Received: 10 October 2023 | Revised: 25 March 2024 | Accepted: 29 March 2024

DOI: http://doi.org/10.1002/cpe.8121

SPECIAL ISSUE PAPER

Quantum optimization algorithms: energetic implications.

Rolando P. Hong Enriquez*[1] | Rosa M. Badia[2] | Barbara Chapman[3] | Kirk Bresniker[1] | Scott Pakin[4] | Alok Mishra[1] | Pedro Bruel[1] | Aditya Dhakal[1] | Gourav Rattihalli[1] | Ninad Hogade[1] | Eitan Frachtenberg[1] | Dejan Milojicic[1]

[1], Hewlett Packard Labs, Milpitas, CA, USA
[2], Barcelona Supercomputing Center, Barcelona, Spain
[3], Hewlett Packard Enterprise, New York, NY, USA
[4], Los Alamos National Laboratory, Los Alamos, NM, USA

Correspondence
*Rolando P. Hong Enriquez. Hewlett Packard Labs. Milpitas CA, USA Email: rhong@hpe.com

Abstract

Since the dawn of Quantum Computing (QC), theoretical developments like Shor's algorithm proved the conceptual superiority of QC over traditional computing. However, such quantum supremacy claims are difficult to achieve in practice because of the technical challenges of realizing noiseless qubits. In the near future, QC applications will need to rely on noisy quantum devices that offload part of their work to classical devices. One way to achieve this is by using Parameterized Quantum Circuits (PQCs) in optimization or even in machine learning tasks.

The energy requirements of quantum algorithms have not yet been studied extensively. In this article we explore several optimization algorithms using both theoretical insights and numerical experiments to understand their impact on energy consumption. Specifically, we highlight why and how algorithms like Quantum Natural Gradient Descent, Simultaneous Perturbation Stochastic Approximations or Circuit Learning methods, are at least $2\times$ to $4\times$ more energy efficient than their classical counterparts; why Feedback-Based Quantum Optimization is energy-inefficient; and how techniques like Rosalin can improve the energy efficiency of other algorithms by a factor of $\geq 20\times$. Finally, we use the NchooseK high-level programming model to run optimization problems on both gate-based quantum computers and quantum annealers. Empirical data indicate that these optimization problems run faster, have better success rates, and consume less energy on quantum annealers than on their gate-based counterparts.

KEYWORDS:
quantum optimization, circuit learning, shot optimization, error mitigation, quantum annealing, heterogeneous computing, sustainability.

1 | INTRODUCTION

1.1 | Quantum Technologies

After nearly a century of quantum theory and more than 40 years since the foundational papers on the subject by Richard Feynman [1], the race to practical Quantum Computing (QC) is still in full swing [2]. To minimize decoherence and other errors in QC, popular technologies like superconducting transmon qubits [3] rely on extremely low temperatures to maintain and manipulate the quantum state. Sustaining these low temperatures—although

not directly related to the calculations—is energy-intensive. On the other hand, there are also emergent QC technologies with little or no dependency on low temperatures and which are therefore less energy-intensive. Examples of these technologies are photonic quantum computers [4] and computers based on diamond nitrogen vacancy (NV) qubits [5].

Perhaps one of the most challenging QC technologies to implement, both theoretically and pragmatically, is topological QC, which is postulated to make use of the elusive Majorana qubits [6]. Although these computers are mostly theoretical at the moment, the technology to build them is within reach. While topological quantum computers might prove to be disruptive by greatly reducing the need for error correction, this actively researched technology may still depend on low temperatures [7].

With the continuing diversification of the sector it is hard to predict which quantum technologies will form the future QC ecosystem. Similarly to what happened with classical computing, energy efficiency surely will help shape this technological race. In this regard, aside from the energy requirements of the quantum calculations themselves, the energy spent on auxiliary systems (*e.g.*, cooling) must also be considered. Lastly, algorithmic choices can also have an impact on sustainability. However, these analyses are far from simple as many of these algorithms are implemented as hybrid workflows with no clear separation between classical and quantum components. Indeed, the focus of this paper is related to the latter subject. Here we perform an exploration of the energetic advantages of quantum optimization algorithms over classical ones. Specifically, in the next sections we examine the theoretical reasons for the energetic advantages of a group of quantum algorithms while further supporting these insights with numerical experiments. Finally, we also evaluate the performance and energy consumption of quantum algorithms on the two dominant hardware paradigms for quantum computers: Gate-Based QC vs Quantum Annealing.

1.2 | Hybrid Workflows and Quantum Algorithms

The development of quantum error correction codes injected new energy into QC research and development. However, the QC community is still far from producing fault-tolerant quantum computers [8]. If the sequence of gates in a quantum circuit (*i.e.*, its "depth") is anything but short, the propagation of errors soon renders the calculation useless. Acknowledging these challenges, the research community has adopted a pragmatic approach: in the long term, researchers continue to investigate the building of precise and scalable quantum computers; in the short term, they try to solve practical computational problems by mixing classical and quantum computers.

The development of a hybrid workflow environment that includes both classical and quantum computations requires dynamic runtimes that optimize application execution. In this regard, multiple challenges need to be solved. First, there are no standards for developing quantum applications. Second, it is unclear how to decide which parts of the application should run in the quantum system and which in the classical system. At the moment, most solutions running these types of hybrid workflows rely on the creation of *ad hoc* runtime environments close to the quantum hardware. Additionally, users can be given access to the runtime environments through some form of cloud technology like in the case of IBM's quantum Qiskit runtime environment [9]. In this contribution we abstract ourselves from the details of the runtime and execution environments. While those features are important, they are rapidly evolving. Here we prefer to concentrate on theoretically-based features of quantum algorithms which are less likely to be fundamentally affected by the nuances of the runtime environments.

The rest of the paper is organized in the following manner: Section 2 focuses on the ideas behind the estimation of energy consumption during elementary quantum operations with emphasis on superconducting qubit devices. Section 3 illustrates the theoretical basis for the energy efficiency of several quantum optimization algorithms and performs numerical experiments to corroborate and quantify these gains. Finally, in Section 4 we perform a deeper evaluation of quantum optimization in both gate-based devices and quantum annealers. Sections for discussion and conclusions wrap up the article.

2 | ENERGY CONSUMED IN QUANTUM OPERATIONS

Different quantum technologies necessarily differ in their power requirements, if only because of the variety of physical principles on which they are based. A full-stack energy model for superconducting qubits has recently been described [10]. A practical implication of this model is the possibility to correlate the power consumption to the *shape* of the circuit that runs on a QC. Here, *shape* is the rectangle formed by the number of qubits (N_{qubits}) and the depth of the circuit (N_{depth}). The area of the shape $A_{shape} = N_{qubits} \times N_{depth}$ is a proxy for the quantum memory used in the calculation. After estimating the energy consumption per gate the power consumption of the full circuit also can be estimated [10].

To estimate energy consumption per gate, we leverage the work of Jaschke and Montangero [11], which recently discussed the green quantum advantage of a few algorithms and hardware platforms. From their work we can derive the energy consumed by different quantum gate technologies. The starting point is the empirical equation for the energy consumed by a quantum circuit:

$$E_{circuit} = \left(N_{gates} \times r \times P_{system}\right) / \omega_{gate}, \tag{1}$$

Quantum technology	Gate operation energy (J)
Rydberg atoms	$\approx 15 \times 10^3$
Trapped ions	≈ 15.0
Superconductors	≈ 0.18

Table 1 Approximate energy consumed during the operation of generic quantum gates using different quantum computing technologies. hardware [11]

Device	Runtime (s)	Power/circuit (W)	Power/gate (W)
`ibmq_qasm_simulator`	0.554	6.823	0.569
`simulator_mps`	1.230	3.073	0.256
`simulator_statevector`	0.859	4.400	0.367
`ibmq_lima`	4.441	0.851	0.071
`ibmq_belem`	4.090	0.924	0.077
`ibmq_quito`	4.610	0.820	0.068

Table 2 Power Consumption per gate for a Quantum Circuit shown in Figure 1 on various IBM Simulators and Devices

where N_{gates} is the number of gates in the circuit, r is the number of circuit repetitions needed to achieve the required fidelity, P_{system} is the total system power during circuit execution including cooling, and ω_{gate} is the estimated gate application frequency. We plugged $N_{gates} = 1$, $r = 1000$, and gate frequencies (ω_{gate}) for different quantum technologies into Equation 1 to estimates for the energy consumed by Rydberg atoms and superconducting technologies, and we used a different estimate from Jaschke and Montangero [11] to calculate the equivalent values for the trapped-ion technology. Our estimates for the energy consumed by a quantum gate using these technologies is shown in Table 1.

With these per-gate energy values we can estimate the total energy consumed by a quantum circuit using superconducting technology. Specifically, given the energy consumption per gate E_{gate}, the number of qubits N_{qubits}, and the depth N_{depth} of a quantum circuit, the total energy consumed by a circuit E_{total} is thus

$$E_{total} = E_{gate} \times N_{qubits} \times N_{depth}. \tag{2}$$

We illustrate the convenience of these equations by using them in the small quantum circuit shown in Figure 1. This circuit has four parameters equally distributed in two parameterized layers. The circuit operates on three qubits, has a total of twelve gates, and has a depth of seven. Assuming an energy consumption per gate of ≈ 0.18J for a superconducting quantum computing system (Table 1), and using Equation 2, the total energy consumption of the circuit in Figure 1 is ≈ 3.78J. We conducted experiments by running this circuit on various IBM superconducting quantum simulators and devices and recorded their respective runtimes, as presented in Table 2. The total power consumption per circuit P_{circ} is computed as $P_{circ} = E_{total}/T_{run}$, where T_{run} is the total runtime on each device, and the power consumption per gate P_{gate} is computed as $P_{gate} = P_{circ}/N_{gates}$, where N_{gates} is the number of gates on the circuit. The results of these computations are also shown in Table 2.

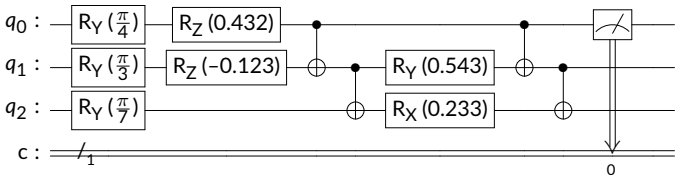

Figure 1 Parameterized quantum circuit (PQC) used in the experiments

We observe in Table 2 that real quantum devices (`ibmq_lima`, `ibmq_belem`, and `ibmq_quito`) have longer runtimes for quantum circuits than simulators (`ibmq_qasm_simulator`, `simulator_mps`, and `simulator_statevector`). This could be due to a variety of factors, including noise, poor

connectivity, and a scarcity of resources. Environmental factors such as temperature fluctuations, magnetic fields, and vibrations cause noise and errors in real quantum devices. These errors can cause qubits to lose their quantum state, resulting in inaccurate measurements. Simulators, on the other hand, lack these environmental factors and can simulate ideal quantum circuits with perfect qubits.

Furthermore, connectivity between qubits in real quantum devices is limited by the physical constraints of the device, which means that operations between qubits that are not directly connected may require additional steps that can increase the circuit's runtime. Simulators often assume complete connectivity between qubits and can perform operations on any pair of qubits with no extra overhead. Real quantum devices have limited resources, such as the number of qubits and operations that can be performed in a single run. This means that more complex circuits may need to be broken down into smaller parts, which can lengthen the circuit's overall runtime. We acknowledge that while simulators can provide a faster and more idealized simulation of quantum circuits, real devices provide a more accurate representation of the challenges and limitations of current quantum technology.

As illustrated in this section, some progress has been made regarding the characterization of energy consumption on quantum computers. However, there is still no clear consensus on the metrics that should be use to report energy efficiency in these calculations. For classic computers, performance is usually reported in floating point operations per second (FLOPs), and the consumed power is reported in watts. Combining these two concepts, energy efficiency can be readily expressed in FLOPs per watt. In quantum computing, metrics of overall system quality such as quantum volume [12,13,14] and algorithmic qubits [15] have industry support from IBM and IonQ, respectively, although neither metric is universally accepted. In this work, in contrast, we take as our main metric the overall energy consumption during a quantum calculation expressed in joules (J).

3 | QUANTUM OPTIMIZATION

One way to capitalize on the strengths of heterogeneous hardware environments is to reformulate problems of interest as a variational optimization problem [16]. The approach is to offload all but the most compute-heavy subroutine to a classical computer. This lowers the requirements for the quantum calculations (*e.g.*, circuit depth) and consequently also reduces the error rate, energy consumption, and overall cost for using quantum devices. Ideally, one should only execute on the quantum computer those subroutines that are intractable or hard for classical computers. In this context, the reformulation of quantum optimization as a variational problem usually implies that the classical computer applies some form of preprocessing to parameterize a quantum circuit. The quantum computer then executes the quantum circuit and performs measurements intended to estimate the expected values of calculations that can, for instance, be encoded as a cost function. These measurements are post-processed and passed to classical optimization or machine-learning (ML) algorithms. Either way, parameters are updated on every iteration to minimize a cost function [17]. As we discuss in the next few sections, working with parameterized quantum circuits (PQCs) enables additional performance improvements on optimization problems as they are a core element in variational approaches.

The following sections target specific quantum optimization algorithms and strategies. Although not a complete list by any means, these algorithms have been selected to reflect the growing variety of ways in which quantum optimization algorithms can operate. For each algorithm we have included sections for theory and numerical experiments. Our take on the theoretical background specifically highlights those elements that affect energy consumption during the execution of these algorithms. The numerical experiments were chosen to display prototypical use cases of these algorithms and a comparison with their classical counterparts where appropriate. In most cases these experiments were performed using the code repositories and runtime environments from PennyLane [18] and Qiskit [9].

3.1 | Quantum Natural Gradient (QN-GD)

Theory

Vanilla Gradient Descent (V-GD) aims to minimize a cost as a function of parameters in Euclidean space. However, Euclidean geometry might be suboptimal [19] so, in practice, different parameterizations of the cost might require different learning rates or step sizes. The shortcomings of V-GD can be addressed by multiplying the Euclidean gradient with the Fisher information matrix (F). This procedure transforms V-GD from an optimization in Euclidean space to an optimization in the space of the probability distributions (*i.e.*, the probability distribution of outputs generated by a certain input). The resulting optimization is known as natural gradient descent [19]. Working with PQCs has several implications (*e.g.*, quantum states in Hilbert spaces). Here, the assumption of Euclidean geometry in the parameter space is clearly inadequate [20]. As in the classical case, we can fix this by multiplying the Euclidean gradient, this time with the Fubini-Study metric tensor (g^+), which nicely reduces to F in the classical case [21]. This generalization is known as quantum natural gradient descent (QN-GD) [22].

Besides increasing the chances of finding the true minima of the system independently of the parameterization used, the introduction of QN-GD has implications related to the number of quantum circuit executions and consequently to the energy spent in these calculations. We first briefly examine how V-GD operates classically and then introduce the differences with its quantum variant.

V-GD and several other classic optimizations that currently are used in quantum calculations use *parameter-shift rules* to evaluate the partial derivatives (*i.e.*, the gradient). Full derivation of these rules can be found in Schuld et al.[23]. For simplicity we show only an abstract general framework illustrating the points related to circuit evaluation.

For a single unitary gate $U(\theta_i)$ depending on the parameter θ_i, a simplified circuit quantum function that aims to estimate the expected value of a Hamiltonian H based on the measurement of the random hermitian observable B can be written as

$$H = \langle\psi|U^\dagger(\theta_i)BU(\theta_i)|\psi\rangle$$

Significant work on deriving parameter-shift rules goes first into expressing the above unitary conjugation as a linear transformation $\mathcal{T}$ acting over B and differentiable in θ_i:

$$U^\dagger(\theta_i)BU(\theta_i) = \mathcal{T}_{\theta_i}(B)$$
$$\nabla_{\theta_i}H = \langle\psi|\nabla_{\theta_i}\mathcal{T}_{\theta_i}(B)|\psi\rangle$$

Lastly, some effort has to be expended on expressing this gradient as a linear combination of the same transform $\mathcal{T}$ but for two different values for the parameter (*e.g.*, θ_{i+s}, θ_{i-s}):

$$\nabla_{\theta_i}\mathcal{T}_{\theta_i}(B) = c[\mathcal{T}_{\theta_{i+s}}(B) - \mathcal{T}_{\theta_{i-s}}(B)]$$

where c is a generic multiplier, the shift s depends on the transformation and does not need to be infinitesimal. Ultimately, that last equation implies that only two quantum circuit evaluations on the shifted parameters are needed to determine this gradient. Note that this is a simplified case; other parameter-shift rules might require additional circuit evaluations[24].

QN-GD differs from V-GD mainly in the evaluation of the Fubini-Study metric tensor g^+, which has a set of interesting properties with applications in pure mathematics and quantum physics. Regarding quantum optimization, although the use of g^+ is theoretically justified, this tensor cannot directly be evaluated on quantum hardware, and therefore its implementation relies on several approximations. For example, the PennyLane library[18] implements the block-diagonal approximation to estimate the Fubini-Study metric tensor[22].

Formally, we start from a parameterized quantum circuit $U(\theta)$, with the parameters $(\theta_1, \theta_2, \ldots, \theta_d)$ distributed in L circuit layers. On each layer we can have non-parameterized gates N_l and parameterized gates $P_l(\theta_l)$ with $\theta_l = \{\theta_1^{(l)}, \theta_2^{(l)}, \ldots, \theta_n^{(l)}\}$, with n_l parameters. It can be proven that a block diagonal approximation of the Fubini-Study tensor has the form,

$$\begin{array}{c} \begin{matrix} \theta_1 & \theta_2 & \cdots & \theta_L \end{matrix} \\ \begin{pmatrix} G^{(1)} & 0 & \cdots & 0 \\ 0 & G^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & G^{(L)} \end{pmatrix} \end{array} \begin{matrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_L \end{matrix}$$

Here G^l is a sub-matrix for layer l with dimensions $n_l \times n_l$. G^l is a covariance matrix built using the observables measured after the evaluation of all the previous layers. (For details see Stokes et al.[22].) This approximation implies that the number of circuit evaluations for QN-GD is $N_{eval} = 2 \times d + L$. For V-GD this number is simply $N_{eval} = 2 \times d$.

Numerical experiments

We ran numerical experiments to compare the optimization convergence of V-GD and QN-GD for two systems. The first simulates a single-qubit circuit, and the second represents a generic PQC from Figure 1. As expected, the single-qubit calculation converged faster. For QN-GD we reached convergence with $\approx$ 250 circuit evaluations for a single qubit and $\approx$ 500 circuit evaluations for the PQC (Figure 2). On the other hand, QN-GD was decisively superior to V-GD. Specifically, with respect to QN-GD, V-GD used 3.2$\times$ more circuit evaluations to optimize a single qubit and 3$\times$ more circuit evaluations to optimize the PQC from Figure 1.

In quantum computing lingo, a *shot* is a single execution of a quantum algorithm or circuit on a quantum device. To estimate the energy-efficiency of QN-GD over V-GD we assume that (a) to obtain the expected value of the cost function we would need to run a quantum circuit $N_{shots} = 1000$ times for every point in Figure 2, (b) the number of steps saved would be $N_{saved} = 550$ for the single qubit and $N_{saved} = 1000$ for the PQC, and (c) the energy and power to run these circuits only once can be used directly or derived from Tables 1 and 2. Then, to estimate the energy savings, we simply multiply $N_{shots} \times N_{saved} \times E_{total}$, giving us $\approx 3.7 \times 10^3$J for the PQC and $\approx$ 100J for a single qubit.

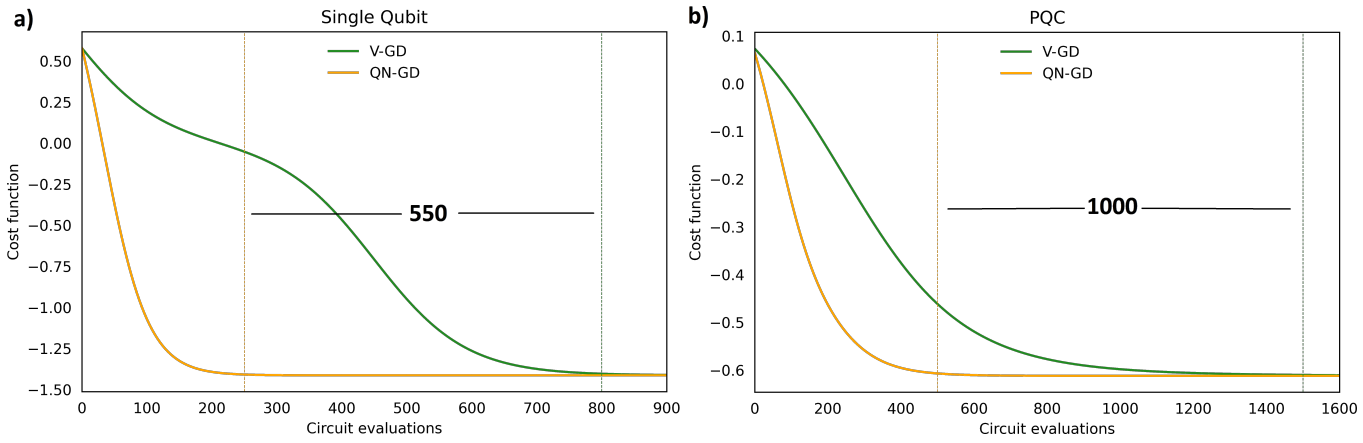

Figure 2 Comparing optimization convergence rates between V-GD and QN-GD for (a) a single qubit and (b) the parameterized quantum circuit (PQC) from Figure 1.

3.2 | Simultaneous Perturbation Stochastic Approximation (SPSA)

Theory

The parameter-shift rules[23] require two circuit evaluations around the selected parameter $\theta_d^{1,2} = \theta_d^* \pm s$ (using the shift $s = \frac{\pi}{4}$ for rotational gates Rx, Ry, and Rz). From those, the partial derivatives (and gradients) used in several quantum optimization algorithms can be obtained directly. Exact gradients are not strictly necessary for minimizing a cost function. Gradient approximations can be used as well, and this is the strategy of the methods described in this section. An approximate gradient can be obtained using a stochastic factor δ instead of a fixed analytical shift s, which can have practical advantages. In an optimization that relies on a parameter vector θ of size p, there are two basic ways to proceed: (1) The stochastic factors can be included in the parameter vector one element at the time. Each update is then immediately followed by an evaluation of the cost function. This path leads to the Kiefer-Wolfowitz finite difference stochastic approximation (FDSA) method[25]; (2) The stochastic factors can be included in all the elements of the parameter vector at the same time, and the evaluation of the cost function can be performed just twice at the end. This path leads to the simultaneous perturbation stochastic approximation (SPSA) method[26]. Therefore for FDSA, the number of cost function evaluations (circuit executions) scales linearly with p while for SPSA, independently of the size of the parameter vector θ, the circuit is evaluated only twice. Besides this difference, the update rule for these algorithms is similar to the classical vanilla gradient descent (V-GD):

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k)$$

where $\hat{g}_k$ is the approximate gradient estimated with parameters $\hat{\theta}_k$ and $a_k > 0$ is the learning rate.

The approximate gradient can be found with this expression:

$$\hat{g}_{ki}(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k\Delta_k) - y(\hat{\theta}_k - c_k\Delta_k)}{2c_k\Delta_{ki}}$$

Here, $\Delta_k = (\Delta_{k1}, \Delta_{k2}, ..., \Delta_{kp})^T$ is a p-dimensional random perturbation vector. It has been shown that if Δ_k is chosen appropriately, the simultaneous perturbation is just as effective for optimization as the FDSA approach and can be performed at a fraction of its computational cost. As mentioned in Section 3.1, the introduction of the Fubini-Study metric tensor (g_{ij}) improves upon V-GD by generalizing the restrictions imposed by an optimization in the Euclidean space of parameters. This generalization increases the probability of finding global minima but is not scalable. g_{ij} is a $p \times p$ tensor, and therefore its calculation turns into a computational burden when performing complex quantum circuits with a large number of parameterized gates. To alleviate these problems, quantum natural SPSA (QN-SPSA) merges the innovations from both SPSA and QN-GD by making stochastic approximations of both the gradient and the Fubini-Study metric tensor[27].

Numerical experiments

The theory section above indicates that the number of parameters to optimize is an important part of this method, but the circuits used here are slightly larger than in Section 3.1. We use a construction template from the PennyLane library called `qml.StronglyEntanglingLayers`, based on the work of Schuld et al.[28]. In the simplest case, by selecting a circuit that should run with $N_{qubits} = 4$ and $N_{layers} = 5$ we create a circuit with $d = N_{qubits} \times N_{layers} \times 3 = 60$ trainable parameters (rotational gates), and $N_{qubits} \times N_{layers} = 20$ non-trainable parameters (CNOT gates).

The number of gradient or circuit evaluations (N_{eval}) for V-GD and SPSA differs noticeably. For V-GD it depends on the number of parameters d, rendering $N_{eval} = 2 \times d \times N_{steps} = 120 \times N_{steps}$ while for SPSA this number is $N_{eval} = 2 \times N_{steps}$, independent of the number of parameters to optimize.

The different convergence rates between V-GD and SPSA as a function of the number of circuit executions or evaluations N_{eval} is shown in Figure 3. For this case, the scaling difference considering the number of trainable parameters d, is certainly pronounced (120×). This advantage

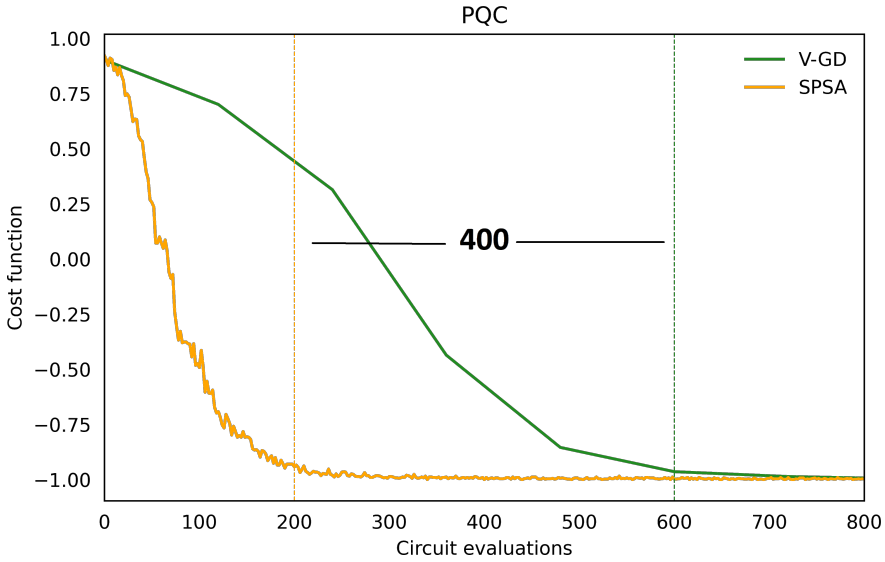

Figure 3 Comparing optimization convergence rates between V-GD, and SPSA.

drops to only 3× when considering the ability to arrive at the same minimized value of the cost function. These gains can be expected to become more important as d increases. We can now estimate the energy savings from using SPSA in the circuit used in this section. Once again we consider $N_{shots} = 1000$ for every point in Figure 3. The number of saved steps from the figure is $N_{saved} = 400$. The circuit used in this section has $N_{gates} = 80$, which we will assume to have identical energy consumption at any position in the circuit and with energy consumption values reported in Tables 1 and 2 for superconducting qubits. Given these parameters, the energy that can be saved by using SPSA instead of V-GD to optimize the circuit is ≈ 5.7 kJ.

3.3 | Quantum circuit structure learning (Rotosolve and Rotoselect)

Theory

PQC optimizers often take advantage of parameter-shift rules[23] to evaluate phase-shifted expectation values of the circuit and return the gradient, which is used to minimize the cost function (encoded as a Hamiltonian). Two circuit evaluations per optimization step are needed to accomplish this. The methods in this section use a similar phase-shifted strategy but avoid the gradient calculation. Both Rotosolve and Rotoselect have a similar theoretical justification[29]. Starting from parameterized gates of the form $U_d = exp(-i(\theta_d/2)H_d)$, where $\theta_d \in (-\pi, \pi]$ and H_d is a Hermitian unitary operator (*i.e.*, we are using rotation gates Rx, Ry, and Rz). Then, the expected value of the Hamiltonian as a function of the selected gate, given that all other parameters and gates are fixed, has a sinusoidal form: $\langle M \rangle_{\theta_d} = A \sin(\theta_d + B) + C$. The sinusoidal function can be characterized by estimating the values of A, B and C, achieved by sampling the expected values of the Hamiltonian at specific gate angles. A closed-form expression for the optimal angle (the one that minimized the Hamiltonian) is

$$\theta_d^* = \theta - \frac{\pi}{2} - \arctan\left(\frac{2\langle \mathrm{M} \rangle_\theta - \langle \mathrm{M} \rangle_{\theta+\frac{\pi}{2}} - \langle \mathrm{M} \rangle_{\theta-\frac{\pi}{2}}}{\langle \mathrm{M} \rangle_{\theta+\frac{\pi}{2}} - \langle \mathrm{M} \rangle_{\theta-\frac{\pi}{2}}}\right) + 2k\pi \ .$$

This expression implies that for a selected gate one can calculate the optimal value analytically (gradient-free) using three circuit evaluations. Furthermore, since the values for any of the rotation gates for $\theta_d = 0$ are identical ($R_x(0) = R_y(0) = R_z(0) = 1$), then for circuits with a number of parameterized gates equal to 1, 2, and 3, one will need 3, 5, and 7 circuit evaluations respectively. The difference between Rotosolve and Rotoselect lies only in the scope of the optimization cycle. With fixed gates and restricting the optimization to the parameters, the resulting algorithm is Rotosolve. If every optimization step includes the type of gate (Rx, Ry, or Rz), then the resulting algorithm is Rotoselect. Either way, within each

optimization cycle a greedy approach is followed: for each generated or fixed gate and every parameter the optimized values are calculated while leaving all the other parameters and gates fixed. The process continues until a stopping criterion is met.

Numerical experiments

We performed numerical experiments to compare Rotosolve and Rotoselect with V-GD. The algorithms were tested on a toy circuit with a single rotational gate applied to each (N_{qubits} = 2, N_{depth} = 1, $N_{parameters}$ = 2). Figure 4 displays the optimization of the corresponding Hamiltonian as a function of circuit evaluations (N_{eval}).

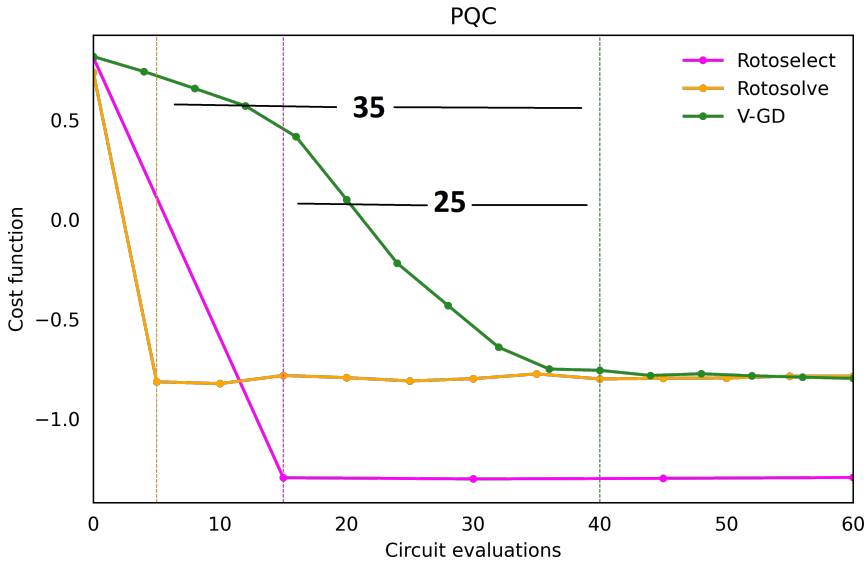

Figure 4 Comparing optimization convergence rates between V-GD, Rotosolve and Rotoselect. We used a simple circuit with two parameterized gates running on 2 qubits.

As in previous sections, we use parameter-shift rules [23] for V-GD. That is, we performed two circuit evaluations per circuit. V-GD for this small circuit reaches a minimum after ten optimization steps. In the cases of Rotosolve and Rotoselect, the minimum is reached in a single step. For Rotosolve, one step takes seven circuit evaluations, and the cost function finds the same value as V-GD. On the other hand, Rotoselect finds a lower value in the cost function by additionally optimizing the type of gates to be used (circuit structure learning). This indicates that both V-GD and Rotosolve were stuck in local minima.

Despite both Rotosolve and Rotoselect making additional circuit evaluations per optimization step, both algorithms found equal or better solutions than V-GD with less than half the overall number of circuit evaluations. In this example, Rotosolve and Rotoselect observe an advantage of $\geq 2.6\times$ with respect to V-GD. The circuit used in this section is fairly small so the energy savings in absolute value is negligible but is reported for completeness. Assuming as usual N_{shots} = 1000, the steps saved from using Rotosolve and Rotoselect are respectively 35 and 25. The energy savings with respect to V-GD are $\approx$ 12J and $\approx$ 9J, for Rotosolve and Rotoselect, respectively.

3.4 | Frugal shot optimization (Rosalin)

Theory

The number or shots or single executions of a quantum circuit has become a *de facto* metric for pricing the rental of quantum devices, and there are at least two powerful reasons why this metric is likely to stay. First, current quantum devices are noisy, which means that to create the illusion of a logical qubit, the information must be encoded into several physical qubits using sophisticated quantum error correction techniques [30]. Depending on the technology, the number of physical qubits required to accomplish this varies significantly. A promising solution to this problem could be the implementation of the elusive Majorana qubit for topological quantum computing [6]. However, even if a noiseless quantum computing is built, the second reason for having multiple shots for any calculation is the nondeterministic nature of quantum computing.

To the best of our knowledge, there is no general formula or solution to estimate the number of shots required to run a quantum algorithm to a certain accuracy. Nevertheless, in this section we describe a method that can be use to optimize the number of shots in some optimization problems. The method is called Random Operator Sampling for Adaptive Learning with Individual Number of Shots (Rosalin)[31].

Rosalin's approach to minimize the number of shots starts with a cost function that can be expressed as the expected value of a Hamiltonian $C = \langle H \rangle$. The underlying assumption is that this Hamiltonian can be expressed as a weighted sum of measurable and generally non-commuting operators $\{h_i\}$:

$$H = \sum_{n=1}^{N} c_i h_i$$

In the sense of quantum mechanics, non-commuting operators represent variables that cannot be measured simultaneously, such as the classic example of position and momentum in Heisenberg's uncertainly principle. The practical implication is that the expected value of the Hamiltonian can be calculated from the expected values of the individual operators or variables $\langle h_i \rangle$ and that a different number of shots might be needed (and optimized) for each of them. Rosalin's goal thus can be framed as follows: given a budget of shots, how can they be distributed over the variables (*i.e.*, by sampling h_i) for a maximum impact of every shot while providing an unbiased estimation of $\langle H \rangle$? As described briefly below, Rosalin combines a sampling redistribution strategy with the core idea of an adaptive optimizer named individual Coupled Adaptive Number of Shots (iCANS)[32].

There are several ways in which shots could be redistributed to sample the variables $\langle h_i \rangle$ although not all of them will lead to an unbiased estimation of $\langle H \rangle$ and its variance. Arrasmith et al. describe these strategies in detail[31], but here we simply list them: (1) *uniform deterministic sampling*, where the shots are equally distributed among the N variables h_i; (2) *weighted deterministic sampling*, where the shots are distributed in proportion to the coefficients c_i of the variables; (3) *weighted random sampling (WRS)*, where a variable h_i is selected for sampling with probability

$$p_i = \frac{|c_i|}{\sum_{n=1}^{N} |c_i|}$$

and (4) *weighted hybrid sampling (WHS)*, a combination of (2) and (3). It is important to note that the first two methods are deterministic while the last two have at least some stochastic elements. To have any hope of obtaining unbiased estimators of $\langle H \rangle$ and its variance, one should avoid using the deterministic methods by themselves. The introduction of randomness is the factor that ultimately enables unbiased estimators with as little as a single shot in Rosalin. In fact, Arrasmith et al.[31] use the last two strategies, naming the resulting algorithms Rosalin1 and Rosalin2, respectively.

Some additional background must be presented to explain the other main component of Rosalin, which is essentially a slight modification of iCANS. The starting point is that shot frugality comes from the study of continuity in smooth uniform functions. These characteristics are related to the learning rate (α) used in most optimization algorithms, most notably gradient descent. Generally speaking, if α is selected to be small, convergence, at least to a local minimum, is practically guaranteed at the cost of having to perform a large number of optimization steps. Ideally, it would be convenient to have an analytical expression for the open bound of the gradient as this would help select a proper learning rate. Kübler et al.[32] use a strong form of function continuity named Lipschitz continuity to explore these issues. By definition, a function is Lipschitz-continuous if there is an L (Lipschitz constant) that fulfills the condition

$$\|\nabla f(\theta_{t+1} - \nabla f(\theta_t))\| \leqslant L\|\theta_{t+1} - \theta_t\|$$

for all θ_{t+1} and θ_t, and where $\|.\|$ is the l_2 Euclidean norm. It can be proven that gradient descent converges as long as $\alpha \leqslant \frac{2}{L}$[33], even though L is an unknown property of the cost function and cannot generally be exploited in machine learning. For cost functions encoded in the type of Hamiltonians described in this section one can estimate an upper bound for L as

$$L < \sum_{n=1}^{N} |c_i|$$

Analytical solutions also can be found (and L can be accessed) if the variables of the Hamiltonian are, for instance, single-qubit rotation gates. In Rosalin, after a first sampling pass using a small number of shots for each operator h_i, the usual optimization routine is followed—estimation of the gradient g_i and its variance v_i using the classic updating rules to perform a step of gradient descent. But an innovation is introduced at this point: the estimation of the expected gain per shot (improvement in $\langle H \rangle$) for every single operator h_i (or parameter θ_i). This improvement takes the form

$$\gamma_i = \frac{1}{s_i}\left[\left(\alpha - \frac{L\alpha^2}{2}\right) g_i^2 - \frac{L\alpha^2}{2s_i} v_i\right] .$$

For the next iteration, one can calculate a new number of shots s_i for each operator h_i as the number of shots that maximizes γ_i:

$$s_i = \left(\frac{2L\alpha}{2 - L\alpha}\right) \frac{v_i}{g_i^2} .$$

Aside from these core features, the algorithm is fine-tuned with additional heuristics as well as hyper-parameters for convergence smoothness and regularization.

Numerical experiments

Using the Pauli matrices X, Y, and Z as generators for operators h_i, a valid Hamiltonian could be

$$H = 3I \otimes X + 6I \otimes Y + I \otimes Z$$

with coefficients $c = [3, 6, 1]$. Following the theory explained above, it is possible to recreate sampling strategies for the number of shots on each gate. Despite this being a toy example, the same analysis applies to complex Hamiltonians with a large number of operators h_i.

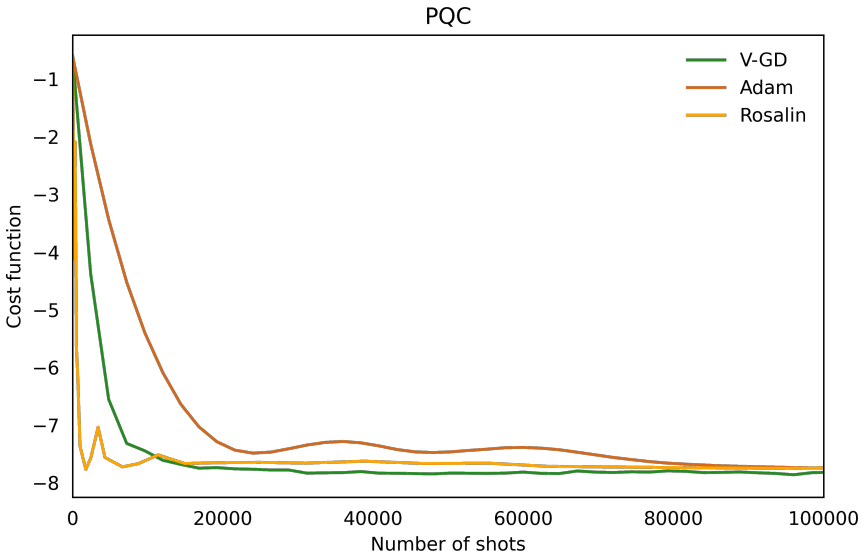

Figure 5 Comparing optimization convergence rates between V-GD, the Adam optimizer and Rosalin.

We performed numerical experiments using a small Hamiltonian explicitly expressed using five non-commuting operators, each with a coefficient: $c_i h_i$. The resulting circuit has two qubits, and the calculations within the circuit are distributed across two layers. Following the theory described above for Rosalin, the cost function built from the Hamiltonian requires a minimal number of shots $\text{Min}_{\text{shots}}$ to determine the expected value for each operator $\langle h_i \rangle$, then monitors how fast the number of shots s_i recommended for each operator h_i changes and adapts accordingly the recommendation for the next optimization step. At each optimization step k, the expected value of the global Hamiltonian is calculated as usual; $\langle H \rangle_k = \sum c_i \langle h_i \rangle_k$.

Under equivalent conditions and using the parameter-shift rule[23] to estimate the parameter gradients, the number of shots needed for Rosalin to estimate the gradient is at least 20 times shorter than the corresponding number of shots for the Adam optimizer or V-GD. For our small Hamiltonian, this leads to an energy savings of only 18J. Due to the reduced number of shots required, Rosalin's advantage should be more than a factor of 20 for larger circuits, and the total energy savings also should scale up quickly—although here we do not make any strong claims regarding scalability beyond what seems reasonable to assume given the theory behind these methods.

3.5 | Combinatorial optimization (FALQON/QAOA)

Theory

This section describes two optimization algorithms: the quantum approximate optimization algorithm (QAOA)[34] and the Feedback-based ALgorithm for Quantum OptimizatioN (FALQON)[35]. These algorithms have similar applications and theoretical backgrounds. Both are based on the relationship between time-dependent Hamiltonians and quantum circuits. The motivation for this relationship comes from both directions, theory and implementation, and appears complementary. On one side, all but the simplest quantum systems evolve in time and are therefore dynamic; here we might be interested in mapping physical reality to quantum calculations. On the other hand, to solve certain optimization problems, it might be useful to think of a quantum circuit as a time-dependent physical Hamiltonian.

The mapping of a complex Hamiltonian into a reliable quantum circuit remains an art more than a science. However, significant progress has been made to characterize the building blocks of these mappings. The success of these constructs relies in understanding that the time evolution is the link that supports the generation of quantum circuits from Hamiltonians. Plainly stated, even a circuit gate can be seen as an implementation

of time evolution under a well-crafted Hamiltonian. That is, a gate is a transformation of the input state described by a unitary time evolution operator that depends explicitly on the Hamiltonian (H) and a scalar time t (although t can be substituted by a generic scalar γ_j):

$$U(H,t) = e^{-iHt/\hbar} = e^{-i\gamma_j H} .$$

In principle, using these newly defined gates to build larger circuits for generic Hamiltonians with many non-commuting terms should be a simple task:

$$H = H_1 + H_2 + H_3 +, ..., + H_N .$$

Alas, this construction is not so simple in practice. Circuit generation is an NP-hard problem because of the exponential growth of the Hilbert space with the size of $U(H,t)$ [36]. Additionally, current quantum computers are noisy [37] so it is desirable to keep the circuit depth as shallow as possible. Due to these limitations, circuit generation relies on approximate time-evolution operators that can be derived using the Trotter-Suzuki (or Lie) formulas [38]. In its simplest form, this *Trotterization* mechanism states that for $m \times m$ real or complex matrices A and B,

$$e^{A+B} = \lim_{n\to\infty} (e^{A/n} e^{B/n})^n .$$

The approximation to the exact Hamiltonian then can be written as

$$U(H,t,n) = \prod_{j=1}^{n} \prod_{k} e^{-i\gamma_j H/n}$$

where $H = \sum_k H_k$ and U approaches the exact solution $e^{-i\gamma_j H}$ as n grows to N.

One last piece of theory needed for these algorithms comes from quantum Lyapunov control methods [39], which aim to identify ways to control the dynamics of a quantum system using feedback loops. To use this framework, we can imagine encoding the solution to our optimization in a cost Hamiltonian H_c and "drive" its expected value to a minimum using another Hamiltonian H_d. That is $H = H_c + \beta(t) H_d$, where the time-dependent term $\beta(t)$ is a control equation that tries to capture the "driving/drifting" strategy. An ideal procedure minimizes monotonically the expected value of the cost Hamiltonian $\langle H_c \rangle = \langle \psi(t)|H_c|\psi(t)\rangle$ by choosing a $\beta(t)$ such as

$$\frac{d}{dt}\langle \psi(t)|H_c|\psi(t)\rangle \leq 0, \forall t .$$

There is ample flexibility to choose $\beta(t)$ to satisfy these constraints. An iterative approach is by selecting

$$\begin{aligned} \frac{d}{dt}\langle \psi(t)|H_c|\psi(t)\rangle &= A(t)\beta(t) \\ A(t) &= \langle i[H_c, H_d]\rangle_t \\ \beta(t) &= -A(t) \end{aligned}$$

which results, by design, in a strictly decreasing minimization of the cost Hamiltonian:

$$\frac{d}{dt}\langle \psi(t)|H_c|\psi(t)\rangle = -|\langle i[H_c, H_d]\rangle_t|^2 \leq 0 .$$

It can be shown that a *Trotterization* of these $[H_c, H_d]$ Hamiltonians takes the shape

$$\begin{aligned} U(T) &= e^{-i\beta_n H_d \Delta t} e^{-iH_c \Delta t}, ..., e^{-i\beta_1 H_d \Delta t} e^{-iH_c \Delta t} \\ &= U_d(\beta_n) U_c, ..., U_d(\beta_1) U_c . \end{aligned}$$

Each term $U_d(\beta_k)U_c$ with $k = 1, 2, 3, ..., l$ can be considered a layer in a quantum circuit. Each layer requires a β_k, which in FALQON is obtained from the previous layer: $\beta_{k+1} = -A_k$. Additionally, by manipulating Δt, the algorithm can alternate the execution of U_d and U_c in the same layer. These features enable estimations of $\langle H_c \rangle$ that improve monotonically with the evaluations of additional layers in the quantum circuit. Therefore, FALQON does not rely on an external (classical) optimization algorithm; everything happens in the quantum device. On the other hand, QAOA introduces an additional set of parameters in the cost Hamiltonian. A layer in QAOA takes the form $U_d(\beta_k)U_c(\gamma_k)$. The optimization in QAOA over the $2l$ parameters $\vec{\beta} = (\beta_1, \beta_2, ..., \beta_l)$ and $\vec{\gamma} = (\gamma_1, \gamma_2, ..., \gamma_l)$, which can be expressed as the minimization of $\langle \psi(\vec{\beta}, \vec{\gamma})|H_c|\psi(\vec{\beta}, \vec{\gamma})\rangle$, is performed by an external, classical optimizer. Therefore, despite using similar equations, QAOA is fundamentally different from FALQON.

Besides the implications of these two strategies for the number of circuit evaluations, QAOA seems to be more likely to get trapped in local minima than FALQON. Certainly, facilitated by their similar theoretical background, both approaches can complement each other when solving complex optimization problems.

Numerical experiments

Although both QAOA and FALQON can be applied to a variety of optimization problem, they both are usually found in the context of the Maximum Cut problem (MAX-CUT) in network theory. We run numerical experiments to measure execution differences between QAOA and FALQON running on a MAX-CUT problem with a small network of five nodes.

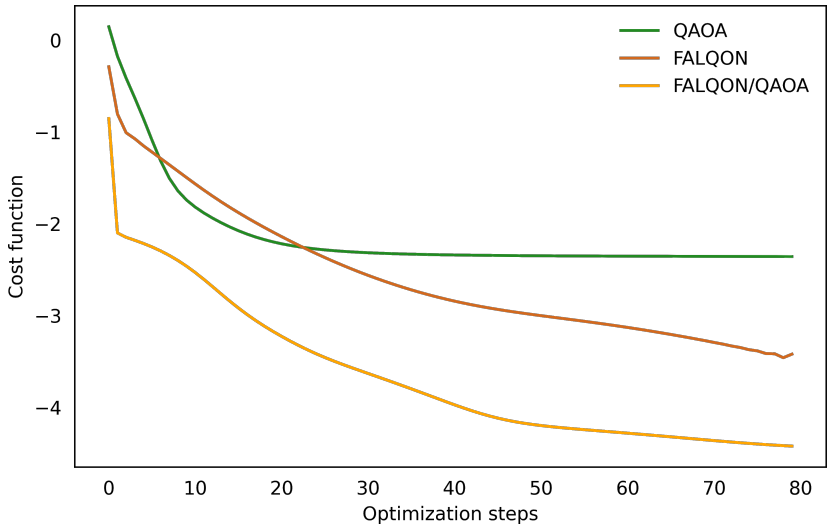

Figure 6 Minimization of the energy cost for a MAX-CUT problem on a five node network using QAOA, FALQON and a combination of both algorithms.

The results of the optimizations for this use case are not completely unexpected given the theoretical background on both methods. QAOA is a well-known optimization procedure, but Figure 6 shows that it can be sub-optimal in finding the global minimum of the cost function. While for simple networks, this is less likely to happen, for complex networks getting trapped in local minima rapidly becomes a liability. Regarding energy consumption though, QAOA uses quantum devices for the evaluation of the cost function and any other classically implemented optimization method to estimate the gradients, which means that it can be implemented with two circuit evaluations per training parameter per optimization step when using parameter-shift rules[23]. This is energetically convenient for small quantum circuits but as we saw in Section 3.2, it scales badly with the number of parameters.

In contrast to QAOA, FALQON appears relentless in its search for global minima, but this comes at a cost. While FALQON avoids using classical devices and optimization algorithms altogether, it relies on the execution of circuits of increasing size at each step. Therefore, the energy consumption per step increases with each circuit evaluation instead of being constant as it is with QAOA. FALQON is, however, energetically efficient in that it avoids the communication overhead between quantum and classical devices. However, the energy savings related to this are difficult to estimate.

Ignoring the overhead of the communication between classical and quantum devices, we can still estimate the energy consumption of the purely quantum calculation using our approach. For instance, a circuit with 25 gates performing five optimization steps will consume 22.5kJ in QAOA and 45kJ in FALQON. These numbers clearly show that FALQON can rapidly grow disadvantageous if its performance is not monitored carefully. For large optimization projects, it might be preferable to estimate or benchmark the consumption of both methods and use a combination of them as represented by the “FALQON/QAOA” curve in Figure 6. That effort would include, for instance, short runs of the energy-consuming FALQON algorithm to start the optimization or whenever QAOA gets trapped in a local minimum and longer runs of QAOA to minimize the cost function at relatively low energy consumption per step. Figure 6 indicates that this strategy is likely to outperform both individual methods without compromising on energy efficiency.

3.6 | Differentiable quantum transforms (DQTs)

Theory

In this section, we describe a strategy for quantum error mitigation implemented under the powerful framework of differentiable quantum transforms (DQTs)[40]. The idea of differentiation is commonplace in optimization and machine learning. DQTs provide a useful abstraction to differentiation and are implemented in the PennyLane library[18,40]. In previous sections, we already used differentiation in the context of PQCs. That

is, if one views a PQC as a quantum function or quantum program S with m parameters $\theta = (\theta_1, \theta_2, ..., \theta_m)$ then S is considered differentiable if $dS/d\theta_i$ is defined for every possible value of each θ_i. The parameter-shift rules [23] that we have used already to calculate partial derivatives in previous sections explicitly take advantage of the differentiability of PQCs. In this context, differentiation itself is a function transform in the sense that takes as input a function with differentiable variables θ (*i.e.*, a PQC or $f(\theta)$) and returns another function with derivatives $g(\theta) = \nabla f(\theta)$. A transform $\mathcal{T}$ can also have n parameters $\tau = (\tau_1, \tau_2, ..., \tau_n)$. A DQT is a function that preserves the differentiability of its transformation while itself being differentiable ($d\mathcal{T}/d\tau_i$ is defined for all τ_i). Additionally, DQTs are composable; that is, if $\mathcal{T}$ and $\mathcal{U}$ are DQTs then $\mathcal{V} = \mathcal{T} \cdot \mathcal{U}$ is also a DQT. This composability property facilitates the use of complex transformations in quantum computing. Here we use one application of DQTs relevant to quantum optimization: error mitigation.

Contemporary quantum computing requires programmers to be able to work with noisy devices [37]. To obtain precise results, quantum calculations necessarily will have to be backed by some form of error-correction or error-mitigation techniques. Given access to low-level hardware control, programmers can manipulate the pulse formulation of each gate, study the noise scaling, and re-calibrate the gates accordingly. However, doing so requires substantial time and expertise. An intermediate approach to use DQTs.

In the framework of DQTs, error mitigation is a transform $\mathcal{M}$ that reduces the noise of a quantum function $f^*(\theta)$ giving as output a mitigated function $\tilde{f}(\theta)$ which is closer to the exact noiseless quantum function $f(\theta)$. That is,

$$f^*(\theta) \longmapsto \tilde{f}(\theta) \simeq f(\theta) \, .$$

To be useful, $\mathcal{M}$ should still ensure that the resulting mitigated function is differentiable ($d\tilde{f}(\theta)/d\theta$ should exist for all values of θ). One such error mitigation transforms is zero noise extrapolation (ZNE) [41]. The basic idea of ZNE is to increase the noise in a quantum calculation in a controlled and scalable way, collect data along the way, and subsequently extrapolate back to estimate the expected value of the calculation at zero noise level. We start by defining the expected value of a quantum calculation at noise level λ as $E(\lambda)$. If $\lambda = 1$ is the current noise of the quantum calculation and $\lambda = 0$ is the ZNE we want to estimate, then we first need to find ways to scale noise (and measure $E(\lambda)$) for $\lambda > 1$. Although in principle, this can be done by correlating λ with any physical measure of noise in the system (*e.g.*, temperature), in practice ZNE is implemented using a trick called *unitary folding*.

If U is a unitary matrix (representing, *e.g.*, a gate, layer, or circuit) then U can be replaced by $U(UU^\dagger)^n$ because $UU^\dagger = I$. The expected value of the calculation does not change but the number of physical operations on the device scales as $1 + 2n$, and the noise should scale accordingly. Giurgica-Tiron et al. report further theoretical details and numerical experiments on noise scaling and extrapolation methods to achieve ZNE [41].

Numerical experiments

To evaluate noise control, we performed our simulations on a Hamiltonian that represents a quantum version of the Ising model:

$$H = -J\left(\sum_{\langle ij \rangle} Z_i Z_j + g \sum_j X_j\right) \, .$$

Here, the observables (Z_i, X_i) can be represented with Pauli gates, and J and g are arbitrary energy and coupling factors, respectively. The Ising model (even the quantum version) is one of the simplest systems for which analytic solutions have been derived. These solutions therefore can be evaluated directly (red dashed line in Figure 7) and estimated by optimization algorithms if the Hamiltonian is mapped to a parameterized quantum circuit.

Aside from providing idealized and noiseless quantum-computing simulators, the PennyLane library [18] also supports the creation of *ad hoc* noisy channels using a choice of noise types and a specified noise level. These noise channels then can be added to an ideal device to simulate noisy qubits or channels. Pennylane additionally implements the ZNE error mitigation technique [41] discussed in the theory section above. ZNE can be applied to a modeled noisy device to mitigate the errors during a calculation and obtain results closely resembling those obtainable on actual quantum devices.

We used Adam as the classical optimizer for the experiments represented by Figure 7. The Adam optimizer ran purely on the classical device (CPU) while the cost function was evaluated on quantum simulators with different noise levels roughly categorized as Noisy, Mitigated, and Ideal. The runs show that while the noise-mitigated device closely follows the ideal one, the value of the cost function never reaches the exact analytical solution. Only the ideal device is able to converge to the analytical solution. Similar results were obtained using V-GD as classical optimizer, but convergence to the exact solution by the ideal device required more optimization steps.

Although it seems obvious that error-mitigation techniques like ZNE should have a significant impact on the energy-efficiency of quantum optimization, unlike the other cases we explored in this article, this effect is difficult to quantify. We performed longer simulations using both the Adam optimizer and V-GD (not shown) and in both cases, the noiseless device eventually found the exact solution, but both the noisy and error-mitigated devices converged at different values of the cost function. We suggest that ZNE's energy-efficiency contributions to quantum optimization require additional investigations.

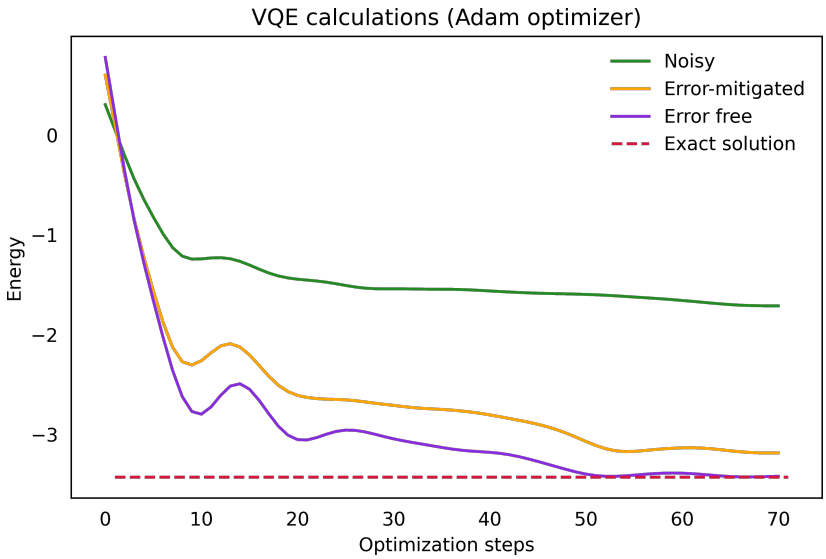

Figure 7 Circuit optimizations highlighting the effect of the Zero Noise Extrapolation (ZNE) error mitigation technique.

While error mitigation is not the primary subject of this paper, in the next section we study the absence of error mitigation in some detail by evaluating a quantum optimization algorithm (QAOA) on quantum hardware. Specifically, we illustrate the relationship between the success rate of these calculations, which are heavily impacted by noise, and their energy efficiency.

4 | EVALUATION ON QUANTUM HARDWARE

Section 3 evaluated the energy of quantum algorithms using numerical experiments. This section complements that evaluation with empirical studies on modern quantum hardware. We constrain our scope to the QAOA algorithm [34], which was discussed in Section 3.5, configured to solve the Minimum Vertex Cover problem. We use the NP-hard formulation of the problem: Given a graph, what is the smallest subset of vertices such that at least one endpoint of each edge lies in that subset? (This subset is called the minimum vertex cover.) By way of example, Figure 8 presents a 20-vertex (disjoint) graph with a minimum vertex cover shaded. Note that minimum vertex covers are not necessarily unique.

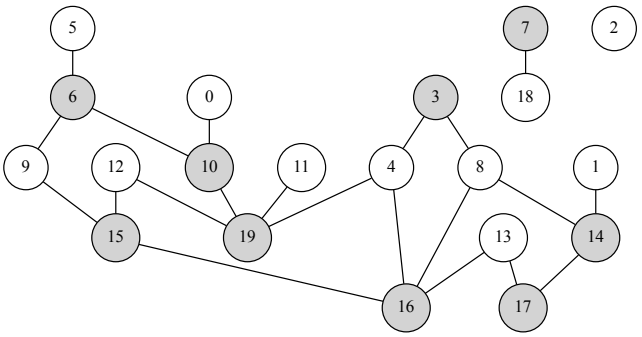

Figure 8 A 20-vertex graph with a minimum vertex cover shaded

4.1 | Methods

As discussed in Section 3.5, QAOA solves problems by minimizing a cost Hamiltonian H_c. Consequently, finding a minimum vertex cover for a given graph must be expressed as a Hamiltonian and ultimately as a quantum circuit. Rather than manually work out a circuit for each problem

instance, we rely on a higher-level programming system, NchooseK, which compiles a set of problem-specific hard and soft constraints into a QAOA formulation and runs this on a quantum computer[42,43]. NchooseK provides the added benefit that it can compile the same constraints for gate-based quantum computers, quantum annealers, and classical solvers. A quantum annealer is a device that is hard-wired to minimize a quadratic H_c using a procedure that is analogous to QAOA but implemented in hardware.

NchooseK is based on repeated use of a single primitive, notated as $nck(N, K)$ and interpreted as "K of N Boolean variables must be *true*". More precisely, N is a multiset of variables and K is a set of whole numbers. If exactly $k \in K$ of the variables in N are *true*, the constraint is satisfied. For example, $nck(\{A, B\}, \{0, 2\})$ is satisfied by $\{A = false, B = false\}$ or $\{A = true, B = true\}$ but no other possibilities. Variables can be repeated both within and across constraints. All instances of a variable are assigned the same value. In addition to the default, "hard" constraints, NchooseK supports "soft" constraints, denoted by $nck(N, K, \text{soft})$. The semantics is that NchooseK will satisfy all hard constraints and as many soft constraints as possible.

A Minimum Vertex Cover problem is straightforward to express in NchooseK:

$$\begin{aligned}&\text{Given a graph } G = (V, E),\\ &\forall (u, v) \in E, nck(\{u, v\}, \{1, 2\}), \text{ and}\\ &\forall v \in V, nck(\{v\}, \{0\}, \text{soft}).\end{aligned}$$

In this formulation, each vertex has an associated Boolean variable, with *true* indicating inclusion in the minimum vertex cover. The first line of constraints requires that each edge have either 1 or 2 of its vertices set to *true*. This constructs a vertex cover but not necessarily a minimal one. The second line of constraints requests—but does not require—that each vertex variable be set to *false* (*i.e.*, not part of the vertex cover). Because NchooseK will minimize the number of vertices it sets to *true*, this promotes the construction of a minimum vertex cover.

The NchooseK compiler first translates a problem into a quadratic unconstrained binary optimization (QUBO) problem: $\arg\min_x x^T Q x$, with $Q \in \mathbb{R}^{N \times N}$ and $x \in \{0, 1\}^N$. For example, NchooseK compiles the graph shown in Figure 8 to the matrix

$$Q = \begin{bmatrix}
-20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & -41 & 21 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & -62 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & -20 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & -62 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -62 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -41 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & -62 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -41 & 0 & 0 & 21 & 0 & 0 & 0 & 21\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -41 & 0 & 0 & 21 & 21 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0 & -62 & 0 & 0 & 21 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0 & -62 & 21 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -83 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -41 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20 & 0\\
0 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -83
\end{bmatrix}$$

This, in turn, is compiled to a circuit for H_c constructed (nominally) with one $R_Z(\theta)$ gate per array element, with the rotation angle being a function of the element's value, plus an two additional CNOT gates per off-diagonal element. Our argument is that it is much easier to express a Minimum Vertex Cover problem with NchooseK than to construct a circuit for H_c by hand.

The following quantum computers were used for an empirical evaluation of the energy required to solve Minimum Vertex Cover problems:

1. **IBM Q Cusco**. Cusco is a 127-qubit Eagle r3 circuit-model quantum processor (v1.0.4). On this platform we consider both one and two QAOA iterations (sets of $\{\beta, \gamma\}$ parameters).

2. **D-Wave Advantage**. The two specific annealing-model quantum processors used were a 5614-qubit Advantage (v6.2) and a 563-qubit Advantage2 prototype (v1.1).

Because QAOA is a variational quantum algorithm, it runs multiple quantum jobs until a termination criterion is reached. In all cases, we use $N_{shots} = 1000$, as in the rest of this paper.

In the current noisy intermediate-scale quantum (NISQ) era[37], errors are rampant. Hence, a meaningful energy estimate must take the QAOA algorithm's success rate into consideration. We use as our metric the estimated mean energy to a successful solution, which we define as follows:

$$E_{QAOA} = E_{total} \times N_{jobs}/P_{success}\ , \tag{3}$$

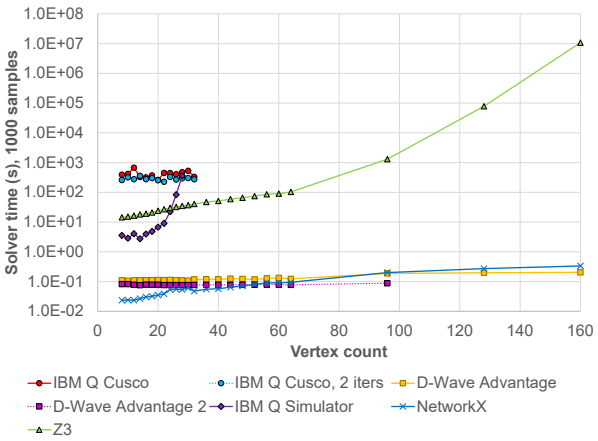

Figure 9 Time to propose 1000 solutions to a minimum vertex cover problem

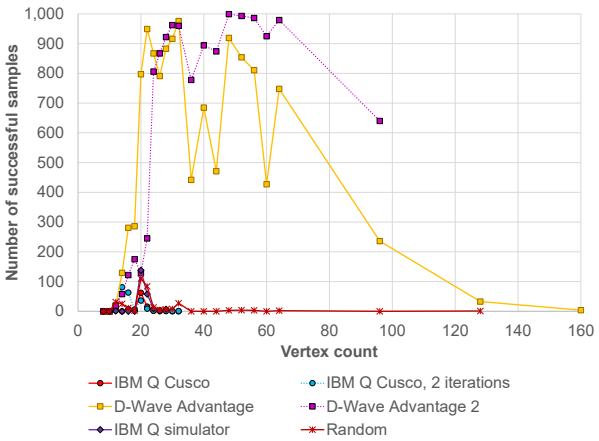

Figure 10 Number of vertex covers found that are no larger than that found by a classical heuristic

where E_{total} is as defined by Equation 2 and uses $E_{gate} = 0.18$ as per Table 1; N_{jobs} is the number of jobs needed for Qiskit's[9] QAOA implementation to converge (an average of 30 for one QAOA iteration and an average of 46 for two iterations); and $P_{success}$ is the probability of observing a successful sample based on the N_{shots} shots taken.

To estimate the energy consumed by solving a Minimum Vertex Cover problem on the D-Wave Advantage quantum annealer we follow a similar but not identical approach:

$$E_{QA} = P \times T/(N_{shots} \times P_{success}) \,, \tag{4}$$

where P is the system's maximum rated power (25 kW according to its data sheet[44]), T is the total QPU (quantum processing unit) access time for N_{shots} shots, and $P_{success}$ is the probability of observing a successful sample.

We define the following terms to describe each sample:

invalid The subset of vertices proposed by the QPU do not cover all edges.

valid The vertices cover all edges, but strictly more are required than the number determined by a classical heuristic.

successful Equal or fewer vertices appear in the cover than what were found by the classical heuristic.

optimal The vertices represent a true minimum vertex cover as determined by a classical, complete solver.

That is, a solution must be valid to be successful and successful to be optimal.

As a classical heuristic, we use the `min_weighted_vertex_cover` function from NetworkX[45]. This returns a vertex cover whose size is guaranteed to be no larger than twice the minimum vertex cover. We classically compute the true minimum vertex cover using Microsoft Research's Z3 satisfiability modulo theories (SMT) solver[46], which is supported as an NchooseK compiler back end and therefore works from the same problem specification as the quantum back ends.

4.2 | Empirical results

We measured the QPU time needed for the IBM Q and D-Wave Advantage to propose 1000 solutions to a Minimum Vertex Cover problem. Problem sizes ranged from 8–160 vertices. To account for variability, each experiment was repeated five times. Figure 9 plots, on a logarithmic scale, the median of those five runs for IBM Q Cusco performing one iteration of QAOA, IBM Q Cusco performing two iterations of QAOA, and D-Wave Advantage and D-Wave Advantage2 performing quantum annealing in hardware. For comparison to a classical solution, Figure 9 includes three additional curves. The IBM Q Simulator curve represents a cloud-based, noise-free simulator running the QAOA algorithm. The NetworkX curve represents NetworkX's heuristic solution to the Minimum Vertex Cover problem. The Z3 curve represents Z3's guaranteed-exact solution to the Minimum Vertex Cover problem.

The data show that the circuit-model runs take substantially longer than the annealing-model runs. Some of this is to be expected because QAOA requires tens of parameterizations of each circuit, at 1000 shots apiece, versus a single 1000-shot run on the quantum annealer. The quantum

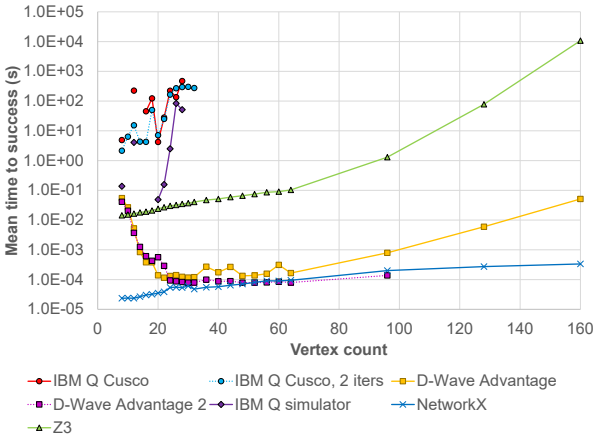

Figure 11 Average time to find a vertex cover of size no larger than that found by a classical heuristic

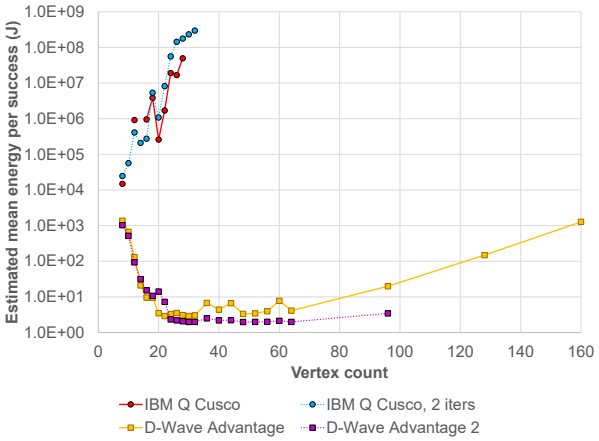

Figure 12 Estimated energy to find a vertex cover of size no larger than that found by a classical heuristic

simulator requires time that increases exponentially in the number of vertices (and therefore simulated qubits) as would be expected due to the cost of exploring an exponential state space. The classical heuristic requires time that grows relatively slowly with vertex count. The time for the exact solver grows exponentially as would be expected for an NP-hard problem.

The next figure, Figure 10, tallies the number of successful samples out of the 1000 samples taken. Recall from Section 4.1 that "successful" is defined as "at least as good as NetworkX's classical heuristic". Each data point plotted in Figure 10 represents the median of five runs. NetworkX and Z3 data are omitted from the graph because they are guaranteed to observe a 100% success rate.

The results are fairly pessimistic for QAOA. Cusco observes a maximum success rate of only 80 samples (1 QAOA iteration) or 120 samples (2 iterations) out of 1000. To distinguish whether environmental noise or the QAOA algorithm is the greater source of the low success rates, we include results from IBM Q's noise-free quantum simulator. The success rate is not much better, achieving a maximum of only 138 out of the 1000 samples. We therefore can conclude that QAOA's failure to converge to a successful solution is more of an issue than the quantum state being disrupted. To put QAOA's success rate in context Figure 10 includes a Random curve, which represents a random selection of between $k/2$ and k vertices, where k is the size of NetworkX's heuristic solution. The IBM Q Cusco, IBM Q Cusco 2 iters, and IBM Q Simulator curves are qualitatively similar to Random, implying little benefit over random guessing. (The coefficients of correlation are 0.90, 0.54, and 0.70, respectively.) Interestingly, QAOA is far more likely to produce a *valid* vertex covering than random guessing, averaging 328, 300, and 586 samples out of 1000 for graphs of size 8–28 versus only 43 for random guessing. No optimal solutions were observed for graphs of larger than 16 vertices, which is why we stopped measuring QAOA data long before running out of qubits.

The D-Wave runs fare much better, reaching a peak success rate of 976/1000 for the Advantage and 999/1000 for the Advantage2. However, success rates drop off rapidly as the problem size increases.

Figure 11 divides the Figure 9 data by the Figure 10 data to compute the mean time to success. That is, if all 1000 shots are successful—the case for all of the NetworkX and Z3 data—the mean time to success is the time for a single shot. If fewer than 1000 shots are successful, the mean time to success corresponds to the time for multiple shots. While visually similar to Figure 9, Figure 11 includes more separation between the curves, highlighting the need for additional runs when success rates are low. While the IBM Q Simulator runs faster than Z3 on small problems, its mean time to success is higher due to its low success rates.

Given the timing data from Figure 11, it is straightforward to apply Equations 3 and 4 to compute the energy for, respectively, IBM Q Cusco and D-Wave Advantage. The results are plotted in Figure 12. As indicated by that figure, the D-Wave Advantage2 consumes the least energy to produce a vertex cover that is no worse than that produced by NetworkX's classical heuristic, with the D-Wave Advantage being only slightly worse. The IBM Q Cusco jobs consume substantially more energy. The high energy costs are due in part to QAOA's low success rates on the Minimum Vertex Cover problem, in part to the circuit area (well over 10,000 for the larger circuit sizes), and in part to the number of QAOA jobs needed to inform the minimum eigenvalue optimizer.

One characteristic of QAOA is that its classical optimizer (*e.g.*, COBYLA[47]) requires multiple samples to form a gradient, which is used to extrapolate new $\{\beta, \gamma\}$ parameters for each job launch. In particular, a single sample—even on an ideal, noise-free QPU or simulator—is inadequate. Hence, the QAOA data in Figures 11 and 12 are more properly interpreted not as the time or energy for a single success but as $1/N_{shots}$ the time or energy needed for N_{shots} successes, assuming a suitably large N_{shots} (*e.g.*, the 1000 used throughout this article).

5 | DISCUSSION

Classical computers' energy consumption is dominated by the electronic circuitry. In contrast, most of the energy consumed by quantum computers goes to secondary tasks such as cooling the quantum system to temperatures close to absolute zero. While the physics related to the manufacture of these cryogenic systems is relatively well understood [48], much less is known about the energetic details of the quantum operations themselves. This is challenging both at an abstract level (*i.e.*, the theoretical energy of the quantum operations themselves, independent of the support infrastructure), and also at a more practical level (*i.e.*, the energy transformations related to specific quantum technologies). The energy transformations during the execution of a quantum algorithm running on a selected quantum device is a complex combination of the aforementioned factors.

In this study we explored the potential for energy efficiency in quantum optimization algorithms. We also explored the impact of other techniques designed for the improvement of practical issues in quantum optimizations. For instance, it would be desirable to reduce of the number of circuit evaluations (shots) and still make good estimates of the expected value of the cost Hamiltonian H_C.

We approached the topic by first exploring the limited literature on energy profiles of quantum computation. We supported the findings in this section with additional estimates of energy consumption in both simulators and quantum devices. Although Table 1 reports the energy consumption for several gate technologies, in practice we only used the values corresponding to superconducting qubits because they are the most widespread in the platforms (IBM Q) and libraries (PennyLane) used in this paper. We extended energy estimates from gates to a small quantum circuit that was executed on several simulators and devices (Table 2). These experiments provide a baseline for the variability of quantum energy evaluations in a single device and also among different quantum devices. The same basic assumptions for energy consumption were then extended to other sections of the paper describing numerical experiments with different optimization algorithms.

We also describe the theoretical basis of each algorithm presented here, highlighting those features with practical consequences for their energy efficiency. For instance, algorithms can differ in the number of quantum circuit evaluations per optimization step they need. While trying to account for different approaches, some common features that can be directly linked to energy efficiency can be understood by analyzing the theory. Trade offs between minimization of the cost function and energy efficiency are common across several algorithms. For instance, QN-GD needs more circuit evaluations per step than V-GD and therefore appears disadvantaged from an energy-consumption perspective. However, QN-GD more than compensates for this disadvantage by searching better for the global minimum. That is, while QN-GD executes more circuit evaluations per optimization step, it needs fewer steps to find the minimum and ends up saving energy. The ultimate reason for the improved optimization efficiency in QN-GD is the introduction of a factor (the Fubini-Study metric tensor) that generalizes the geometry of the parameter search space. For other algorithms, however, we observe that while the reason for the advantage might differ (*e.g.*, introduction of gradient approximations for SPSA, gate optimization for Rotoselect, etc.), the effect is similar to that described for QN-GD. We note that in some cases, such as with the FALQON algorithm, the energy efficiency can fade rapidly if the global minimum is not found in relatively few steps. While FALQON has the advantage of performing its optimization purely in a quantum device, the number of layers in the circuit grows at every iteration and therefore each circuit evaluation becomes more expensive in terms of energy consumption. As the optimization problem grows in scale (*e.g.*, MAX-CUT applied to increasingly large networks), it becomes more advantageous to use a combination of FALQON and QAOA to exploit both the relatively high search efficiency of FALQON and energy-efficiency of QAOA.

We performed numerical experiments to gather evidence of optimization algorithms' energy efficiency. We found that for the most part the experiments support the expectations behind the theoretical basis of the selected algorithms. Specifically, most of the quantum optimization algorithms explored here appear to be between 2 and 4 times more energy-efficient than their classical counterparts (with the exception of FALQON). On the other hand, a general technique like Rosalin showed a substantial energy advantage, with $\geq 20\times$ gains) Although ZNE was not methodically examined here, we believe that this technique also has great potential.

A systematic exploration of energy efficiency in quantum optimization is far from trivial. In the theory sections, we highlighted different strategies, some still under active development. In this regard, our study has some known shortcomings. For instance. we could not easily accommodate algorithms using specialized gates [49], techniques to escape barren plateaus during optimization [50] or the use of quantum analytic descent to build approximate classical models of the quantum landscape [51], to name a few. In our exploration of these topics, we found invaluable the code base, demos and papers from the development communities of both Pennylane [18] and Qiskit [9].

While numerical studies are important, measurements taken on quantum hardware help put these studies in perspective. Implementing optimization problems (specifically, Minimum Vertex Cover problems) with NchooseK [42] enabled these problems to be run on gate-based quantum computers by expressing them as cost Hamiltonians to QAOA [34] and the exact same problem to be run on quantum annealers by expressing them as QUBOs. Under this fair comparison we found that QAOA struggled to converge to a solution that was at least as good as NetworkX's [45] classical heuristic. The quantum annealer—which is designed specifically to solve problems of this form—fared much better. As a result, the energy required to find a solution as good as the classical heuristic differed by several orders of magnitude between the two types of quantum computers.

6 | CONCLUSIONS

In this paper we explored the energy efficiency of quantum optimization algorithms. We reviewed the existing literature regarding the energy consumed in quantum computing operations and performed simulations to estimate power-based measures for a toy circuit and a generic superconducting qubit. Subsequently, we theoretically and numerically explored quantum optimization algorithms and related techniques. While providing numerical examples of energy (and power) savings for each case, we determined that some of the algorithms (QN-GD, SPSA, Rotosolve/Rotoselect) are between $2\times$ to $4\times$ more energy-efficient than their classical counterparts. A notable exception was FALQON, an algorithm that uniquely does not use classical devices but does enlarge its circuit with each iteration, which is devastating to energy efficiency. Rosalin, a technique that can be applied to other optimization algorithms, reducing energy by at least $20\times$ per minimization step. This improvement is accomplished by optimizing the number of shots (repetitions) needed to estimate the expected value of the cost function. Error-mitigation techniques like ZNE, although general and clearly effective, require additional study to assess their impact on energy consumption. Using QAOA to solve NP-hard optimization problems on contemporary gate-based quantum hardware consumes a substantial amount of energy. Our measurements indicate that this is due more to the number of circuits and number of shots per circuit needed to converge to a solution than it is to environmental noise, as evidenced by similar circuit counts being required on a noise-free quantum simulator.

7 | ACKNOWLEDGMENTS

Some of the research presented in this paper was supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory under project number 20210397ER. Los Alamos National Laboratory is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy (contract no. 89233218CNA000001). This research used quantum resources provided by Los Alamos National Laboratory's Institutional Computing Program. A subset of quantum resources were made available through a subcontract with the New Mexico Consortium. We would like to thank Ellis Wilson for adding IBM Q support to the NchooseK compiler and for expressing Minimum Vertex Cover with NchooseK.

The authors have declared no conflict of interest.

8 | DATA AVAILABILITY STATEMENT

Data available on request from the authors.

References

1. Preskill J. Quantum computing 40 years later. 2023.
2. Gibney E. Quantum Computer Race Intensifies as Alternative Technology Gains Steam. *Nature* 2020; 587(7834): 342—343. doi: 10.1038/d41586-020-03237-w
3. Koch J, Yu TM, Gambetta J, et al. Charge-Insensitive Qubit Design Derived from the Cooper Pair Box. *Phys. Rev. A* 2007; 76: 042319. doi: 10.1103/PhysRevA.76.042319
4. Bourassa J, Alexander R, Vasmer M, et al. Blueprint for a Scalable Photonic Fault-Tolerant Quantum Computer. *Quantum* 2021; 5: 392. doi: 10.22331/q-2021-02-04-392
5. Childress L, Hanson R. Diamond NV Centers for Quantum Computing and Quantum Networks. *MRS Bulletin* 2013; 38(2): 134-138. doi: 10.1557/mrs.2013.20
6. Aguado R, Kouwenhoven LP. Majorana Qubits for Topological Quantum Computing. *Physics Today* 2020; 73(6): 44-50. doi: 10.1063/PT.3.4499
7. Shumiya N, Hossain MS, Yin J, et al. Evidence of a Room-Temperature Quantum Spin Hall Edge State in a Higher-Order Topological Insulator. *Nature Materials* 2022; 21: 1-5. doi: 10.1038/s41563-022-01304-3
8. Bravyi S, Dial O, Gambetta J, Gil D, Nazario Z. The Future of Quantum Computing with Superconducting Qubits. *Journal of Applied Physics* 2022; 132: 160902. doi: 10.1063/5.0082975

9. Qiskit contributors . Qiskit: An Open-Source Framework for Quantum Computing. https://github.com/Qiskit; 2023

10. Fellous-Asiani M. The Resource Cost of Large Scale Quantum Computing. https://doi.org/10.48550/arXiv.2112.04022; 2022.

11. Jaschke D, Montangero S. Is Quantum Computing Green? An Estimate for an Energy-Efficiency Quantum Advantage. *Quantum Science and Technology* 2023; 8(2): 025001. doi: 10.1088/2058-9565/acae3e

12. Moll N, Barkoutsos P, Bishop LS, et al. Quantum Optimization using Variational Algorithms on Near-Term Quantum Devices. *Quantum Science and Technology* 2018; 3(3): 030503. doi: 10.1088/2058-9565/aab822

13. Baldwin CH, Mayer K, Brown NC, Ryan-Anderson C, Hayes D. Re-examining the Quantum Volume Test: Ideal Distributions, Compiler Optimizations, Confidence Intervals, and Scalable Resource Estimations. *Quantum* 2022; 6: 707. doi: 10.22331/q-2022-05-09-707

14. Miller K, Broomfield C, Cox A, Kinast J, Rodenburg B. An Improved Volumetric Metric for Quantum Computers via more Representative Quantum Circuit Shapes. https://arxiv.org/abs/2207.02315; 2022.

15. Algorithmic Qubits: a Better Single-Number Metric. https://ionq.com/resources/algorithmic-qubits-a-better-single-number-metric; 2023.

16. Moll N, Barkoutsos P, Bishop L, et al. Quantum Optimization using Variational Algorithms on Near-Term Quantum Devices. *Quantum Science and Technology* 2017; 3. doi: 10.1088/2058-9565/aab822

17. Benedetti M, Lloyd E, Sack S, Fiorentini M. Parameterized Quantum Circuits as Machine Learning Models. *Quantum Science and Technology* 2019; 4. doi: 10.1088/2058-9565/ab4eb5

18. Bergholm V, Izaac J, Schuld M, et al. PennyLane: Automatic Differentiation of Hybrid Quantum-Classical Computations. https://doi.org/10.48550/arXiv.1811.04968; 2018.

19. Amari Si. Natural Gradient Works Efficiently in Learning. *Neural Computation* 1998; 10(2): 251-276. doi: 10.1162/089976698300017746

20. Yamamoto N. On the Natural Gradient for Variational Quantum Eigensolver. https://doi.org/10.48550/arXiv.1909.05074; 2019.

21. Mondal D. Generalized Fubini-Study Metric and Fisher Information Metric. https://doi.org/10.48550/arXiv.1503.04146; 2015.

22. Stokes J, Izaac J, Killoran N, Carleo G. Quantum Natural Gradient. *Quantum* 2020; 4: 269. doi: 10.22331/q-2020-05-25-269

23. Schuld M, Bergholm V, Gogolin C, Izaac J, Killoran N. Evaluating Analytic Gradients on Quantum Hardware. *Phys. Rev. A* 2019; 99: 032331. doi: 10.1103/PhysRevA.99.032331

24. Wierichs D, Izaac J, Wang C, Lin CYY. General Parameter-Shift Rules for Quantum Gradients. *Quantum* 2022; 6: 677. doi: 10.22331/q-2022-03-30-677

25. Kiefer J, Wolfowitz J. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics* 1952; 23(3): 462–466.

26. Spall JC. Multivariate Stochastic Approximation using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control* 1992; 37(3): 332-341. doi: 10.1109/9.119632

27. Gacon J, Zoufal C, Carleo G, Woerner S. Simultaneous Perturbation Stochastic Approximation of the Quantum Fisher Information. *Quantum* 2021; 5: 567. doi: 10.22331/q-2021-10-20-567

28. Schuld M, Bocharov A, Svore KM, Wiebe N. Circuit-Centric Quantum Classifiers. *Phys. Rev. A* 2020; 101: 032308. doi: 10.1103/PhysRevA.101.032308

29. Ostaszewski M, Grant E, Benedetti M. Structure Optimization for Parameterized Quantum Circuits. *Quantum* 2021; 5: 391. doi: 10.22331/q-2021-01-28-391

30. Roffe J. Quantum Error Correction: An Introductory Guide. *Contemporary Physics* 2019; 60(3): 226-245. doi: 10.1080/00107514.2019.1667078

31. Arrasmith A, Cincio L, Somma R, Coles P. Operator Sampling for Shot-frugal Optimization in Variational Algorithms. https://doi.org/10.48550/arXiv.2004.06252; 2020.

32. Kübler J, Arrasmith A, Cincio L, Coles P. An Adaptive Optimizer for Measurement-Frugal Variational Algorithms. *Quantum* 2020; 4: 263. doi: 10.22331/q-2020-05-11-263

33. Balles L, Romero J, Hennig P. Coupling Adaptive Batch Sizes with Learning Rates. https://doi.org/10.48550/arXiv.1612.05086; 2017.

34. Farhi E, Goldstone J, Gutmann S. A Quantum Approximate Optimization Algorithm. https://doi.org/10.48550/arXiv.1411.4028; 2014.

35. Magann A, Rudinger K, Grace M, Sarovar M. Feedback-Based Quantum Optimization. *Physical Review Letters* 2022; 129(25): 250502. doi: 10.1103/PhysRevLett.129.250502

36. Khaneja N, Glaser S. Cartan Decomposition of SU(2*n*) and Control of Spin Systems. *Chemical Physics* 2001; 267: 11-23. doi: 10.1016/S0301-0104(01)00318-4

37. Preskill J. Quantum Computing in the NISQ Era and Beyond. *Quantum* 2018; 2: 79. doi: 10.22331/q-2018-08-06-79

38. Dhand I, Sanders B. Stability of the Trotter-Suzuki Decomposition. *Journal Of Physics A: Mathematical And Theoretical* 2014; 47: 265206. doi: 10.1088/1751-8113/47/26/265206

39. Cong S, Fangfang M. A Survey of Quantum Lyapunov Control Methods. *The Scientific World Journal* 2013; 18: 967529. doi: 10.1155/2013/967529

40. Matteo O, Izaac J, Bromley T, et al. Quantum Computing With Differentiable Quantum Transforms. *ACM Transactions on Quantum Computing* 2023. doi: 10.1145/3592622

41. Giurgica-Tiron T, Hindy Y, LaRose R, Mari A, Zeng WJ. Digital Zero Noise Extrapolation for Quantum Error Mitigation. *IEEE International Conference on Quantum Computing and Engineering (QCE)* 2020: 306-316. doi: 10.1109/QCE49297.2020.00045

42. Wilson E, Mueller F, Pakin S. Mapping Constraint Problems onto Quantum Gate and Annealing Devices. In: *Second International Workshop on Quantum Computing Software*, IEEE/ACM. ; 2021; St. Louis, Missouri, USA. doi: 10.1109/QCS54837.2021.00016

43. Wilson E, Mueller F, Pakin S. Combining Hard and Soft Constraints in Quantum Constraint-Satisfaction Systems. In: *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, IEEE/ACM. IEEE Computer Society; 2022; Dallas, Texas, USA: 161–174. doi: 10.1109/SC41404.2022.00018

44. D-Wave Systems, Inc. . Advantage: The First and Only Quantum Computer Built for Business. data sheet, D-Wave Systems, Inc.; Burnaby, Canada: 2022.

45. Hagberg AA, Schult DA, Swart PJ. Exploring Network Structure, Dynamics, and Function using NetworkX. In: Varoquaux G, Vaught T, Millman J., eds. *Proceedings of the 7th Python in Science Conference (SciPy 2008)*; 2008; Pasadena, California, USA: 11–15.

46. de Moura L, Bjørner N. Z3: An Efficient SMT Solver. In: Ramakrishnan CR, Rehof J., eds. *Tools and Algorithms for the Construction and Analysis of Systems*. 4963 of *Lecture Notes in Computer Science*. Springer; 2008; Berlin and Heidelberg, Germany: 337–340.

47. Powell MJD. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*: 51–67; Dordrecht, Netherlands: Springer . 1994.

48. Martin MJ, Hughes C, Moreno G, et al. Energy Use in Quantum Data Centers: Scaling the Impact of Computer Architecture, Qubit Performance, Size, and Thermal Parameters. *IEEE Transactions on Sustainable Computing* 2022; 7(4): 864-874. doi: 10.1109/TSUSC.2022.3190242

49. Wiersema R, Lewis D, Wierichs D, Carrasquilla J, Killoran N. Here Comes the SU(*N*): Multivariate Quantum Gates and Gradients. https://doi.org/10.48550/arXiv.2303.11355; 2023.

50. Grant E, Wossnig L, Ostaszewski M, Benedetti M. An Initialization Strategy for Addressing Barren Plateaus in Parametrized Quantum Circuits. *Quantum* 2019; 3: 214. doi: 10.22331/q-2019-12-09-214

51. Koczor B, Benjamin S. Quantum Analytic Descent. *Physical Review Research* 2022; 4: 023017. doi: 10.1103/PhysRevResearch.4.023017