate

$G'_{1,2}$ can be decomposed into the Z-X-Z rotations; the Z rotations commute with the CZ and the CPHASE, and they do not affect the Schmidt coefficients of the two-qubit operation defined in Figure E.2. We neglect the Z rotations and simplify $G'_{1,2}$ to single-qubit X rotations

$$G'_1 = \cos\theta_1 I + i \sin\theta_1 X\,, \qquad G'_2 = \cos\theta_2 I + i \sin\theta_2 X\,. \tag{E.2}$$

The Pauli matrix representation of $G'_1 \otimes G'_2$ in Eq. (E.2) is

$$A = \begin{pmatrix} c_1c_2 & ic_1s_2 & 0 & 0 \\ is_1c_2 & -s_1s_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{E.3}$$

where $c_{1,2} = \cos\theta_{1,2}$ and $s_{1,2} = \sin\theta_{1,2}$. The rank of the matrix A is one, representing a product unitary. After being conjugated by the CZ gates, i.e, $O \mapsto \text{CZ}\, O\, \text{CZ}$, the matrix A becomes

$$A \mapsto B = \begin{pmatrix} c_1c_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & is_1c_2 \\ 0 & 0 & -s_1s_2 & 0 \\ 0 & ic_1s_2 & 0 & 0 \end{pmatrix}, \tag{E.4}$$

where we use the relations for $O \mapsto \text{CZ}\, O\, \text{CZ}$,

$$X_1X_2 \mapsto Y_1Y_2\,, \quad X_1 \mapsto X_1Z_2\,, \quad X_2 \mapsto Z_1X_2\,. \tag{E.5}$$

The CPHASE part in the SYC gate is

$$e^{i\phi Z\otimes Z} = \cos\phi\, I\otimes I + i\sin\phi\, Z\otimes Z\,, \tag{E.6}$$

where $\phi = -\pi/24$. An arbitrary operator O left and right multiplied by CPHASE part is expressed as

$$e^{i\phi Z\otimes Z} O e^{i\phi Z\otimes Z} = (\cos\phi)^2 O + \frac{i}{2}\sin(2\phi)\left(Z^{\otimes 2}O + OZ^{\otimes 2}\right) - (\sin\phi)^2 Z^{\otimes 2} O Z^{\otimes 2}\,. \tag{E.7}$$

Applying the operation $O \mapsto \frac{1}{2}(Z^{\otimes 2}O + OZ^{\otimes 2})$ to the operator B, we have

$$B \mapsto C = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & s_1 s_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 c_2 \end{pmatrix} . \tag{E.8}$$

Applying the operation $O \mapsto Z^{\otimes 2} O Z^{\otimes 2}$ to the operator B, we have

$$B \mapsto D = \begin{pmatrix} c_1 c_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -i s_1 c_2 \\ 0 & 0 & -s_1 s_2 & 0 \\ 0 & -i c_1 s_2 & 0 & 0 \end{pmatrix} . \tag{E.9}$$

The resulting two-qubit gate at the output of the circuit in Figure E.2 takes the form

$$M = (\cos\phi)^2 B + i \sin(2\phi)\, C - (\sin\phi)^2 D\,. \tag{E.10}$$

Two singular values of M are $\cos(2\phi)c_1c_2$ and $\cos(2\phi)s_1s_2$ corresponding to the diagonal matrix elements $M_{0,0}$ and $M_{2,2}$, and the magnitudes of these two singular values are bounded by the angle ϕ. Consider the two-dimensional subspace of the matrix B, C, and D with the two known singular values removed

$$B \mapsto B' = \begin{pmatrix} 0 & i s_1 c_2 \\ i c_1 s_2 & 0 \end{pmatrix} , \quad C \mapsto C' = \begin{pmatrix} s_1 s_2 & 0 \\ 0 & c_1 c_2 \end{pmatrix} , \tag{E.11}$$

$$D \mapsto D' = \begin{pmatrix} 0 & -i s_1 c_2 \\ -i c_1 s_2 & 0 \end{pmatrix} \tag{E.12}$$

The Pauli representation matrix in the reduced space is

$$M' = (\cos\phi)^2 B' + i \sin(2\phi)\, C' - (\sin\phi)^2 D' \tag{E.13}$$

$$= i \begin{pmatrix} \sin(2\phi) s_1 s_2 & s_1 c_2 \\ c_1 s_2 & \sin(2\phi) c_1 c_2 \end{pmatrix} = i c_1 c_2 \begin{pmatrix} \sin(2\phi) t_1 t_2 & t_1 \\ t_2 & \sin(2\phi) \end{pmatrix} . \tag{E.14}$$

To calculate the singular values of a 2×2 matrix

$$M_{\boldsymbol{\alpha}} = \alpha_0 I + \alpha_1 X + \alpha_2 Y + \alpha_3 Z\,, \tag{E.15}$$

we used the formula

$$\sigma_{\pm} = \sqrt{\eta \pm \sqrt{\eta^2 - \xi^2}}\,, \tag{E.16}$$

where $\eta = |\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2$ and $\xi = |\alpha_0^2 - \alpha_1^2 - \alpha_2^2 - \alpha_3^2|$. For matrix M', we have,

$$\begin{aligned}
\eta &= \frac{1}{2}\sum_{j,k} |M'_{jk}|^2 && \text{(E.17)}\\
&= \frac{1}{2}\left(\sin(2\phi)^2 s_1^2 s_2^2 + s_1^2 c_2^2 + c_1^2 s_2^2 + \sin(2\phi)^2 c_1^2 c_2^2\right) && \text{(E.18)}\\
&= \frac{1}{2} - \frac{1}{2}\cos(2\phi)^2\left(s_1^2 s_2^2 + c_1^2 c_2^2\right)\,. && \text{(E.19)}
\end{aligned}$$

and

$$\begin{aligned}
\xi = \frac{1}{4}\Big|\, &\sin(2\phi)^2\left(s_1 s_2 + c_1 c_2\right)^2 - \left(s_1 c_2 + c_1 s_2\right)^2 \\
&+ \left(s_1 c_2 - c_1 s_2\right)^2 - \sin(2\phi)^2\left(s_1 s_2 - c_1 c_2\right)^2 \Big| && \text{(E.20)}\\
= \;&\cos(2\phi)^2\left|s_1 s_2 c_1 c_2\right|. && \text{(E.21)}
\end{aligned}$$

We have solved all the four singular values of the 2-qubit unitary at the output of Figure E.1,

$$\begin{aligned}
&\lambda_0 = |\cos(2\phi) c_1 c_2|, \;\; \lambda_1 = |\cos(2\phi) s_1 s_2|,\\
&\lambda_2 = \sqrt{\eta + \sqrt{\eta^2 - \xi^2}}, \;\; \lambda_3 = \sqrt{\eta - \sqrt{\eta^2 - \xi^2}}.
\end{aligned} \tag{E.22}$$

For the case $s_1 = 0$ and $c_1 = 1$, we have $\lambda_1 = \lambda_3 = 0$ and the other two singular values

$$\lambda_0 = |\cos(2\phi) c_2| \in [0,\; \cos(2\phi)]\,, \quad \lambda_2 = \sqrt{2\eta} = \sqrt{1 - \cos(2\phi)^2 c_2^2}\,. \tag{E.23}$$

Since $\cos(2\phi) \simeq 0.966 > 1/\sqrt{2}$, we can implement any CPHASE gate using only two SYC gates. This is achieved by matching the Schmidt coefficients of $e^{-i\theta ZZ/2}$ to λ_0 and λ_2. If $|\cos(\theta)| > \cos(2\phi)$ then we can reset $c_{1,2}$ and $s_{1,2}$ appropriately to select out the other pair of singular values.

Compilation of SWAP. A SWAP gate requires three applications of SYC and is used for the 3-regular MaxCut problem circuits. The SWAP gate was numerically compiled by optimizing the angles of the circuit in Figure E.3 to match the KAK interaction coefficients for the SWAP gate.

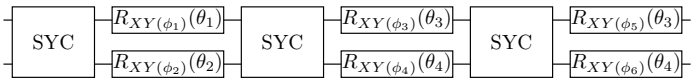


Figure E.3: Circuit used to match the KAK coefficients of the SWAP gate. The $R_{XY}(\phi)(\theta)$ is a rotation of θ around an axis in the XY-plane defined by ϕ. This is implemented in Cirq as a PhasedXPow gate.

After the the angles in the circuit depicted in Figure E.3 are determined to match the KAK coefficient of the swap gate we add single qubit rotations to make the circuit fully equivalent to SWAP.

Compilation of $e^{-i\gamma w ZZ}\cdot$ SWAP. This composite interaction can be effected with three applications of SYC and is used for SK-model circuits. The SYC gate KAK coefficients are $(\pi/4, \pi/4, \pi/24)$ which is locally equivalent to a CPHASE$(\pi/4 - \pi/24)$ followed by a SWAP. Therefore, to implement a $ZZ(\gamma)$ followed by a swap we need to apply a single SYC gate followed by the composite CPHASE$(\gamma - \pi/24 + \pi/4)$. The total composite gate now involves 3 SYC gates, a single Rx gate and two Rz gates.

Scheduling of Hardware Grid gates. An efficient planar graph edge-coloring can be used to schedule as many simultaneous ZZ interactions as possible. We activate links on the graph in the following order: 1) horizontal edges starting from even nodes; 2) horizontal edges starting from odd nodes; 3) vertical edges starting from even nodes; 4) vertical edges starting from odd nodes. Viewed as cardinal directions and choosing an even node as the central point this corresponds to a west, east, south, north (W, E, S, N) activation sequence.

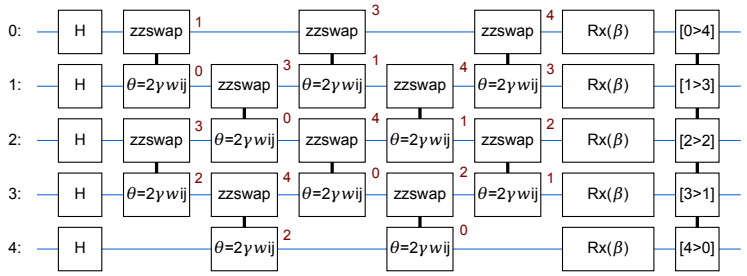


Figure E.4: $p = 1$ swap network for a 5-qubit SK-model. Physical qubits are indicated by horizontal lines and logical node indices are indicated by red numbers. The network effects all-to-all logical interactions with nearest-neighbor interactions in depth n.

Fully Connected Swap Network. All-to-all interactions can be implemented optimally with a swap network in which pairs of linear-nearest-neighbor qubits are repeatedly interacted and swapped. Crucially, the required interactions SWAP and $e^{-i\gamma ZZ}$ between all

pairs all mutually commute so we are free to re-order all two-qubit interactions to minimize compiled circuit depth. After n applications of layers of $e^{-i\gamma w ZZ} \cdot \text{SWAP}$ interactions (alternating between even and odd qubits), every qubit has been involved in a ZZ interaction with every other qubit and logical qubit indices have been reversed. This can be viewed as a (parallel) bubble sort algorithm initialized with a reverse-sorted list of logical qubit indices. An example at $n = 5$ is shown in Figure E.4. If p is even, two applications of the swap network return qubit indices to their original mapping. Otherwise, post-processing can reverse the measured bitstrings.

APPENDIX F

Correcting for Readout Error in the QAOA Experiment

The experimentally measured expectation values plotted in Figure 3.3 were adjusted with a procedure used to compensate for qubit readout error. This procedure reduced the average energy error by 46% for the hardware grid problem, 19% for the 3-regular graph problem, and 12% for the Sherrington-Kirkpatrick model.

We model readout error as a classical bit-flip error channel that changes the measurement result of qubit i from 0 to 1 with probability $p_{0,i}$ and from 1 to 0 with probability $p_{1,i}$. Under the effect of this error channel, a measurement of a single qubit in the computational basis is described by the following positive operator-valued measure (POVM) elements (we drop the subscript i here for clarity):

$$\tilde{\Pi}_0 = (1 - p_0)\Pi_0 + p_1\Pi_1 \tag{F.1}$$

$$\tilde{\Pi}_1 = p_0\Pi_0 + (1 - p_1)\Pi_1, \tag{F.2}$$

where $\Pi_0 = |0\rangle\langle 0|$, $\Pi_1 = |1\rangle\langle 1|$. The uncorrected Z observable can be written as

$$\tilde{Z} = \tilde{\Pi}_0 - \tilde{\Pi}_1 = (p_1 - p_0)I + (1 - p_1 - p_0)Z. \tag{F.3}$$

Solving for Z, we have

$$Z = \frac{\tilde{Z} - (p_1 - p_0)I}{1 - p_1 - p_0}. \tag{F.4}$$

For our problems we are interested in the two-qubit observable Z_iZ_j, so the corrected observable is

$$Z_iZ_j = \frac{\tilde{Z}_i - (p_{1,i} - p_{0,i})I}{1 - p_{1,i} - p_{0,i}} \cdot \frac{\tilde{Z}_j - (p_{1,j} - p_{0,j})I}{1 - p_{1,j} - p_{0,j}}. \tag{F.5}$$

This expression tells us how to adjust the measured observable to compensate for the readout error. In the above analysis, we can replace p_0 and p_1 by their average $(p_0 + p_1)/2$ if we perform measurements in the following way: for half of the measurements, apply a layer of X gates immediately before measuring, and then flip the measurement results. In this case, the corrected observable is

$$Z_i Z_j = \tilde{Z}_i \tilde{Z}_j \cdot \frac{1}{1 - p_{1,i} - p_{0,i}} \cdot \frac{1}{1 - p_{1,j} - p_{0,j}}. \tag{F.6}$$

We estimated the value of $p_{0,i}$ on the device by preparing and measuring the qubit in the $|0\rangle$ state 1,000,000 times and counting how often a 1 was measured; $p_{1,i}$ was estimated in the same way but by preparing the $|1\rangle$ state instead of the zero state. This estimation was performed periodically during the data collection for Figure 3.3 to account for drift following automated calibration.

At the time of the primary data collection for this experiment, all automated calibration routines were performed with each qubit in isolation. Subsequently, a calibration which optimizes qubit detunings during readout was implemented to mitigate these correlated readout errors caused by frequency collisions. Figure F.1 shows $|0\rangle$ and $|1\rangle$ state errors for *simultaneous* readout of all 23 qubits (which are used to correct $\langle ZZ \rangle$ observables) both as they were during primary data taking for Figure 3.3 (top) and after implementing the improved readout detuning calibration (bottom). During primary data collection, the median isolated readout error was 4.4% as measured during the previous automated calibration. The discrepancy between these figures and the calibration values shown in Figure F.1, top can be attributed to drift since the automated system calibration in addition to the simultaneity effects described above.

Data presented in Figure 3.4 and Figure 3.5 was taken on a different date with median isolated readout error as 4.1% as reported in the main text. Readout correction was not used for these two figures.

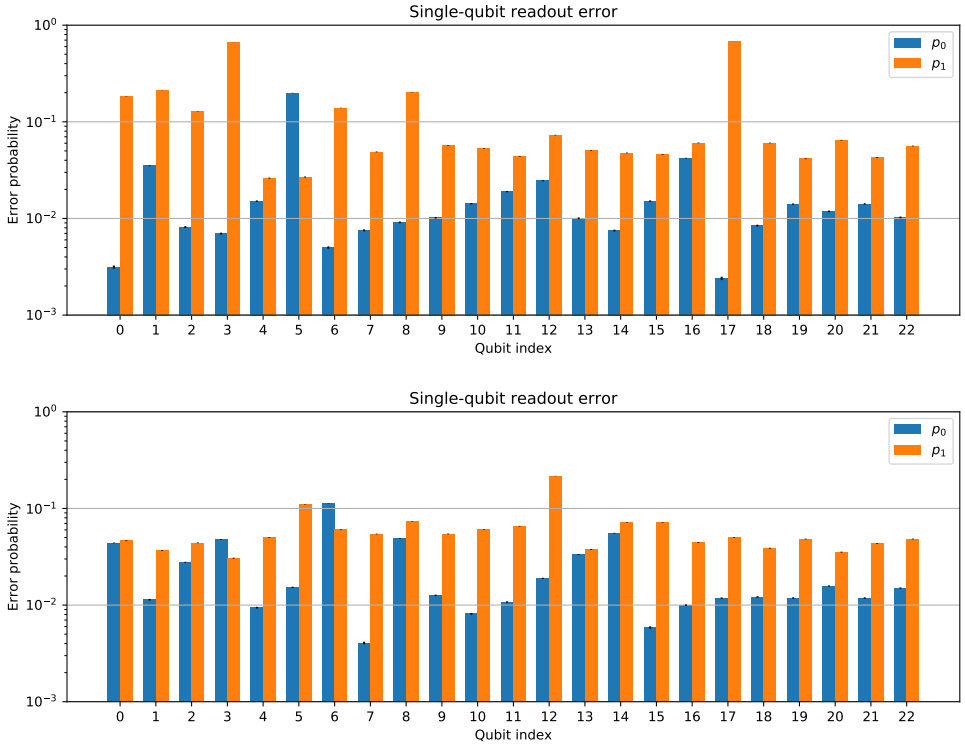


Figure F.1: (Top) Marginalized error probabilities $p_{0,i}$ and $p_{1,i}$ for simultaneous readout of all qubits from a representative calibration used to correct Figure 3.3 for readout error. (Bottom) Values for typical marginalized simultaneous readout error probabilities after the implementation of an improved automated calibration routine. Error bars (barely visible) represent a 95% confidence interval.

APPENDIX G

Analysis of Noise in the QAOA Experiment

In this appendix, we provide analysis to support the modeling of errors using the depolarizing channel for the experiments in Chapter 3 on the Sherrington-Kirkpatrick model and 3-regular graph problems.

There are two relevant mechanisms when considering the difference in performance between problems. One is the propagation of faults through the circuit and the other is fidelity decay due to circuit depth. A single fault on low-degree problems (Hardware Grid and 3-regular MaxCut, with degree four and three, respectively) can only propagate to terms p edges away from the original location of the fault, irrespective of the total number of qubits. However, if compilation results in circuits extensive in the system size, the probability of a fault increases. For the SK-model, the degree of the problem is extensive in system size so both the propensity for fault propagation as well as the probability of faults grows with n. Additionally, compilation of the 3-regular problems onto the hardware topology introduces SWAPs, which can propagate faults through nodes which would otherwise not be adjacent in the problem graph.

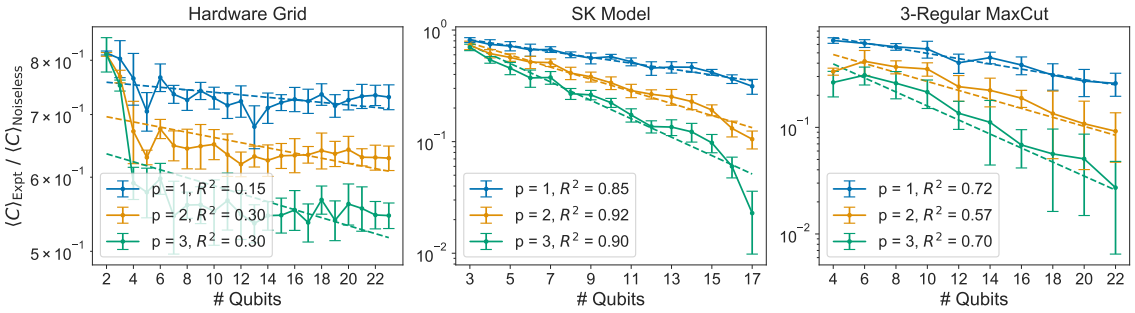


Figure G.1: An exponential model compatible with a depolarizing error channel reasonably models the performance of compiled SK Model and 3-Regular MaxCut problems because their circuits are extensive in system size and faults are rapidly mixed. This error model is a poor fit for Hardware Grid problems due to the low degree of the problem graph and simple compilation.

To probe these two effects, we fit a global depolarizing channel to the results for the three problems. A global depolarizing channel results in the mixed state

$$\rho = f_c|\psi\rangle\langle\psi| + \frac{1 - f_c}{d}\mathbf{I}$$

where $|\psi\rangle$ is the noiseless QAOA state, $\mathbf{I}$ is the n-qubit identity matrix, and f_c is the total circuit fidelity. $\mathrm{Tr}(\mathbf{I}C) = 0$ because of the ZZ structure of the cost function, so the experimental objective function is simply a scaled version of the noiseless version, $\langle C\rangle_{\mathrm{Expt}} = f_c\langle C\rangle_{\mathrm{Noiseless}}$. We perform a linear regression on $f_c = f^n \times f_0 \leftrightarrow \log(f_c) = n\log(f) + \log(f_0)$ where $\log(f)$ and $\log(f_0)$ are fittable parameters physically corresponding to a per-qubit fidelity and a qubit-independent offset. For the Hardware Grid, a depolarizing model is inappropriate, as the limited fault propagation and fixed circuit depth yield a largely n-independent noise signature. The exponential decay expected from a global depolarizing channel reasonably fits both the SK model and MaxCut results. We note that the fit is considerably stronger for the high-degree SK model where faults are rapidly mixed.

APPENDIX H

Additional Data from the QAOA Experiment

In this appendix, we give plots from experiments on the QAOA evaluated at optimal angles that were not presented in Chapter 3.

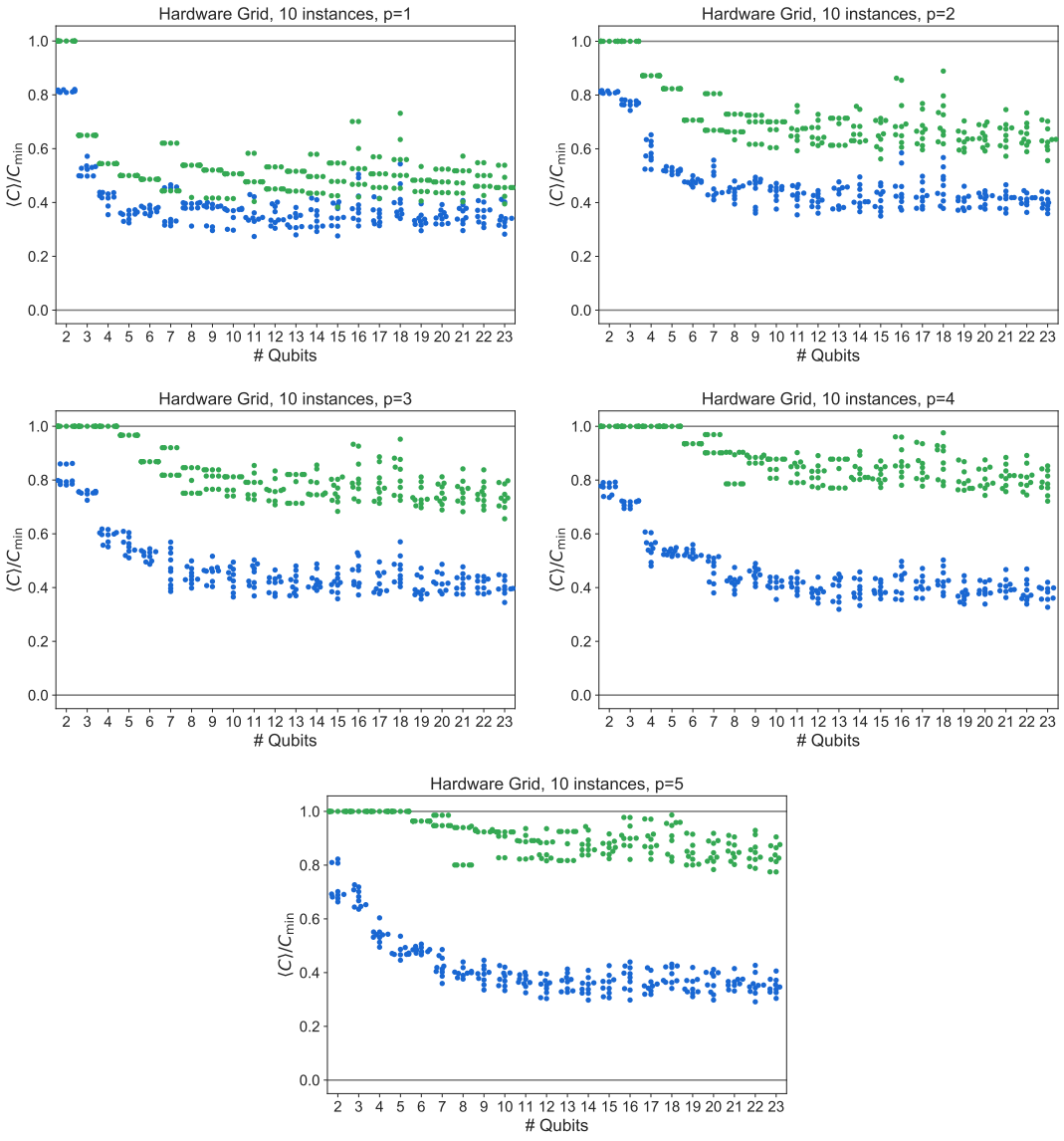


Figure H.1: Performance of QAOA at $p \in [1, 5]$ and $n \in [2, 23]$ over random instantiations of couplings as described in the main text. Points have been perturbed along the x-axis to avoid overlap. **Green:** Noiseless **Blue:** Experimental

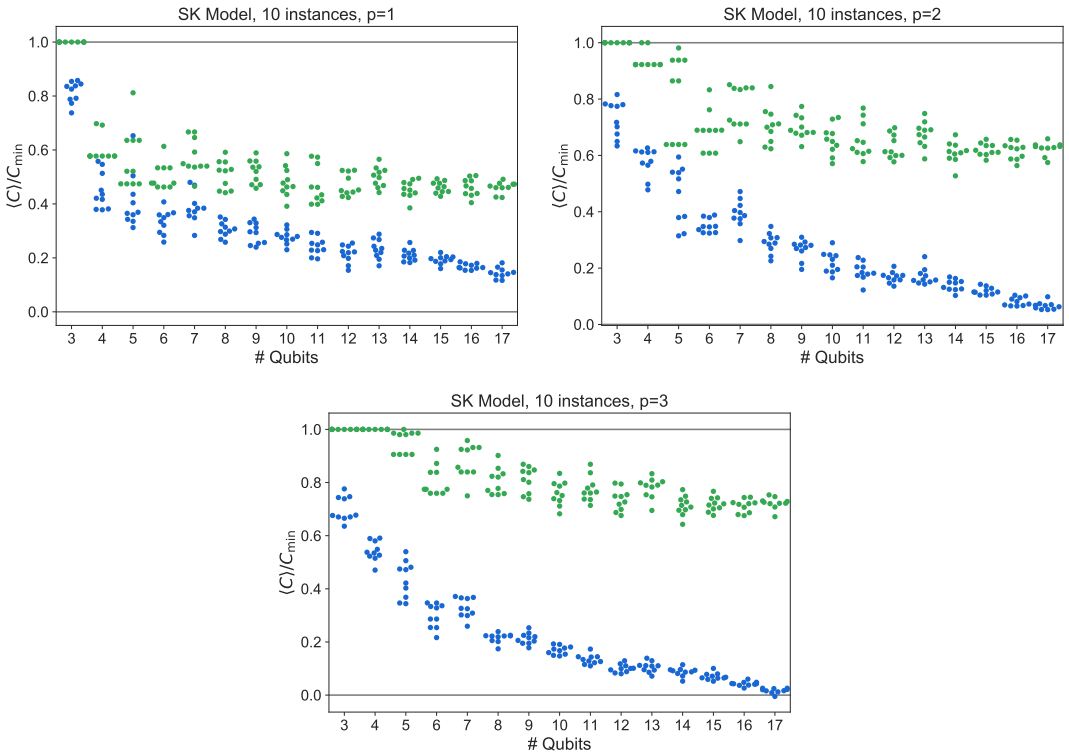


Figure H.2: Performance of QAOA at $p \in [1, 3]$ and $n \in [3, 17]$ over random SK model instances as described in the main text. Points have been perturbed along the x-axis to avoid overlap. **Green:** Noiseless **Blue:** Experimental

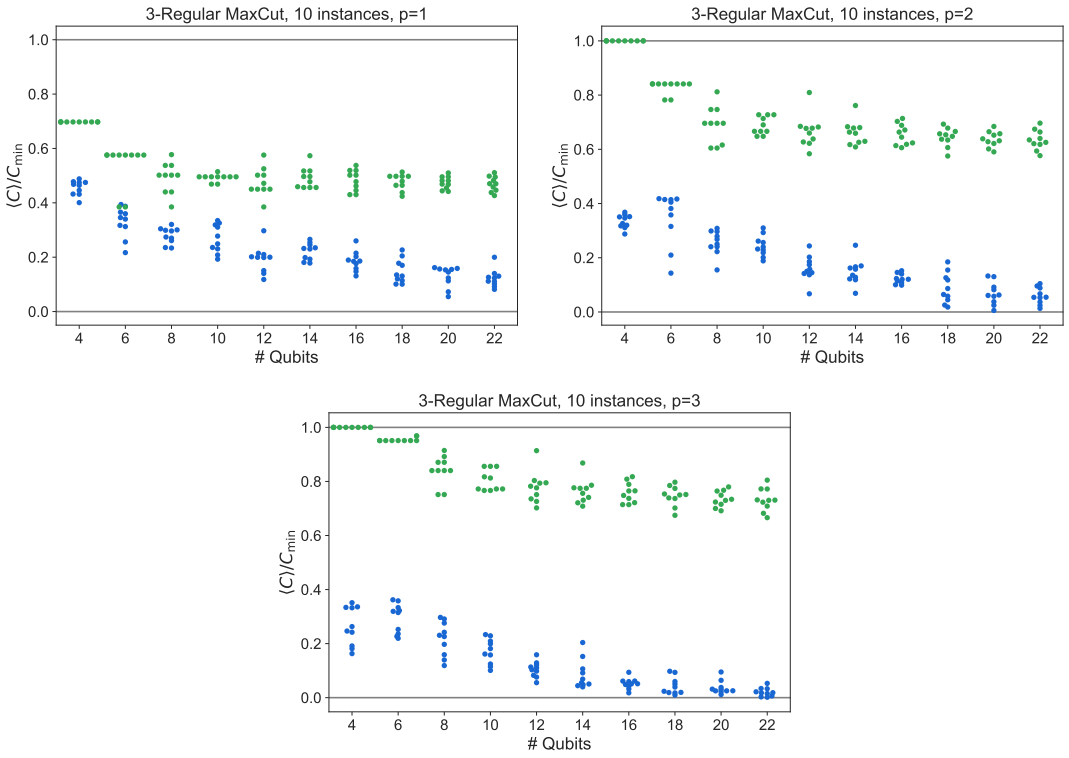


Figure H.3: Performance of QAOA at $p \in [1, 3]$ and $n \in [4, 22]$ over random 3-regular MaxCut problems as described in the main text. Points have been perturbed along the x-axis to avoid overlap. k-regular graphs must satisfy $n \geq k + 1$ and nk must be even, hence only even n are considered here. **Green:** Noiseless **Blue:** Experimental

APPENDIX I

Measuring the 1-RDM in the Hartree-Fock Experiment

In this appendix, we describe how we measured the 1-RDM for the experiment of Chapter 5 in a way that allowed for post-selection on the total particle number, using $N+1$ distinct circuits. For each circuit, we performed 250,000 measurements. The 1-RDM is an $N \times N$ hermitian positive semidefinite matrix with elements equal to the expectation values $\langle a_i^\dagger a_j \rangle$ where $\{i, j\}$ index the row and column of the matrix. The matrix of expectation values is depicted in Figure I.1. As a motivator for our measurement protocol we start by describing circuits required to measure the diagonal elements of the 1-RDM of a six qubit system at half filling–i.e. $\langle a_i^\dagger a_i \rangle$.

I.1 Diagonal terms

Given a circuit U implementing the basis rotation e^κ the diagonal elements of the 1-RDM are obtained by measuring the Z expectation value on each qubit. The correspondence between $a_i^\dagger a_i$, measurement result M_i from qubit i, qubit operators is derived using the Jordan-Wigner transform

$$\langle a_i^\dagger a_i \rangle = \frac{I - \langle Z_i \rangle}{2} = \langle M_i \rangle \tag{I.1}$$

where Z_i is the Z-qubit operator on qubit labeled i. The expectation value $\langle a_i^\dagger a_i \rangle$ is equivalent to the probability of measuring a 1 bit on qubit i–i.e $\langle M_i \rangle$. Because we are measuring in the computational basis we can post-select on the three excitations in the measurement result. This process is depicted in Figure I.1.

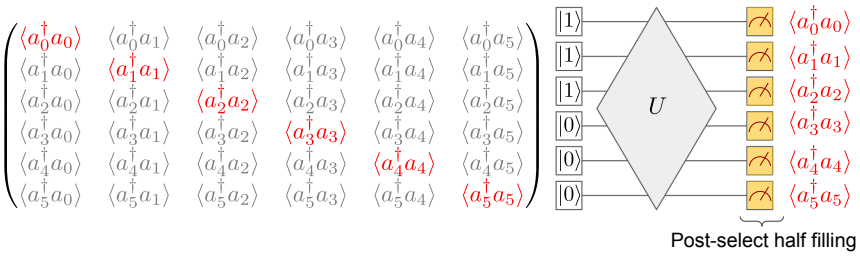


Figure I.1: Measurement circuit associated with estimating all diagonal elements of the 1-RDM simultaneously. The elements that are acquired with this circuit are highlighted in red.

I.2 One-off-diagonal terms

The hermiticity of the 1-RDM demands that $\langle a_i^\dagger a_{i+1}\rangle = \langle a_{i+1}^\dagger a_i\rangle^*$. The 1-RDM has no imaginary component because we use an initial basis built from real valued orbitals and the basis rotation circuit implements an element of $SO(N)$–i.e. the basis rotation circuit involves a unitary matrix with real values. Therefore, we only measure the real part of all one-off-diagonal terms $a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i$ which corresponds to $2\Re\langle a_i^\dagger a_{i+1}\rangle$. Using the Jordan-Wigner transform to map fermionic ladder operators to qubits

$$\langle a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i\rangle = \frac{1}{2}\left(\langle X_i X_{i+1}\rangle + \langle Y_i Y_{i+1}\rangle\right) = 2\Re\langle a_i^\dagger a_{i+1}\rangle \tag{I.2}$$

we see that we must measure XX on all pairs and YY on all pairs. This measurement can be accomplished with two circuits depicted in Figure I.2.

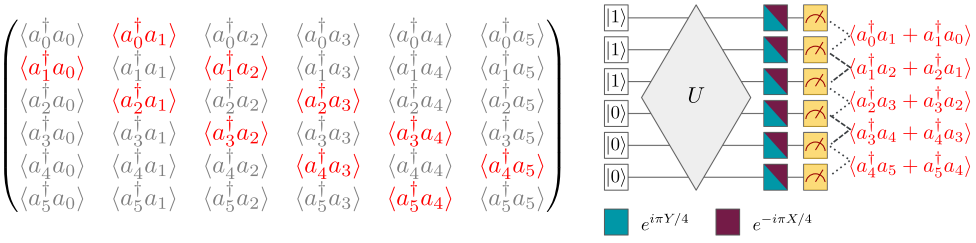


Figure I.2: The two circuits allowing for the measurement of all one-off-diagonal elements of the 1-RDM simultaneously. The teal circuit involves performing an Ry rotation (to measure in the X basis) at the end of the circuit and the purple circuit contains an Rx rotation (to measure in the Y basis). The 1-RDM elements that are acquired with these circuits are highlighted in red. We label which pairs contribute to which expectation values with grey dashed lines. The thinner dashes are for the even 1-RDM pairs and the thicker dashes are for the odd 1-RDM pairs. Because Ry and Rx operations do not preserve particle number we cannot post-select on total particle number with these measurement circuits.

I.3 General off-diagonal terms and virtual swapping

The label of each fermionic mode is an arbitrary choice, so we are free to reorder the labels such that measuring nearest-neighbor pairs of qubits corresponds to measuring different off-diagonal 1-RDM elements. Every relabeling of the qubits requires us to recompile the Givens rotation circuit. The structure of the circuit stays the same but the rotation angles are different. In this section we describe how to recompute the Givens rotation angles based on a new label ordering. Using the label sets $\{1, 3, 0, 5, 2, 4\}$ and $\{3, 5, 1, 4, 0, 2\}$ we are able to use the two measurement circuits in Figure I.2 to measure the remaining off-diagonal 1-RDM elements.

Formally, we build the new qubit labels by virtually swapping fermionic modes at the end of the original circuit implement e^{κ}. We note that performing nearest-neighbor fermionic swaps between adjacent pairs twice (even swaps and odd swaps) we obtain a new ordering of qubits. For example, consider six fermionic modes $\{0, 1, 2, 3, 4, 5\}$. Performing a set of fermionic swaps on modes labeled $\{(0, 1), (2, 3), (4, 5)\}$ followed by swaps on $\{(1, 2), (3, 4)\}$ leaves our mode ordering as $\{1, 3, 0, 5, 2, 4\}$. We can then perform X-Pauli and Y-Pauli measurements on each qubit to recover expectation values associated with

$$\{\Re(a_1^\dagger a_3 + a_3^\dagger a_1), \Re(a_3^\dagger a_0 + a_0^\dagger a_3), \Re(a_0^\dagger a_5 + a_5^\dagger a_0), \Re(a_5^\dagger a_2 + a_2^\dagger a_5), \Re(a_2^\dagger a_4 + a_4^\dagger a_2)\}. \tag{I.3}$$

This procedure can be repeated once more to measure all the required two-body fermionic correlators to construct the 1-RDM. In general, $N/2$ settings of fermionic swaps (including the trivial setting with no swaps) are needed to measure all of the off-diagonal terms.

At first glance, the addition of fermionic swaps appears to incur additional overhead in executing the circuit. However, we can exploit the fact that one-body fermionic swaps are generated by $\exp(-i\pi\text{FSWAP}/2)$ where FSWAP is

$$\text{FSWAP} = a_p^\dagger a_q + a_q^\dagger a_p - a_p^\dagger a_p - a_q^\dagger a_q. \tag{I.4}$$

This one-body permutation can be viewed as a basis rotation which can be concatenated with the original circuit at no extra cost due to (5.32). The swapping unitary simply shuffles the columns of e^{κ} that is used to generate the Givens rotation network. The same effect could have been achieved by relabeling the fermionic modes which would have been equivalent to permuting the rows and columns of e^{κ}. This relabeling technique can be applied beyond basis rotation circuits. For example, one can relabel the fermionic modes of a generalized swap network such that different sets of RDM elements can be measured as

nearest-neighbor pairs. The same logic can be applied to k-RDM elements.

Each of the $N/2$ fermionic swap settings gives rise to two circuits needed to measure the expectation values, so a total of N circuits are needed to measure the off-diagonal 1-RDM elements. This, combined with the single circuit needed to measure the diagonal elements, yields a total of $N+1$ circuits needed to measure the 1-RDM. This is a quadratic improvement over the naive measurement scheme that uses $\Theta(N^2)$ different measurement settings.

I.4 Off-diagonal terms with post-selection

The circuits depicted in Figure I.2 did not allow for post-selection because the rotations to measure in the X-basis and Y-basis do not commute with the total number operator. In this section we design a basis rotation circuit that commutes with the total number operator and diagonalizes the $\frac{1}{2}(XX+YY)$ Hamiltonian. The diagonal form means that after performing the basis rotation we can measure in the computational basis to obtain expectation values $\frac{1}{2}\langle XX+YY\rangle$.

The circuit that diagonalizes $\frac{1}{2}(XX+YY)$ is described in Figure I.3 and is denoted U_M below. Its commutation with the total number operator can be easy seen by recognizing that the T-gate (Rz$(\pi/4)$) commutes with the total number operator and so does the $\sqrt{i\text{SWAP}}$. Applying U_M to the $\frac{1}{2}(XX+YY)$ Hamiltonian

$$U_M \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} U_M^\dagger = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{I.5}$$

transforms the operator into a diagonal representation. Given an ordered pair of qubits $\{a, a+1\}$ the last matrix in (I.5) is $\frac{1}{2}(Z_a - Z_{a+1})$ in qubit representation. Finally, we can relate the Z expectation values, the transformed $XX+YY$ expectation values, fermionic ladder operators, and binary measurements $\{M_a, M_{a+1}\}$ via

$$\begin{aligned} \langle U_m \left(a_a^\dagger a_{a+1} + a_{a+1}^\dagger a_a\right) U_m^\dagger\rangle &= \langle U_m \frac{1}{2}\left(X_a X_{a+1} + Y_a Y_{a+1}\right) U_m^\dagger\rangle \\ &= \frac{1}{2}\langle Z_a - Z_{a+1}\rangle \\ &= \frac{1}{2}\left(M_{a+1} - M_a\right). \end{aligned} \tag{I.6}$$

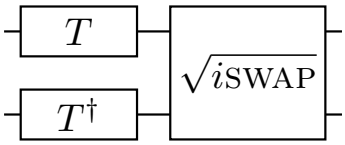


Figure I.3: Two-mode fermionic fast Fourier transform that diagonalizes the $XX + YY$ Hamiltonian.

The measurement circuit can only be applied to non-overlapping pairs and thus we can obtain estimates of $X_a X_{a+1} + Y_a Y_{a+1}$ for a values corresponding to even integers or a corresponding to odd integers. More concretely, we describe this process in Figure I.4 for a six qubit problem. All experiments involved circuits that allowed for post-selection based on total Hamming weight. The "raw" data indicates analysis of the resulting bitstrings without post-selection.

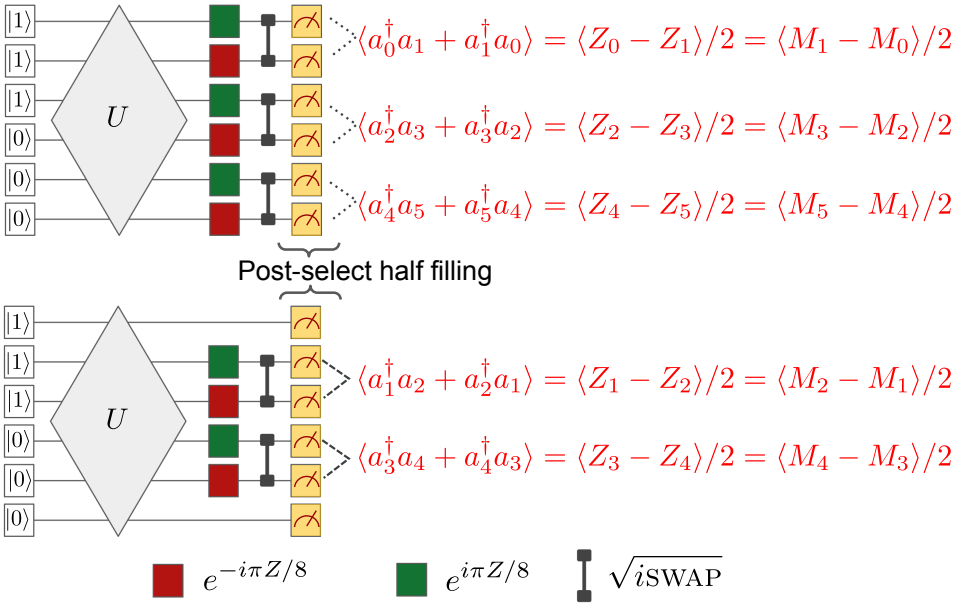


Figure I.4: Two circuits measuring the one-off-diagonal of the 1-RDM such that the total particle number can be measured simultaneously. This circuit allows us to post select on the correct number of excitations in the measured bitstring. The top circuit measures the even pairs and the bottom circuit measures the odd pairs. Local Z expectation values are measured on all the qubits and used to construct the expectation value for $\langle a_i^\dagger a_{i+1}\rangle$.

I.5 Computing error bars for elements of the 1-RDM

We use two methods to estimate error bars for all quantities in our experiments. The procedures differ in how the covariance between 1-RDM terms is estimated. In the first procedure, error bars are generated by estimating the covariance between terms in the 1-RDM at the same time as the mean estimation. Mean values of off-diagonal 1-RDM terms involve estimating the expectation values for $(Z_a - Z_b)/2$. Therefore, the covariance between two off-diagonal elements of the 1-RDM is

$$\begin{aligned} &\mathrm{Cov}\left[\frac{1}{2}\left(Z_a - Z_b\right), \frac{1}{2}\left(Z_p - Z_q\right)\right] \\ &\quad = \frac{1}{4}(\mathrm{Cov}\left[Z_a, Z_p\right] - \mathrm{Cov}\left[Z_a, Z_q\right] - \mathrm{Cov}\left[Z_b, Z_p\right] + \mathrm{Cov}\left[Z_b, Z_q\right]) \end{aligned} \tag{I.7}$$

for all pair sets $\{(a,b),(p,q)\}$ measured simultaneously. All quantities can be estimated from the simultaneous measurement of all qubits. Therefore, for each circuit permutation we obtain two covariance matrices of size $(N/2) \times (N/2)$ and $(N/2-1) \times (N/2-1)$, when N is even. For the circuit with no label permutation we also obtain the covariances for all $a_i^\dagger a_i$ terms.

In the second procedure for estimating covariance matrices we assume we are sampling from a pure Gaussian state. This assumption is applicable when the fidelity is high enough as any change to the covariance matrix would be a second order effect. For these states the 2-RDM is exactly described by the 1-RDM and therefore all covariances between the 1-RDM elements are perfectly defined by a nonlinear function of the 1-RDM elements. For any wavefunction ψ corresponding to the output of a basis rotation circuit the covariance of 1-RDM elements computed from such a wavefunction are as follows:

$$\begin{aligned} &\mathrm{Cov}\left[a_i^\dagger a_j + a_j^\dagger a_i, a_p^\dagger a_q + a_q^\dagger a_p\right]_\psi \\ &\quad = D_q^i \delta_p^j - D_q^i D_j^p + D_p^i \delta_q^j - D_p^i D_j^q + D_q^j \delta_p^i - D_q^j D_i^p + D_p^j \delta_q^i - D_p^j D_i^q. \end{aligned} \tag{I.8}$$

With the estimates of the covariances we are able to re-sample the 1-RDM assuming central-limit theorem statistics. We use a multinomial distribution where the mean values are $\langle a_{\sigma(i)}^\dagger a_{\sigma(i+1)} \rangle$ and the covariance matrix of the multinomial distribution is obtained by dividing the estimates of the covariance matrix above by $\alpha\times$ 250,000. α is a number less than 1 reflecting the probability that a bitstring is rejected. α is estimated from prior N-qubit experiments. Once the new 1-RDM is obtained it can be purified, used to estimate a fidelity witness, and compute the energy. For all error bars we re-sample the 1-RDM 1000 times and compute a mean value and standard deviation from this set. All quantities

estimated are sensitive to the N-representability of the resampled 1-RDM. We use the fixed trace positive projection described in [156] to ensure that each resampled 1-RDM is positive semidefinite and has the correct trace. The correction procedure is only applied when the resampled 1-RDM has eigenvalues below zero.

APPENDIX J

Computing a Fidelity Witness for the Hartree-Fock Experiment

In this appendix, we describe how we obtained the fidelity witnesses used in 5. Efficient fidelity witnesses exist for quantum circuits simulating non-interacting fermion dynamics. The formal derivation for general non-interacting fermion wavefunctions is described in Ref. [124]. Here we adapt this result to the special case of particle conserving dynamics generated by one-body fermionic generators.

A fidelity witness is an observable that provides a strict lower bound to the fidelity for all input states. The fidelity witness is efficient in the sense that for an L-qubit system only L^2 expectation values are required to evaluate the fidelity witness. Given that U is a unitary corresponding to a basis transformation circuit and $|\omega\rangle$ is the initial computational basis state corresponding to $\omega = (\omega_1, ..., \omega_L)$ any L-bit string which satisfies $n_j|\omega\rangle = \omega_j|\omega\rangle$ for $j = 1, ..., L$ allows us to define a basis state annihilator operator

$$n^{(\omega)} = \sum_{j=1}^{L} [(1-\omega_j)n_j + \omega_j(\mathbb{I} - n_j)] \tag{J.1}$$

$$= \sum_{j=1}^{L} [n_j - \omega_j n_j + \omega_j \mathbb{I} - \omega_j n_j] \tag{J.2}$$

$$= \sum_{j=1}^{L} [n_j + \omega_j \mathbb{I} - 2\omega_j n_j] \tag{J.3}$$

which satisfies $n^{(\omega)}|\omega\rangle = 0$. The computational basis state $|\omega\rangle$ is the zero energy eigenstate of n^{ω} and any other computational basis state is an excitation from this state. The excitation energy is exactly the number of bits that are different from ω for each Fock basis state which can be computed by summing the resulting bit string from the XOR operation between the

two Fock basis states being considered. The fidelity witness

$$\mathcal{W} = U\left(\mathbb{I} - n^{\omega}\right)U^{\dagger} \tag{J.4}$$

can be evaluated with knowledge of the measured 1-RDM. To relate the fidelity witness to the 1-RDM it is important to note the following

$$\mathrm{Tr}\left[U\rho_p U^{\dagger}a_i^{\dagger}a_j\right] = \left[\mathbf{u}D\mathbf{u}^{\dagger}\right]_{i,j} \tag{J.5}$$

where D is the matrix of expectation values $\langle \rho_p, a_i^{\dagger}a_j\rangle$ and $\mathbf{u} = e^{\kappa}$ because any one-body rotation on the state ρ_p can be equated to a similarity transform of the generating matrix for that one-body transformation. This logic is similar to the logic used in [157] which moved one-body basis rotations at the end of the circuit into the Hamiltonian as an error mitigation technique. Using this relationship we can evaluate the fidelity witness with the following expression

$$F_{\mathcal{W}}(\rho_p) = 1 - \sum_{j=1}^{L}\left(\left[\mathbf{u}^{\dagger}\mathbf{D}\mathbf{u}\right]_{j,j} + \omega_j - 2\omega_j\left[\mathbf{u}^{\dagger}\mathbf{D}\mathbf{u}\right]_{j,j}\right) \tag{J.6}$$

where $\mathbf{D}$ is the 1-RDM that is measured, $\mathbf{u} = e^{\kappa}$ is the unitary rotation representing the new Slater determinant.

BIBLIOGRAPHY

[1] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics* **21** no. 6, (Jun, 1982) 467–488.

[2] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, p. 124–134. IEEE Computer Society, USA, 1994.

[3] S. Lloyd, "Universal quantum simulators," *Science* **273** no. 5278, (1996) 1073–1078.

[4] M.-H. Yung, J. D. Whitfield, S. Boixo, D. G. Tempel, and A. Aspuru-Guzik, "Introduction to quantum algorithms for physics and chemistry," in *Quantum Information and Computation for Chemistry*, pp. 67–106. John Wiley & Sons, Ltd, 2014.

[5] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, "Elucidating reaction mechanisms on quantum computers," *Proceedings of the National Academy of Sciences* **114** no. 29, (2017) 7555–7560, https://www.pnas.org/content/114/29/7555.full.pdf.

[6] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," arXiv:0904.2557 [quant-ph].

[7] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum* **2** (2018) 79.

[8] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum

supremacy using a programmable superconducting processor," *Nature* **574** no. 7779, (2019) 505–510.

[9] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," arXiv:1411.4028 [quant-ph].

[10] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics* **18** no. 2, (2016) 23023.

[11] R. Colbeck, *Quantum And Relativistic Protocols For Secure Multi-Party Computation*. PhD thesis, University of Cambridge, 2006. arXiv:0911.3814 [quant-ph].

[12] K. J. Sung, M. P. Harrigan, N. C. Rubin, Z. Jiang, R. Babbush, and J. R. McClean, "An exploration of practical optimizers for variational quantum algorithms on superconducting qubit processors," arXiv:2005.11011 [quant-ph].

[13] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. Dunsworth, E. Farhi, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, S. Habegger, M. P. Harrigan, A. Ho, S. Hong, T. Huang, L. B. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P. V. Klimov, A. N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, M. Lindmark, M. Leib, E. Lucero, O. Martin, J. M. Martinis, J. R. McClean, M. McEwen, A. Megrant, X. Mi, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, M. Neeley, C. Neill, F. Neukart, H. Neven, M. Y. Niu, T. E. O'Brien, B. O'Gorman, E. Ostby, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, A. Skolik, V. Smelyanskiy, D. Strain, M. Streif, K. J. Sung, M. Szalay, A. Vainsencher, T. White, Z. J. Yao, P. Yeh, A. Zalcman, and L. Zhou, "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," arXiv:2004.04197 [quant-ph].

[14] Z. Jiang, K. J. Sung, K. Kechedzhi, V. N. Smelyanskiy, and S. Boixo, "Quantum algorithms to simulate many-body physics of correlated fermions," *Phys. Rev. Applied* **9** (Apr, 2018) 044036.

[15] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. Dunsworth, E. Farhi, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, S. Habegger, M. P. Harrigan, A. Ho, S. Hong, T. Huang, W. J. Huggins, L. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P. V. Klimov, A. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, M. Lindmark, E. Lucero, O. Martin, J. M. Martinis, J. R. McClean, M. McEwen, A. Megrant, X. Mi, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, M. Neeley, C. Neill,

H. Neven, M. Y. Niu, T. E. O'Brien, E. Ostby, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, D. Strain, K. J. Sung, M. Szalay, T. Y. Takeshita, A. Vainsencher, T. White, N. Wiebe, Z. J. Yao, P. Yeh, and A. Zalcman, "Hartree-Fock on a superconducting qubit quantum computer," arXiv:2004.04174 [quant-ph].

[16] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, S. Boixo, L. T. A. N. Brandão, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, J. Chau, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. Derk, A. Dunsworth, D. Eppens, E. Farhi, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, S. Habegger, M. P. Harrigan, A. Ho, S. Hong, T. Huang, L. B. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P. V. Klimov, A. N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, G. Laun, M. Lindmark, E. Lucero, O. Martin, J. M. Martinis, J. R. McClean, M. McEwen, A. Megrant, X. Mi, C. A. Miller, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, M. Neeley, C. Neill, H. Neven, M. Y. Niu, T. E. O'Brien, E. Ostby, R. Peralta, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, D. Strain, K. J. Sung, M. Szalay, A. Vainsencher, T. White, Z. J. Yao, P. Yeh, and A. Zalcman, "Experimental blueprint for certifiable randomness from a superconducting quantum processor,".

[17] J. R. McClean, N. C. Rubin, K. J. Sung, I. D. Kivlichan, X. Bonet-Monroig, Y. Cao, C. Dai, E. S. Fried, C. Gidney, B. Gimby, P. Gokhale, T. Häner, T. Hardikar, V. Havlíček, O. Higgott, C. Huang, J. Izaac, Z. Jiang, X. Liu, S. McArdle, M. Neeley, T. O'Brien, B. O'Gorman, I. Ozfidan, M. D. Radin, J. Romero, N. P. D. Sawaya, B. Senjean, K. Setia, S. Sim, D. S. Steiger, M. Steudtner, Q. Sun, W. Sun, D. Wang, F. Zhang, and R. Babbush, "OpenFermion: the electronic structure package for quantum computers," *Quantum Science and Technology* **5** no. 3, (Jun, 2020) 034014.

[18] A. Rahmani, K. J. Sung, H. Putterman, P. Roushan, P. Ghaemi, and Z. Jiang, "Creating and manipulating a Laughlin-type $\nu = 1/3$ fractional quantum hall state on a quantum computer with linear depth circuits," arXiv:2005.02399 [cond-mat.str-el].

[19] G. Verdon, M. Broughton, J. R. McClean, K. J. Sung, R. Babbush, Z. Jiang, H. Neven, and M. Mohseni, "Learning to learn with quantum neural networks via classical neural networks," arXiv:1907.05415 [quant-ph].

[20] 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** no. 4213, (2014) 1–7.

[21] C. Hempel, C. Maier, J. Romero, J. McClean, T. Monz, H. Shen, P. Jurcevic, B. P. Lanyon, P. Love, R. Babbush, A. Aspuru-Guzik, R. Blatt, and C. F. Roos, "Quantum chemistry calculations on a trapped-ion quantum simulator," *Phys. Rev. X* **8** (Jul, 2018) 031022.

[22] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal* **7** no. 4, (01, 1965) 308–313.

[23] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature* **549** no. 7671, (Sep, 2017) 242–246.

[24] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, C. A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, R. S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, B. R. Johnson, M. Reagor, M. P. d. Silva, and C. Rigetti, "Unsupervised machine learning on a hybrid quantum computer," arXiv:1712.05771 [quant-ph].

[25] J. I. Colless, V. V. Ramasesh, D. Dahlen, M. S. Blok, M. E. Kimchi-Schwartz, J. R. McClean, J. Carter, W. A. de Jong, and I. Siddiqi, "Computation of molecular spectra on a quantum processor with an error-resilient algorithm," *Phys. Rev. X* **8** (Feb, 2018) 011021.

[26] 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** (May, 2019) 355–360.

[27] G. Pagano, A. Bapat, P. Becker, K. S. Collins, A. De, P. W. Hess, H. B. Kaplan, A. Kyprianidis, W. L. Tan, C. Baldwin, L. T. Brady, A. Deshpande, F. Liu, S. Jordan, A. V. Gorshkov, and C. Monroe, "Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator," arXiv:1906.02700 [quant-ph].

[28] J. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control* **37** no. 3, (Mar., 1992) 332–341.

[29] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE* **104** (Jan, 2016) 148–175.

[30] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing* **1** no. 2, (2002) 235–306.

[31] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the Lipschitz constant," *Journal of Optimization Theory and Applications* **79** (Oct, 1993) 157–181.

[32] D. Wecker, M. B. Hastings, and M. Troyer, "Progress towards practical quantum variational algorithms," *Phys. Rev. A* **92** (Oct, 2015) 042303.

[33] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," arXiv:1812.01041 [quant-ph].

[34] K. M. Nakanishi, K. Fujii, and S. Todo, "Sequential minimal optimization for quantum-classical hybrid algorithms," arXiv:1903.12166 [quant-ph].

[35] R. M. Parrish, J. T. Iosue, A. Ozaeta, and P. L. McMahon, "A Jacobi diagonalization and Anderson acceleration algorithm for variational quantum algorithm parameter optimization," arXiv:1904.03206 [quant-ph].

[36] J. M. Kübler, A. Arrasmith, L. Cincio, and P. J. Coles, "An adaptive optimizer for measurement-frugal variational algorithms," arXiv:1909.09083 [quant-ph].

[37] A. Arrasmith, L. Cincio, R. D. Somma, and P. J. Coles, "Operator sampling for shot-frugal optimization in variational algorithms," arXiv:2004.06252 [quant-ph].

[38] G. G. Guerreschi and M. Smelyanskiy, "Practical optimization for hybrid quantum-classical algorithms," arXiv:1701.01450 [quant-ph].

[39] J. Romero, R. Babbush, J. McClean, C. Hempel, P. Love, and A. Aspuru-Guzik, "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz," arXiv:1701.02691 [quant-ph].

[40] G. Nannicini, "Performance of hybrid quantum-classical variational heuristics for combinatorial optimization," *Phys. Rev. E* **99** (Jan, 2019) 013304.

[41] J. Yao, M. Bukov, and L. Lin, "Policy gradient based quantum approximate optimization algorithm," arXiv:2002.01068 [quant-ph].

[42] W. Lavrijsen, A. Tudor, J. Müller, C. Iancu, and W. de Jong, "Classical optimizers for noisy intermediate-scale quantum devices," arXiv:2004.03004 [quant-ph].

[43] Z. Leng, P. Mundada, S. Ghadimi, and A. Houck, "Robust and efficient algorithms for high-dimensional black-box quantum optimization," arXiv:1910.03591 [quant-ph].

[44] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing* **16** (1995) 1190–1208.

[45] M. J. D. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in Optimization and Numerical Analysis*, S. Gomez and J.-P. Hennart, eds., pp. 51–67. Springer Netherlands, Dordrecht, 1994.

[46] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal* **7** no. 2, (01, 1964) 155–162.

[47] A. Costa and G. Nannicini, "RBFOpt: an open-source library for black-box optimization with costly function evaluations," *Mathematical Programming Computation* **10** (2018) 597–629.

[48] W. Huyer and A. Neumaier, "SNOBFIT – stable noisy optimization by branch and fit," *ACM Trans. Math. Softw.* **35** no. 2, (July, 2008) .

[49] M. J. D. Powell, "The BOBYQA algorithm for bound constrained optimization without derivatives," 2009. `http://www.damtp.cam.ac.uk/user/na/NA_papers/NA2009_06.pdf`.

[50] S. Le Digabel, "Algorithm 909: Nomad: Nonlinear optimization with the mads algorithm," *ACM Trans. Math. Softw.* **37** no. 4, (Feb., 2011) .

[51] C. T. Kelley, *Implicit Filtering*. Society for Industrial and Applied Mathematics, 2011.

[52] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning* **8** no. 3, (1992) 229–256.

[53] J. Kempe, A. Kitaev, and O. Regev, "The complexity of the local Hamiltonian problem," *SIAM Journal on Computing* **35** no. 5, (2006) 1070–1097.

[54] F. G. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven, "For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances," `arXiv:1812.04170 [quant-ph]`.

[55] D. Sherrington and S. Kirkpatrick, "Solvable model of a spin-glass," *Phys. Rev. Lett.* **35** (Dec, 1975) 1792–1796.

[56] J. Hubbard, "Electron correlations in narrow energy bands," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* **276** no. 1365, (1963) 238–257.

[57] E. Dagotto, "Correlated electrons in high-temperature superconductors," *Reviews of Modern Physics* **66** no. 3, (July, 1994) 763–840.

[58] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush, "Quantum simulation of electronic structure with linear depth and connectivity," *Phys. Rev. Lett.* **120** (Mar, 2018) 110501.

[59] P. Jordan and E. Wigner, "Über das Paulische äquivalenzverbot," *Zeitschrift für Physik* **47** no. 9-10, (1928) 631–651.

[60] A. Jena, S. Genin, and M. Mosca, "Pauli partitioning with respect to gate sets," 2019.

[61] A. F. Izmaylov, T.-C. Yen, and I. G. Ryabinkin, "Revising the measurement process in the variational quantum eigensolver: is it possible to reduce the number of separately measured operators?" *Chem. Sci.* **10** (2019) 3746–3755.

[62] W. J. Huggins, J. McClean, N. Rubin, Z. Jiang, N. Wiebe, K. B. Whaley, and R. Babbush, "Efficient and noise resilient measurements for quantum chemistry on near-term quantum computers," arXiv:1907.13117 [quant-ph].

[63] A. F. Izmaylov, T.-C. Yen, R. A. Lang, and V. Verteletskyi, "Unitary partitioning approach to the measurement problem in the variational quantum eigensolver method," *Journal of Chemical Theory and Computation* **16** no. 1, (Jan, 2020) 190–195.

[64] V. Verteletskyi, T.-C. Yen, and A. F. Izmaylov, "Measurement optimization in the variational quantum eigensolver using a minimum clique cover," *The Journal of Chemical Physics* **152** no. 12, (2020) 124114.

[65] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Phys. Rev. A* **98** (Sep, 2018) 032309.

[66] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Evaluating analytic gradients on quantum hardware," *Phys. Rev. A* **99** (Mar, 2019) 032331.

[67] G. E. Crooks, "Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition," arXiv:1905.13311 [quant-ph].

[68] A. Harrow and J. Napp, "Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms," arXiv:1901.05374 [quant-ph].

[69] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature* **508** (Apr, 2014) 500.

[70] A. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature Communications* **6** (2015) 6979.

[71] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization* **21** no. 4, (2001) 345–383.

[72] C. Cartis, J. Fiala, B. Marteau, and L. Roberts, "Improving the flexibility and robustness of model-based derivative-free optimization solvers," *ACM Trans. Math. Softw.* **45** no. 3, (Aug., 2019) .

[73] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and Scipy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods* **17** (2020) 261–272.

[74] H. Abraham, AduOffei, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, G. Alexandrowics, E. Arbel, A. Asfaw, C. Azaustre, AzizNgoueya, P. Barkoutsos, G. Barron, L. Bello, Y. Ben-Haim, D. Bevenius, L. S. Bishop, S. Bolos, S. Bosch, S. Bravyi, D. Bucher, A. Burov, F. Cabrera, P. Calpin, L. Capelluto, J. Carballo, G. Carrascal, A. Chen, C.-F. Chen, R. Chen, J. M. Chow, C. Claus, C. Clauss, A. J. Cross, A. W. Cross, S. Cross, J. Cruz-Benito, C. Culver, A. D. Córcoles-Gonzales, S. Dague, T. E. Dandachi, M. Dartiailh, DavideFrr, A. R. Davila, A. Dekusar, D. Ding, J. Doi, E. Drechsler, Drew, E. Dumitrescu, K. Dumon, I. Duran, K. EL-Safty, E. Eastman, P. Eendebak, D. Egger, M. Everitt, P. M. Fernández, A. H. Ferrera, A. Frisch, A. Fuhrer, M. GEORGE, J. Gacon, Gadi, B. G. Gago, C. Gambella, J. M. Gambetta, A. Gammanpila, L. Garcia, S. Garion, A. Gilliam, J. Gomez-Mosquera, S. de la Puente González, J. Gorzinski, I. Gould, D. Greenberg, D. Grinko, W. Guan, J. A. Gunnels, M. Haglund, I. Haide, I. Hamamura, V. Havlicek, J. Hellmers, Ł. Herok, S. Hillmich, H. Horii, C. Howington, S. Hu, W. Hu, H. Imai, T. Imamichi, K. Ishizaki, R. Iten, T. Itoko, JamesSeaward, A. Javadi, A. Javadi-Abhari, Jessica, K. Johns, T. Kachmann, N. Kanazawa, Kang-Bae, A. Karazeev, P. Kassebaum, S. King, Knabberjoe, A. Kovyrshin, R. Krishnakumar, V. Krishnan, K. Krsulich, G. Kus, R. LaRose, R. Lambert, J. Latone, S. Lawrence, D. Liu, P. Liu, Y. Maeng, A. Malyshev, J. Marecek, M. Marques, D. Mathews, A. Matsuo, D. T. McClure, C. McGarry, D. McKay, D. McPherson, S. Meesala, M. Mevissen, A. Mezzacapo, R. Midha, Z. Minev, A. Mitchell, N. Moll, M. D. Mooring, R. Morales, N. Moran, MrF, P. Murali, J. Müggenburg, D. Nadlinger, K. Nakanishi, G. Nannicini, P. Nation, E. Navarro, Y. Naveh, S. W. Neagle, P. Neuweiler, P. Niroula, H. Norlen, L. J. O'Riordan, O. Ogunbayo, P. Ollitrault, S. Oud, D. Padilha, H. Paik, S. Perriello, A. Phan, F. Piro, M. Pistoia, A. Pozas-iKerstjens, V. Prutyanov, D. Puzzuoli, J. Pérez, Quintiii, R. Raymond, R. M.-C. Redondo, M. Reuter, J. Rice, D. M. Rodríguez, RohithKarur, M. Rossmannek, M. Ryu, T. SAPV, SamFerracin, M. Sandberg, H. Sargsyan, N. Sathaye, B. Schmitt, C. Schnabel, Z. Schoenfeld, T. L. Scholten, E. Schoute, J. Schwarm, I. F. Sertage, K. Setia, N. Shammah, Y. Shi, A. Silva, A. Simonetto, N. Singstock, Y. Siraichi, I. Sitdikov, S. Sivarajah, M. B. Sletfjerding, J. A. Smolin, M. Soeken, I. O. Sokolov, SooluThomas, D. Steenken, M. Stypulkoski, J. Suen, S. Sun, K. J. Sung, H. Takahashi, I. Tavernelli, C. Taylor, P. Taylour, S. Thomas, M. Tillet, M. Tod, E. de la Torre, K. Trabing, M. Treinish, TrishaPe, W. Turner, Y. Vaknin, C. R. Valcarce, F. Varchon, A. C. Vazquez, D. Vogt-Lee, C. Vuillot, J. Weaver, R. Wieczorek, J. A. Wildstrom, R. Wille,

E. Winston, J. J. Woehr, S. Woerner, R. Woo, C. J. Wood, R. Wood, S. Wood, S. Wood, J. Wootton, D. Yeralin, R. Young, J. Yu, C. Zachow, L. Zdanski, C. Zoufal, Zoufalc, a matsuo, adekusar drl, azulehner, bcamorrison, brandhsn, chlorophyll zz, dan1pal, dime10, drholmie, elfrocampeador, faisaldebouni, fanizzamarco, gadial, gruu, jliu45, kanejess, klinvill, kurarrr, lerongil, ma5x, merav aharoni, michelle4654, ordmoj, sethmerkel, strickroman, sumitpuri, tigerjack, toural, vvilpas, welien, willhbang, yang.luh, yelojakit, and yotamvakninibm, "Qiskit: An open-source framework for quantum computing," 2019.

[75] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, and M. Head-Gordon, "Simulated quantum computation of molecular energies," *Science* **309** no. 5741, (2005) 1704–1707.

[76] P. J. J. O'Malley, R. Babbush, I. D. Kivlichan, J. Romero, J. R. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, P. V. Coveney, P. J. Love, H. Neven, A. Aspuru-Guzik, and J. M. Martinis, "Scalable quantum simulation of molecular energies," *Phys. Rev. X* **6** (Jul, 2016) 031007.

[77] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature* **549** no. 7671, (2016) 195–202.

[78] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Phys. Rev. E* **58** (Nov, 1998) 5355–5363.

[79] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem," *Science* **292** no. 5516, (2001) 472–475.

[80] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics* **2** (2014) 5.

[81] F. Barahona, "On the computational complexity of Ising spin glass models," *Journal of Physics A: Mathematical and General* **15** no. 10, (Oct, 1982) 3241–3253.

[82] V. Choi, "Minor-embedding in adiabatic quantum computation: I. the parameter setting problem," *Quantum Information Processing* **7** no. 5, (2008) 193–209.

[83] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, "What is the computational value of finite-range tunneling?" *Phys. Rev. X* **6** (Aug, 2016) 031015.

[84] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem," arXiv:1412.6062 [quant-ph].

[85] R. Biswas, Z. Jiang, K. Kechezhi, S. Knysh, S. Mandrà, B. O'Gorman, A. Perdomo-Ortiz, A. Petukhov, J. Realpe-Gómez, E. Rieffel, D. Venturelli, F. Vasko, and Z. Wang, "A NASA perspective on quantum computing: Opportunities and challenges," *Parallel Computing* **64** (2017) 81–98.

[86] D. Wecker, M. B. Hastings, and M. Troyer, "Training a quantum optimizer," *Phys. Rev. A* **94** (Aug, 2016) 022309.

[87] E. Farhi and A. W. Harrow, "Quantum supremacy through the quantum approximate optimization algorithm," arXiv:1602.07674 [quant-ph].

[88] Z. Jiang, E. G. Rieffel, and Z. Wang, "Near-optimal quantum circuit for Grover's unstructured search using a transverse field," *Phys. Rev. A* **95** (Jun, 2017) 062317.

[89] Z. Wang, S. Hadfield, Z. Jiang, and E. G. Rieffel, "Quantum approximate optimization algorithm for MaxCut: A fermionic view," *Phys. Rev. A* **97** (Feb, 2018) 022304.

[90] S. Hadfield, Z. Wang, B. O'Gorman, E. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms* **12** no. 2, (2017) 34.

[91] S. Lloyd, "Quantum approximate optimization is computationally universal," arXiv:1812.11075 [quant-ph].

[92] E. Farhi, J. Goldstone, S. Gutmann, and L. Zhou, "The quantum approximate optimization algorithm and the Sherrington-Kirkpatrick model at infinite size," arXiv:1910.08187 [quant-ph].

[93] M. Willsch, D. Willsch, F. Jin, H. D. Raedt, and K. Michielsen, "Benchmarking the quantum approximate optimization algorithm," arXiv:1907.02359 [quant-ph].

[94] D. M. Abrams, N. Didier, B. R. Johnson, M. P. d. Silva, and C. A. Ryan, "Implementation of the XY interaction family with calibration of a single pulse," arXiv:1912.04424 [quant-ph].

[95] A. Bengtsson, P. Vikstål, C. Warren, M. Svensson, X. Gu, A. F. Kockum, P. Krantz, C. Križan, D. Shiri, I.-M. Svensson, G. Tancredi, G. Johansson, P. Delsing, G. Ferrini, and J. Bylander, "Quantum approximate optimization of the exact-cover problem on a superconducting quantum processor," arXiv:1912.10495 [quant-ph].

[96] X. Qiang, X. Zhou, J. Wang, C. M. Wilkes, T. Loke, S. O'Gara, L. Kling, G. D. Marshall, R. Santagati, T. C. Ralph, J. B. Wang, J. L. O'Brien, M. G. Thompson, and J. C. F. Matthews, "Large-scale silicon quantum photonics implementing arbitrary two-qubit processing," *Nature Photonics* **12** no. 9, (2018) 534–539.

[97] The Cirq Developers, "Cirq: A python framework for creating, editing, and invoking noisy intermediate scale quantum (NISQ) circuits," 2020. https://github.com/quantumlib/Cirq. https://github.com/quantumlib/Cirq.

[98] R. O'Donnell, *Analysis of Boolean Functions*. Cambridge University Press, 2014. www.analysisofbooleanfunctions.net.

[99] C. Y.-Y. Lin and Y. Zhu, "Performance of QAOA on typical instances of constraint satisfaction problems with bounded degree," arXiv:1601.01744 [quant-ph].

[100] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, "Defining and detecting quantum speedup," *Science* **345** no. 6195, (2014) 420–424.

[101] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima, "An efficient conversion of quantum circuits to a linear nearest neighbor architecture," *Quantum Info. Comput.* **11** no. 1, (Jan., 2011) 142–166. https://dl.acm.org/doi/10.5555/2011383.2011393.

[102] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM* **42** no. 6, (Nov., 1995) 1115–1145.

[103] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, "Obstacles to state preparation and variational optimization from symmetry protection," arXiv:1910.08980 [quant-ph].

[104] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, "On the qubit routing problem," arXiv:1902.08091 [quant-ph].

[105] Z.-C. Yang, A. Rahmani, A. Shabani, H. Neven, and C. Chamon, "Optimizing variational quantum algorithms using Pontryagin's minimum principle," *Phys. Rev. X* **7** (May, 2017) 021027.

[106] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature Communications* **9** no. 1, (2018) 4812.

[107] K. Burke, "Viewpoint: Improving electronic structure calculations," *Physics* **9** no. 108, (Sept., 2016) . https://physics.aps.org/articles/v9/108.

[108] D. Wecker, M. B. Hastings, N. Wiebe, B. K. Clark, C. Nayak, and M. Troyer, "Solving strongly correlated electron models on a quantum computer," *Phys. Rev. A* **92** (Dec, 2015) 062318.

[109] M. A. Nielsen, "The fermionic canonical commutation relations and the Jordan-Wigner transform,". http://michaelnielsen.org/blog/archive/notes/fermions_and_jordan_wigner.pdf.

[110] G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme, "Quantum algorithms for fermionic simulations," *Phys. Rev. A* **64** (Jul, 2001) 022319.

[111] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, "Error mitigation extends the computational reach of a noisy quantum processor," *Nature* **567** no. 7749, (2019) 491–495.

[112] S. E. Smart and D. A. Mazziotti, "Quantum-classical hybrid algorithm using an error-mitigating n-representability condition to compute the Mott metal-insulator transition," *Phys. Rev. A* **100** (Aug, 2019) 022517.

[113] R. Sagastizabal, X. Bonet-Monroig, M. Singh, M. A. Rol, C. C. Bultink, X. Fu, C. H. Price, V. P. Ostroukh, N. Muthusubramanian, A. Bruno, M. Beekman, N. Haider, T. E. O'Brien, and L. DiCarlo, "Experimental error mitigation via symmetry verification in a variational quantum eigensolver," *Physical Review A* **100** (Jul, 2019) 010302.

[114] **Simons Collaboration on the Many-Electron Problem** Collaboration, M. Motta, D. M. Ceperley, G. K.-L. Chan, J. A. Gomez, E. Gull, S. Guo, C. A. Jiménez-Hoyos, T. N. Lan, J. Li, F. Ma, A. J. Millis, N. V. Prokof'ev, U. Ray, G. E. Scuseria, S. Sorella, E. M. Stoudenmire, Q. Sun, I. S. Tupitsyn, S. R. White, D. Zgid, and S. Zhang, "Towards the solution of the many-electron problem in real materials: Equation of state of the hydrogen chain with state-of-the-art many-body methods," *Phys. Rev. X* **7** (Sep, 2017) 031059.

[115] P. A. Limacher, P. W. Ayers, P. A. Johnson, S. De Baerdemacker, D. Van Neck, and P. Bultinck, "A new mean-field method suitable for strongly correlated electrons: Computationally facile antisymmetric products of nonorthogonal geminals," *Journal of chemical theory and computation* **9** no. 3, (2013) 1394–1401.

[116] J. Hachmann, W. Cardoen, and G. K.-L. Chan, "Multireference correlation in long molecules with the quadratic scaling density matrix renormalization group," *The Journal of chemical physics* **125** no. 14, (2006) 144101.

[117] R. K. Chaudhuri, K. F. Freed, S. Chattopadhyay, and U. Sinha Mahapatra, "Potential energy curve for isomerization of N_2H_2 and C_2H_4 using the improved virtual orbital multireference Møller-Plesset perturbation theory," *The Journal of Chemical Physics* **128** no. 14, (2008) 144304.

[118] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry*. Dover Publications, Inc., Mineola, New York, 1996. First published by the Macmillan Publishing Company, New York, in 1982.

[119] D. J. Thouless, "Stability conditions and nuclear rotations in the Hartree-Fock theory," *Nuclear Physics* **21** (1960) 225–232.

[120] R. McWeeny, "Some recent advances in density matrix theory," *Rev. Mod. Phys.* **32** (Apr, 1960) 335–369.

[121] A. J. Coleman, "Structure of fermion density matrices," *Rev. Mod. Phys.* **35** (Jul, 1963) 668–686.

[122] W. Kutzelnigg, "Generalized k-particle Brillouin conditions and their use for the construction of correlated electronic wavefunctions," *Chemical Physics Letters* **64** no. 2, (1979) 383–387.

[123] Q. Sun, "Co-iterative augmented Hessian method for orbital optimization," arXiv:1610.08423 [physics.chem-ph].

[124] M. Gluza, M. Kliesch, J. Eisert, and L. Aolita, "Fidelity witnesses for fermionic quantum simulations," *Phys. Rev. Lett.* **120** (May, 2018) 190501.

[125] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Phys. Rev. Lett.* **119** (Nov, 2017) 180509.

[126] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, "An adaptive variational algorithm for exact molecular simulations on a quantum computer," *Nature communications* **10** no. 1, (2019) 1–9.

[127] S. Pironio, A. Acín, S. Massar, A. B. de la Giroday, D. N. Matsukevich, P. Maunz, S. Olmschenk, D. Hayes, L. Luo, T. A. Manning, and C. Monroe, "Random numbers certified by Bell's theorem," *Nature* **464** no. 7291, (Apr, 2010) 1021–1024.

[128] U. Vazirani and T. Vidick, "Certifiable quantum dice," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **370** no. 1971, (2012) 3432–3448.

[129] S. Pironio and S. Massar, "Security of practical private randomness generation," *Phys. Rev. A* **87** (Jan, 2013) 012336.

[130] S. Fehr, R. Gelles, and C. Schaffner, "Security and composability of randomness expansion from Bell inequalities," *Phys. Rev. A* **87** (Jan, 2013) 012335.

[131] M. Coudron, T. Vidick, and H. Yuen, "Robust randomness amplifiers: Upper and lower bounds," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, P. Raghavendra, S. Raskhodnikova, K. Jansen, and J. D. P. Rolim, eds., pp. 468–483. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[132] M. Coudron and H. Yuen, "Infinite randomness expansion with a constant number of devices," in *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, p. 427–436. Association for Computing Machinery, New York, NY, USA, 2014.

[133] C. A. Miller and Y. Shi, "Robust protocols for securely expanding randomness and distributing keys using untrusted quantum devices," *J. ACM* **63** no. 4, (Oct., 2016) .

[134] C. A. Miller and Y. Shi, "Universal security for randomness expansion from the spot-checking protocol," *SIAM Journal on Computing* **46** no. 4, (2017) 1304–1335.

[135] R. Arnon-Friedman, F. Dupuis, O. Fawzi, R. Renner, and T. Vidick, "Practical device-independent quantum cryptography via entropy accumulation," *Nature Communications* **9** no. 1, (Jan, 2018) 459.

[136] R. Arnon-Friedman, R. Renner, and T. Vidick, "Simple and tight device-independent security proofs," *SIAM Journal on Computing* **48** no. 1, (2019) 181–225.

[137] Y. Zhang, H. Fu, and E. Knill, "Efficient randomness certification by quantum probability estimation," *Phys. Rev. Research* **2** (Jan, 2020) 013016.

[138] P. Bierhorst, E. Knill, S. Glancy, Y. Zhang, A. Mink, S. Jordan, A. Rommal, Y.-K. Liu, B. Christensen, S. W. Nam, M. J. Stevens, and L. K. Shalm, "Experimentally generated randomness certified by the impossibility of superluminal signals," *Nature* **556** no. 7700, (Apr, 2018) 223–226.

[139] Y. Liu, Q. Zhao, M.-H. Li, J.-Y. Guan, Y. Zhang, B. Bai, W. Zhang, W.-Z. Liu, C. Wu, X. Yuan, H. Li, W. J. Munro, Z. Wang, L. You, J. Zhang, X. Ma, J. Fan, Q. Zhang, and J.-W. Pan, "Device-independent quantum random-number generation," *Nature* **562** no. 7728, (Oct, 2018) 548–551.

[140] W.-Z. Liu, M.-H. Li, S. Ragy, S.-R. Zhao, B. Bai, Y. Liu, P. J. Brown, J. Zhang, R. Colbeck, J. Fan, Q. Zhang, and J.-W. Pan, "Device-independent randomness expansion against quantum side information," arXiv:1912.11159 [quant-ph].

[141] L. K. Shalm, Y. Zhang, J. C. Bienfang, C. Schlager, M. J. Stevens, M. D. Mazurek, C. Abellán, W. Amaya, M. W. Mitchell, M. A. Alhejji, H. Fu, J. Ornstein, R. P. Mirin, S. W. Nam, and E. Knill, "Device-independent randomness expansion with entangled photons," arXiv:1912.11158 [quant-ph].

[142] Y. Zhang, L. K. Shalm, J. C. Bienfang, M. J. Stevens, M. D. Mazurek, S. W. Nam, C. Abellán, W. Amaya, M. W. Mitchell, H. Fu, C. A. Miller, A. Mink, and E. Knill, "Experimental low-latency device-independent quantum randomness," *Phys. Rev. Lett.* **124** (Jan, 2020) 010505.

[143] J. S. Bell, "On the Einstein Podolsky Rosen paradox," *Physics Physique Fizika* **1** (Nov, 1964) 195–200.

[144] Z. Brakerski, P. Christiano, U. Mahadev, U. Vazirani, and T. Vidick, "A cryptographic test of quantumness and certifiable randomness from a single quantum device," in *2018 IEEE 59th Annual Symposium on Foundations of*

Computer Science (FOCS), pp. 320–331, IEEE. 2018. http://ieee-focs.org/FOCS-2018-Papers/pdfs/59f320.pdf.

[145] O. Regev, "The learning with errors problem (invited survey)," in *2010 IEEE 25th Annual Conference on Computational Complexity*, pp. 191–204. 2010.

[146] S. Aaronson and L. Chen, "Complexity-theoretic foundations of quantum supremacy experiments," in *Proceedings of the 32nd Computational Complexity Conference*, CCC '17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, DEU, 2017.

[147] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Physics* **14** no. 6, (Jun, 2018) 595–600.

[148] A. Bouland, B. Fefferman, C. Nirkhe, and U. Vazirani, "On the complexity and verification of quantum random circuit sampling," *Nature Physics* **15** no. 2, (Feb, 2019) 159–163.

[149] S. P. Vadhan, "Pseudorandomness," *Foundations and Trends® in Theoretical Computer Science* **7** no. 1–3, (2012) 1–336.

[150] C. Huang, F. Zhang, M. Newman, J. Cai, X. Gao, Z. Tian, J. Wu, H. Xu, H. Yu, B. Yuan, M. Szegedy, Y. Shi, and J. Chen, "Classical simulation of quantum supremacy circuits," arXiv:2005.06787 [quant-ph].

[151] J. Gray and S. Kourtis, "Hyper-optimized tensor network contraction," arXiv:2002.01935 [quant-ph].

[152] R. Raz, O. Reingold, and S. Vadhan, "Extracting all the randomness and reducing the error in Trevisan's extractors," *Journal of Computer and System Sciences* **65** no. 1, (2002) 97 – 128.

[153] L. Trevisan, "Construction of extractors using pseudo-random generators (extended abstract)," in *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, p. 141–148. Association for Computing Machinery, New York, NY, USA, 1999.

[154] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980 [cs.LG].

[155] J. Zhang, J. Vala, S. Sastry, and K. B. Whaley, "Geometric theory of nonlocal two-qubit operations," *Phys. Rev. A* **67** (Apr, 2003) 042313.

[156] N. C. Rubin, R. Babbush, and J. McClean, "Application of fermionic marginal constraints to hybrid quantum algorithms," *New Journal of Physics* **20** no. 5, (2018) 053020.

[157] T. Takeshita, N. C. Rubin, Z. Jiang, E. Lee, R. Babbush, and J. R. McClean, "Increasing the representation accuracy of quantum simulations of chemistry without extra quantum resources," *Phys. Rev. X* **10** (Jan, 2020) 011004.