Estimating energy-efficiency in quantum optimization algorithms

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

Rosa M Badia
Barcelona Supercomputing Center
Barcelona, Spain
rosa.m.badia@bsc.es

Barbara Chapman
Hewlett Packard Enterprise
New York, NY USA
barbara.chapman@hpe.com

Kirk Bresniker
Hewlett Packard Labs
Milpitas, CA USA
kirk.bresniker@hpe.com

Aditya Dhakal
Hewlett Packard Labs
Milpitas, CA USA
aditya.dhakal@hpe.com

Eitan Frachtenberg
Hewlett Packard Labs
Portland, OR USA
eitan.frachtenberg@hpe.com

Gourav Rattihalli
Hewlett Packard Labs
Milpitas, CA USA
gourav.rattihalli@hpe.com

Ninad Hogade
Hewlett Packard Labs
Ft Collins, CO USA
ninad.hogade@hpe.com

Pedro Bruel
Hewlett Packard Labs
Milpitas, CA USA
bruel@hpe.com

Alok Mishra
Hewlett Packard Labs
Milpitas, CA USA
alok.mishra@hpe.com

Dejan Milojicic
Hewlett Packard Labs
Milpitas, CA USA
dejan.milojicic@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 due to 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. A way to achieve this is by using Parameterized Quantum Circuits (PQCs) in optimization or even machine learning tasks.**

The energy consumption of quantum algorithms has been poorly studied. Here 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 a technique like Rosalin, could boost the energy-efficiency of other algorithms by a factor of $\geq 20\times$.

***Index Terms*—quantum optimization, circuit learning, shot optimization, error mitigation, heterogeneous computing, sustainability.**

I. Introduction

A. Quantum Technologies

Even after nearly a century of quantum theory, the race to practical Quantum Computing (QC) is still in full swing [1]. To minimize decoherence and other errors in QC, popular technologies like superconducting transmon qubits [3] rely on extremely low temperatures to fine-tune the manipulation of the quantum states. Maintaining these low temperatures, although not directly related to the calculations, is energy-expensive. Alternative low-temperature QC technologies, or instance diamond nitrogen vacancy (NV) qubits [4] and photonic quantum computers [7], are also emerging.

Perhaps one of the most challenging quantum computing technologies to implement, both theoretically and pragmatically, is topological quantum computer which will use the elusive Majorana qubits [30]. Although these computers are mostly theoretical at the moment, the technology to build them is within reach. While topological quantum computers might have advantages to solve some problems, their technology as envisioned today might still be dependent of relatively low temperatures although this is still an area of active research [5].

It is too soon to predict how and when these competing technologies will survive the quantum race. Additionally, besides the energy spent on cooling systems, we must also consider the energy requirements of the quantum calculations themselves. Lastly, besides qubit technologies, algorithmic choices can also make a difference for sustainability. In fact, in this work we explore the energetic advantages of quantum optimization algorithms over classical ones under the current computing models that implement quantum algorithms as hybrid workflows on heterogeneous classic/quantum devices.

B. Hybrid Workflows and Quantum Algorithms

The discovery of quantum error correction codes injected new energy into QC research and development. However, we are 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.

One research area is the development of hybrid workflow environment that include computations to be executed in quantum computers and classical supercomputers. These hybrid environments require dynamic runtimes that optimize the execution of the applications considering all the computer resources, both quantum and classical. There are multiple challenges to solve. First, there is no standard for the development of 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. Finally, there is no clear solution to communicating data between the quantum and classic systems. Concomitantly, the adoption of serverless technology in heterogeneous quantum/classic computing environments like IBM Qiskit runtime [11] could enhance the design flexibility for hybrid workflows and algorithms by fine-tuning and facilitating the use of these technologies by the final users [6]. These novelties, however, are still only on the development roadmap and to the best of our knowledge no implementation has yet been integrated and/or released in any mayor quantum computing library.

II. Energy consumed in quantum operations

Different quantum technologies should almost necessarily differ greatly in their power fairly scarce. A full-stack energy model for superconducting qubits was however described in a recent thesis [39]. Of practical utility, their final results can be use 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_{qubit}) and the depth of the circuit (N_{depth}). The area of the shape $A_{shape} = N_{qubit} \times N_{depth}$, 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 can also be estimated [39].

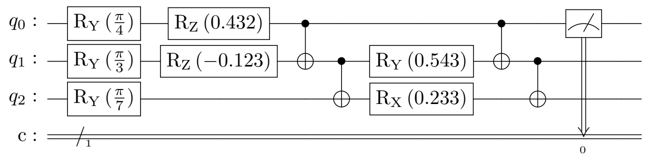
To estimate energy consumption per gate we leverage the work of Daniel Jaschke and Simone Montangero [40], that recently discussed green quantum advantage of a few algorithms and hardware platforms and also provided estimates that enabled computing the energy consumed by different quantum gate technologies. An estimate for the energy consumed by a quantum circuit can be written

$$E_{\text{circuit}} = (g \times r \times P_{\text{system}}) \,/\, \omega_{\text{gate}}, \tag{1}$$

where g is the number of gates in the circuit, r is the number of circuit repetitions needed to achieve the required fidelity, P_{power} is the total system power during circuit execution including cooling, and ω_{gate} is the estimated gate application frequency. We plugged $g = 1$, $r = 1000$, and numbers for different quantum technologies into Equation 1 to compute estimates for the energy consumed by Rydberg Atoms and Superconducting technologies, and we used a different estimate from Jaschke and Montangero [40] to compute estimates for the Trapped Ion technology. Our estimates for the energy consumed by a quantum gate using these technologies is shown in Table II.

Quantum Tech.	**Gate operation energy (J)**
Rydberg Atoms	$\approx 15 \times 10^3$
Trapped Ion	≈ 15.0
Superconducting	≈ 0.18

TABLE I
APPROXIMATE ENERGY CONSUMED DURING THE OPERATION OF GENERIC QUANTUM GATES USING DIFFERENT QUANTUM COMPUTING TECHNOLOGIES. HARDWARE [40]



q_0 :
q_1 :
q_2 :
c :
$\mathrm{R_Y}\left(\frac{\pi}{4}\right)$
$\mathrm{R_Y}\left(\frac{\pi}{3}\right)$
$\mathrm{R_Y}\left(\frac{\pi}{7}\right)$
$\mathrm{R_Z}\,(0.432)$
$\mathrm{R_Z}\,(-0.123)$
$\mathrm{R_Y}\,(0.543)$
$\mathrm{R_X}\,(0.233)$
1
0

Fig. 1. Parametrized Quantum Circuit (PQC) used in the experiments.

These per-gate energy estimates enable the estimation of the total energy consumed by a quantum circuit using the superconducting technology. With an estimate of the energy consumption per gate E_g, the number of qubits N_q, and the depth N_d of a quantum circuit, we can estimate the circuit total energy consumption E_T by

$$E_T = E_g \times N_q \times N_d. \tag{2}$$

We can experiment with a small example of the quantum circuit shown in Figure 1, comprised of 4 parameters that form two parameterized layers, each containing 2 parameters. The circuit operates on 3 qubits, involves 12 gates, and has a depth of 7. Assuming an energy consumption per gate of $\approx 0.18J$ for a superconducting quantum computing system, from Table II, and using Equation 2, the total energy consumption of the circuit in Figure 1 is $\approx 3.78J$. We conducted experiments by running this circuit on various IBM superconducting quantum simulators and devices, and we recorded their respective runtimes, as presented in Table II. The total power consumption per circuit P_{cir} is written $P_{cir} = E_T/T_{run}$, where T_{run} is the total runtime on each device, and the power consumption per gate P_g is written $P_g = P_{cir}/g$, where g is the number of gates on the circuit. The results of these computations are also shown in Table II.

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 II
POWER CONSUMPTION PER GATE FOR A QUANTUM CIRCUIT SHOWN IN FIGURE 1 ON VARIOUS IBM SIMULATORS AND DEVICES

We observe in Table II that real quantum devices (such as IBM's `ibmq_lima`, `ibmq_belem`, and `ibmq_quito`) have longer runtimes for quantum circuits than simulators (such as `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 operations which can increase the circuit's runtime. Simulators quite 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.

The longer runtime of quantum circuits on real devices compared to simulators is primarily due to the effects of noise, limited connectivity, and device resources. We also acknowledge that while simulators can provide a faster and more ideal simulation of quantum circuits, real devices provide a more accurate representation of the challenges and limitations of current quantum technology.

III. Quantum optimization

One way to capitalize on the strengths of heterogeneous hardware environments is to reformulate problems of interest as a variational optimization problem [9]. The intuition is to offload some of the subroutines to classical computers. 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, we should execute in the quantum computer only the 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 will take care of some form of pre-processing to parameterize a quantum circuit. The quantum computer will execute and perform measurements on the quantum circuit. These measurements will be post-processed and passed to classical optimizations or machine-learning (ML) algorithms. Either way, parameters are updated on every iteration to minimize a cost function [12]. As we discuss in the next few sections, working with Parametrized Quantum Circuits (PQCs) allows additional performance improvements in 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 is biased by design to highlight the reasons for energy-related execution advantages of the methods. The numerical experiments are intended to display prototypical use cases of these algorithms and a comparison with classical counterparts where appropriate. In most cases these experiments were performed using the code repository from the PennyLane framework [38] and the IBM qiskit runtime [10].

A. Quantum Natural Gradient (QN-GD)

1) Theory: Vanilla Gradient Descent (V-GD) aims to minimize a cost as a function of parameters in the Euclidean space. However, Euclidean geometry might be sub-optimal [35] and so we find in practice that different parametrizations of the cost might need 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 another one 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 [35]. 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 [36]. As in the classical case, we can fix this by multiplying the Euclidean gradient, this time with the Fubini-Study metric tensor (g^{+}), which unsurprisingly reduces to F in the classical case [37]. This generalization is known as Quantum Natural Gradient Descent (QN-GD) [26].

Besides increasing the chances of finding the true minima of the system independently of the parameterization used, the introduction of QN-GD has implications regarding to the number of quantum circuit executions and consequently to the energy spend in these calculations. First let us briefly examine how V-GD proceeds to better understand the difference with its quantum version.

V-GD and several other classic optimizations that are currently used in quantum calculations uses an approach called parameter-shift rules to evaluate the partial derivatives (i.e., the gradient). Full derivation of these rules can be found in [27]. In this paper, the authors derived several specific rules. For simplicity, here we only show an abstract general framework just to illustrate the point in circuit evaluation.

If we reduce to a case of a single unitary gate $U(\theta_i)$ that depends on a parameter θ_i, then 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 could 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 unitary conjugation above as a linear transformation $\mathcal{T}$ acting over B and differentiable in θ_i, that is:

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

Lastly, some effort has to be expended on expressing this gradient as a linear combination of the same transform $\mathcal{T}$ but in 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 general multiplier and the shift s depends on the transformation and in general does not need to be infinitesimal. Ultimately, the last equation also means that to determine this gradient we need only two quantum circuit evaluations on the shifted parameters. This is a simplified case and other parameter-shift rules require additional circuit evaluations [28].

On the other hand, 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. However, regarding quantum optimization, although the use of g^+ is theoretically justified, this tensor cannot be directly evaluated on quantum hardware and therefore its implementation relies on several approximations. Specifically, the library PennyLane [38] implements the block-diagonal approximation to estimate the Fubini-Study metric tensor [26].

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

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

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 [26]). This approximation implies that the number of circuit evaluations for QN-GD is $N_{eval} = 2 \times d + L$. As we saw before, for V-GD, this number is simply $N_{eval} = 2 \times d$

2) 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 Fig. 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. Fig. 2

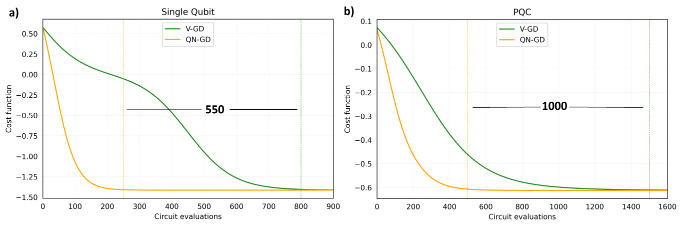


Fig. 2. Comparing optimization convergence rates between V-GD and QN-GD for: (a) a single qubit and (b) the Parametrized Quantum Circuit (PQC) from Fig. 1.

On the other hand, QN-GD was decisively superior. With respect to QN-GD, V-GD used 3.2× more circuit evaluations to optimize a single qubit and 3× more circuit evaluations to optimize the PQC from Fig. 1.

To estimate the energy-efficiency of QN-GD over V-GD we could 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 Fig. 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) that the energy and power to run these circuits only once can be directly used or derived from Tables I and II. Then to estimate the energy savings we simply do $N_{shots} \times N_{saved} \times Energy$. These give us $\approx 3.7 \times 10^3$ J ($\approx$ 6.8 kW) for the PQC and $\approx$ 100 J ($\approx$ 0.3 kW) for a single qubit (for context, a typical laptop uses $\approx$ 50W).

B. Simultaneous Perturbation Stochastic Approximation (SPSA)

1) Theory: The parameter-shift rules [27] 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, Rz). From here, the partial derivatives (and gradients) used in several quantum optimization algorithms can be directly obtained. However, exact gradients are not strictly needed to minimize a cost function. Gradient approximations can be used as well and this is the strategy of the method(s) described in this section. A way to obtain an approximate gradient is using a stochastic factor δ instead of a fixed analytical shift s. As we will see, this 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 for each element of the parameter vector at the time followed by an evaluation of the cost function; this path leads to the Kiefer-Wolfowitz Finite Difference Stochastic Approximation method (FDSA) [31], (2) the stochastic factors can be included in every element of the parameter vector at the same time and the evaluation of the cost function can be done just twice at the end. This leads to the Simultaneous Perturbation Stochastic Approximation method (SPSA) [32]. In practice, for quantum algorithms this means that for FDSA the number of evaluations of the cost function (circuit executions) scales linearly with p

while for SPSA we only have to evaluate the circuit twice independently of the size of the parameter vector θ. 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 the fraction of its computational cost. As we mentioned in the section about Quantum Natural Gradient Descent (QN-GD), 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 a 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 [33].

2) Numerical experiments: From the theory section above we learned that the number of parameters to optimize is an important part of this method, the circuits used here are slightly larger than in the previous section on QN-GD. In particular we used a construct template from the Pennylane library called *qml.StronglyEntanglingLayers* based on [34]. In the simplest case, by selecting a circuit that should run in $N_{qubits} = 4$ 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).

We also now know that 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 $N_{eval} = 2 \times N_{steps}$ independently of the number of parameters to optimize.

The different convergence rates between V-GD and SPSA considering the number of circuit executions or evaluations N_{eval} is shown in Fig. 3. For this case, the scaling difference considering the number of parameters d is certainly pronounced (120×). On the other hand, if we consider the advantage of SPSA to arrive to the same minimized value of the cost function, then this advantage is 3×. However, it is to be expected that these gains become more important as we evaluate circuits with ever larger number of trainable parameters d. All being considered, we can now estimate the energy savings from using SPSA in the 80 gate circuit used in this section. Once again we will a number of shots $N_{shots} = 1000$ for every point in Fig. 3, the number of saved steps from the figure is $N_{saved} = 400$ and we already described our circuit before as containing a number of gates $N_{gates} = 80$, which we will assume to have identical energy consumption at any position in the circuit and have energy consumption values as those reported in tables I and II for superconducting qubits. The energy we can save by using SPSA instead of V-GD to optimize our circuit is then ≈ 5.7 kJ (≈ 17.9 kW).

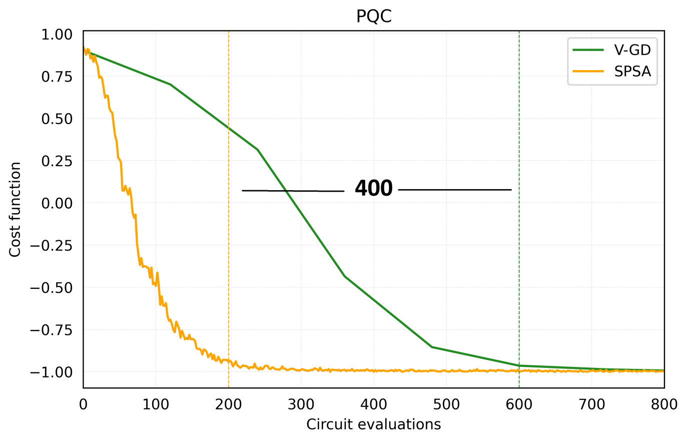


Fig. 3. Comparing optimization convergence rates between V-GD, and SPSA.

C. *Quantum circuit structure learning (Rotosolve and Rotoselect)*

1) Theory: PQC optimizers often takes advantage of parameter-shift rules [27]. This neat trick evaluates phase-shifted expectation values of the circuit and returns the gradient which is used to minimize the cost function (that can be 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 [13]. 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, 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 characterization of the sinusoidal function can be done by estimating the values of A, B and C which can be derived from sampling the expected values of the Hamiltonian at specific angles of the gate. Finally, a closed-form expression for the optimal angle (the one that minimized the Hamiltonian) can also be found:

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

This expression implies that for a selected gate we can calculate the optimal value analytically (gradient-free) using three circuit evaluations. Furthermore, as 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 we will need 3, 5, and 7 circuit evaluations respectively. The difference between Rotosolve and Rotoselect relies only in the scope of the optimization cycle. If we have fixed gates and we restrict the optimization to the parameters, the resulting algorithm is *Rotosolve*. If for every optimization step we also include the type of gate (R_x, R_y, R_z), 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.

2) Numerical experiments: we performed numerical experiments to compare both rotosolve and rotoselect with V-GD. The algorithms were tested on a toy circuit using only three qubits each running a single rotational gate at the same time ($N_{qubit} = 2$, $N_{depth} = 1$, $N_{parameters} = 2$). In Fig. 4 we display the optimization of the corresponding Hamiltonian as a function of circuit evaluations (N_{eval}).

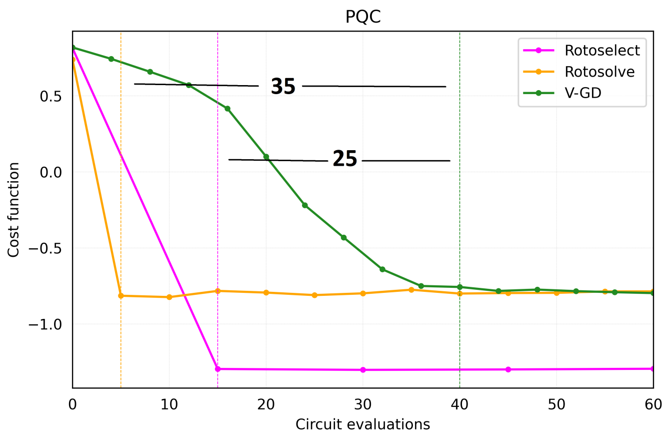


Fig. 4. Comparing optimization convergence rates between V-GD, rotosolve and rotoselect. We used a simple circuit with two parametrized gates running on 2 qubits.

As we did in previous sections, we are using parameter-shift rules [27] for V-GD, that is, we made two circuit evaluations per circuit. Therefore, V-GD for this small circuit reaches a minimum after ten optimization steps. For the cases of rotosolve and rotoselect we reach the minimum in a single step. This case, however, this means that for rotosolve one step takes seven circuit evaluations. With such a small number of optimization steps we would not have understood that both V-GD and rotosolve were stuck in a local minimum if not by the implementation of rotoselect. Indeed, rotoselect reaches a deeper minimum in the cost function by optimizing also the type of gates to be used (circuit structure learning). 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 number of circuit evaluations overall. That is, in this example, the new algorithms had 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 values is negligible but we reported for completion. Assuming as usual a number of shots $N_{shots} = 1000$ and the steps saved from using rotosolve and rotoselect are respectively 35 and 25. Then the energy savings of these algorithms with respect to V-GD are ≈ 12 J (≈ 40 W) and ≈ 9 J (≈ 28) kW, for rotosolve and rotoselect respectively.

D. Frugal shot optimization (Rosalin)

1) Theory: In quantum computing lingo, a shot is a single execution of a quantum algorithm or circuit in a quantum device. This idea has become the *de-facto* metric for pricing the rental of quantum devices and there are at least two powerful reasons why this metric is unlikely to disappear. 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 [29]. Depending on the technology, the number of physical qubits to accomplish this varies significantly. A promising solution to this problem could be the implementation of the elusive Majorana qubit for topological quantum computing [30]. However, even if a noiseless quantum computing is built, the second reason for having multiple shots for any calculation is the non-deterministic 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 though 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) [14].

Rosalin's approach to minimize the number of shots starts by breaking the problem of estimating a cost function that can be expressed as the expected value of a Hamiltonian $C = \langle H \rangle$. The underlying assumption is that our Hamiltonian can be expanded as the 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 simultaneously measured, 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 then 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 one of them. Rosalin's goal can thus be framed as follows: given a budget of shots, can we find how to redistribute them on the variables (i.e., sampling

h_i) for a maximum impact of every shot while providing an unbiased estimation of $\langle H \rangle$? As we will briefly describe below, Rosalin combines a sampling redistribution strategy with the core ideas of an adaptive optimizer named individual Coupled Adaptive Number of Shots (iCANS) [15].

Conceivably, there are several ways in which we could redistribute the shots 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. In [14], the authors describe some of these strategies in detail, but here we will just 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)*, a variable h_i will be selected for sampling with probability:

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

and finally (4) *Weighted hybrid sampling (WHS)*, a combination of (2) and (3) above. It is important to notice that the first two methods are deterministic and the last two have at least some stochastic elements. To have any hope of obtaining unbiased estimators of $\langle H \rangle$ and its variance, we should avoid using the deterministic methods by themselves. The introduction of randomness is the factor that ultimately allows us to have unbiased estimators with as little as a single shot in Rosalin. In fact, in [14], the authors used the last two strategies calling the resulting algorithms Rosalin1 and Rosalin2 respectively.

We need some additional background to understand the other main component of Rosalin, which is essentially a slight modification of iCANS. The starting point to understand 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 optimizations 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 performing 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 us select a proper learning rate. In [15], the authors 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 $\alpha \leqslant \frac{2}{L}$ [16]. Crucially, 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 we can estimate an upper bound for L as:

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

As we have hinted in other sections (see Quantum Circuit Learning), analytical solutions can also 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 we 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.

2) Numerical experiments: If to create a Hamilnonian we use, for instance, the set of rotational Pauli matrices (X, Y, Z) as the generators for our operators h_i then 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 before, it is possible to recreate sampling strategies for the number of shots on each gate. Besides this toy example, the same analysis would be more effective for complex Hamiltonians with a large number of operators h_i.

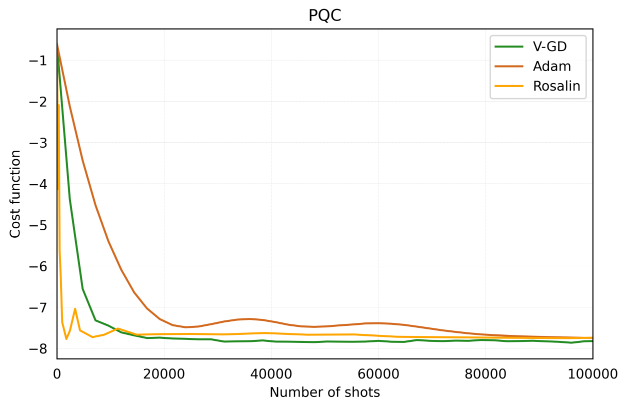


Fig. 5. Comparing optimization convergence rates between V-GD, the Adam optimizer and Rosalin.

We performed numerical experiments using a small Hamiltonian explicitly expressed using five non-commuting operators with their coefficient $c_i h_i$. The resulting circuit has two qubits and the calculations within the circuit were distributed in two layers. Following the theory described above for Rosalin, the cost function build from our Hamiltonian uses a minimal number of shots Min_{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 the recommendations for the next optimization step accordingly. 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 of using the parameter-shift rule [27] to estimate the parameter gradients, the number of shots needed for Rosalin to estimate is at least 20 times shorter than the corresponding number of shots for the Adam optimizer or V-GD. Energetically, for our small Hamiltonian, this only means a savings of 18 J ($\approx$ 56 kW). However, Rosalin's advantage should be more than 20 times for larger circuits, and the total energy savings should also 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.

E. Combinatorial optimization (FALQON/QAOA)

1) Theory: In this section we describe the basics of two optimization algorithms: the Quantum Approximate Optimization Algorithm (QAOA) [20] and the Feedback-based ALgorithm for Quantum OptimizatioN (FALQON) [21]. These algorithms have similar applications and theoretical backgrounds. Particularly, a deeper understanding of the relationship between Hamiltonians and quantum circuits is needed. 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.

Although for the general case the of a complex Hamiltonian to an efficient quantum circuit remains more an art than a science a lot of progress has been made characterizing some of the building blocks of these mappings. The success of these constructs relies in understanding that the time evolution is the link that allows 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:

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

In practice, this is an NP-hard problem because of the exponential growth of the Hilbert space with the size of U(H,t) [17]. Additionally, current quantum computers are still too noisy [18] and it is desirable to keep the circuit depth as shallow as possible. These limitations currently impose circuit generation based on approximate time evolution operators that can be derived using the Trotter-Suzuki (or Lie) formulas [19]. In the simplest forms, this *Trotterization* mechanism states that for $m \times m$ real or complex matrices A and B this formula holds:

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

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 to 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 [22], 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. Ideally, we want to create a procedure that minimizes monotonically the expected value of the cost Hamiltonian $\langle H_c \rangle = \langle \psi(t) | H_c | \psi(t) \rangle$ by choosing $\beta(t)$ 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 gives us one strictly decreasing minimization of the cost Hamiltonian by design:

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

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

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

We can see each term $U_d(\beta_k)U_c$ with $k = 1, 2, 3, ..., l$ as 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 allow us to make estimations of $\langle H_c \rangle$ that improve monotonically with the evaluations of additional layers in the quantum circuit. Therefore, FALQON does not need external (classical) optimization algorithms; 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 then looks like $U_d(\beta_k)U_c(\gamma_k)$. The optimization in QAOA, that now can be expressed as the minimization of $\langle \psi(\vec{\beta}, \vec{\gamma})|H_c|\psi(\vec{\beta}, \vec{\gamma})\rangle$, happens simultaneously by an external classic optimizer over the $2l$ parameters $\vec{\beta} = (\beta_1, \beta_2, ..., \beta_l)$ and $\vec{\gamma} = (\gamma_1, \gamma_2, ..., \gamma_l)$. 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.

2) Numerical experiments: Although both QAOA and FALQON can be applied to a variety of optimization problem, they are both 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.

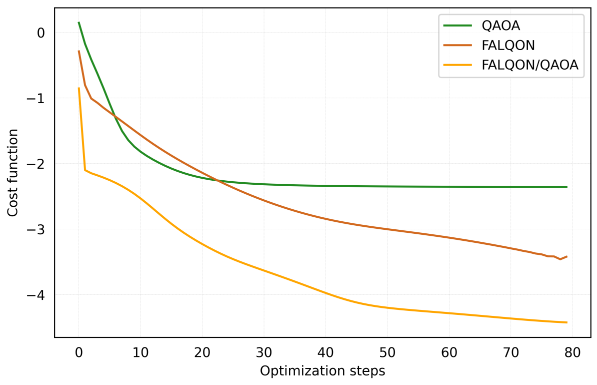


Fig. 6. Minimization of the energy cost for a MAX-CUT problem on a five node network using QAOA, FALQON and a combination of both algorithms.

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 Fig. 6 shows that it can be suboptimal 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 [27]. This is energetically convenient for small quantum circuits but as we saw (see SPSA section), it scales badly with the number of parameters.

In contrast to QAOA, FALQON seems to be relentless in the search for the global minima, but it comes at a cost. While it avoids using classical devices and optimization algorithms altogether, it relies on the execution of circuits with increasing size at every step. Therefore, the energy consumption per step increases with every circuit evaluation instead of been constant like QAOA. It is, however, energetically efficient in that FALQON avoids the communication overhead between quantum and classical devices but 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. For instance, a circuit with 25 gates performing five optimization steps will consume 22.5kJ ($\approx$ 70 kW) in QAOA and 45kJ ($\approx$ 140 kW) 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 shown in Fig. 6. That effort would include, for instance, short runs of the energy-consuming FALQON algorithm to start the optimization or whenever we get trapped in a local minima, and longer runs of QAOA to minimize the cost function at relatively low energy consumption per step. Fig. 6 shows that this strategy is likely to outperform both individual methods while compromising on energy efficiency.

F. Differentiable quantum transforms (DQTs)

1) Theory: In this section, we describe one strategy for quantum error mitigation implemented under the powerful framework of differentiable quantum transforms (DQTs) [23]. The idea of differentiation is commonplace in optimization and machine learning. DQTs provide a useful abstraction to differentiation and is implemented in the PennyLane library [23], [38]. In previous sections we have already used differentiation in the context of PQCs. That is, if we view a PQC as a quantum function or quantum program S with m parameters $\theta = (\theta_1, \theta_2, ..., \theta_m)$ then we simply say that S is differentiable if $dS/d\theta_i$ is defined for every possible value of each θ_i. The parameter-shift rules [27] that we have already used 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., PQC or $f(\theta)$) and returns another function, the derivatives ($g(\theta) = \nabla f(\theta)$). Now a transform $\mathcal{T}$ could 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 [18]. Therefore, to obtain precise results, quantum calculations will necessarily have to be backed by some form of error-mitigation techniques. Ideally, experts should have coding access to low-level hardware control. For instance, by manipulating the pulse duration on each gate, programmers could study noise scaling and re-calibrate the gates of superconducting qubit devices. However, such level of access is rarely made available to programmers; but even if it was available, programmers would still need expert knowledge of the underlying physics in those devices. In this sense the manipulation of noise at gate or circuit level and DQTs able to facilitate these studies are truly valuable today.

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) [24].

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 can 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 (e.g., or gate, layer, circuit) then we can replace U by $U(\mathbf{U}\mathbf{U}^\dagger)^n$. Because $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}$. The expected value of the calculation does not change but the number of physical operation in the device scales as $1+2n$ and the noise should scale accordingly. Further theoretical details and numerical experiments on noise scaling and extrapolation methods to achieve ZNE are developed in [24].

2) Numerical experiments: To evaluate the noise control, our simulations were performed on a Hamiltonian that represents a quantum version of the Ising model:

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

Here, the observables (Z_i, X_i) can be represented as rotational Pauli gates, while 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 Fig. 7) and estimated by optimization algorithms if the Hamiltonian is mapped to a parameterized quantum circuit.

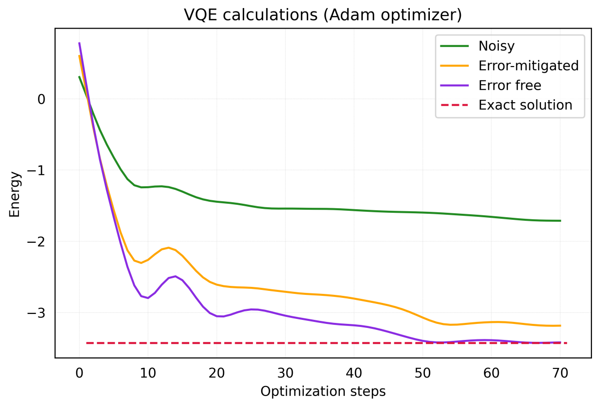


Fig. 7. Circuit optimizations highlighting the effect of the Zero Noise Extrapolation (ZNE) error mitigation technique.

Besides giving us access to ideal and noiseless quantum computing simulators, the library PennyLane [38] also allow us to create *ad hoc* noisy channels using a choice of noise types and a specified noise level. These noise channels can then be added to an ideal device to simulate noisy qubits or channels. Additionally, Pennylane has an implementation of the error mitigation technique known as Zero Noise Extrapolation (ZNE [24]) explained in the theory section above. ZNE can be then applied to our modelled noisy device to mitigate the errors during a calculation and obtain result that closely resemble those obtained on actual quantum devices.

In Fig. 7 we show runs using the classical Adam optimizer while trying to find the ground state of our Ising model. The Adam optimizer ran purely on classical devices (CPU) while the evaluation of the cost function was performed 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 only that the convergence to the exact solutions by the ideal device took 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.

IV. Discussion

In this study we explored the potential for energy-efficiency advantages in quantum optimization algorithms with respect to their classical counterparts. We also explored the impact of other techniques aimed to improve specific problems affecting quantum optimizations, such as the reduction of the number of circuit evaluations (shots) while preserving reasonable estimates of the cost Hamiltonian's expected value and the use of error mitigation techniques during these calculations.

We approached the topic by first exploring the limited literature on the energetic profile of quantum computation. We supported the findings in this section with additional estimates of energy consumption in both simulators and quantum devices. Although Table I reports the energy consumption for several technologies, in practice we only used the values corresponding to superconducting qubits since they are widespread in the platforms and libraries used in this paper (IBM quantum devices and the Pennylane library). We extended energy estimates from gates to a small quantum circuit that was executed on several simulators and devices (Table II). These experiments provided a baseline for the variability of quantum energy evaluations in a single device and also between 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 attempted to 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. Compromises between minimization of cost function and energy efficiency appear commonplace for several algorithms. For instance, QN-GD needs more circuit evaluations per step than V-GD and therefore appears disadvantaged energetically. 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 observed 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. However, We should notice 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 energetically more expensive. As the problem to be optimized grows in scale (e.g., MAX-CUT in ever-larger networks), it would be advisable to use a combination of both FALQON and QAOA to exploit both the relatively high search efficiency of FALQON and energy-efficiency of QAOA.

Finally, we performed numerical experiments to gather some practical evidence of energy efficiency using these algorithms. For the most part, our numerical experiments support the expectations behind the theoretical basis of these algorithms. Additionally, these experiments provide examples of the energy gains we can expect from the use of one algorithm over the other.

While most of the optimization algorithms explored here seem to be approximately between $2\times$ to $4\times$ more energy-efficient than their classical counterparts (except FALQON), it is general techniques like Rosalin (with $\geq 20\times$ gains) and potentially ZNE that show real promise in this regard.

In this contribution we tried to explore quantum optimization algorithms and techniques with as much methodological variety as possible. The rationale was to highlight different strategies that are being actively developed specifically for quantum computations. This was not trivial task, and we could not accommodate other algorithms such as those using specialized gates [41], techniques to escape barren plateaus during optimization [42] or using the quantum analytic descent technique to build approximate classical models of the quantum landscape [25], just to name a few.

Finally, it is also important to recognize that our exploration of these topics has been greatly facilitated by the code base, demos, papers, and development communities of both Pennylane [38] and IBM quantum [10].

V. Conclusion

In this paper we explored the implications for 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 both a toy circuit and a generic superconducting qubit. Subsequently, we explored both theoretically and numerically 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 unlike the others, does not use classical devices and has energy-consuming growing circuits on each iteration. Rosalin, a technique that can be applied to any other optimization algorithm, showed the greatest energy-efficiency potential, with at least $\geq 20\times$ more energy saved per minimization step. This improvement can be achieved by optimizing the number of shots (repetitions) needed to estimate the expected value of the cost function. Error-mitigation techniques like ZNE, while general and clearly effective, need additional study to assess their impact on energy consumption.

References

[1] Gibney, E. Quantum computer race intensifies as alternative technology gains steam. *Nature*. **587** pp. 342-343 (2020,11)

[2] Aboy, M., Minssen, T. & Kop, M. Mapping the Patent Landscape of Quantum Technologies: Patenting Trends, Innovation and Policy Implications. *IIC - International Review Of Intellectual Property And Competition Law*. **53**, 853-882 (2022,7), https://doi.org/10.1007/s40319-022-01209-3

[3] Koch, J., Yu, T., Gambetta, J., Houck, A., Schuster, D., Majer, J., Blais, A., Devoret, M., Girvin, S. & Schoelkopf, R. Charge-insensitive qubit design derived from the Cooper pair box. *Phys. Rev. A*. **76**, 042319 (2007,10), https://link.aps.org/doi/10.1103/PhysRevA.76.042319

[4] Childress, L. & Hanson, R. Diamond NV centers for quantum computing and quantum networks. *MRS Bulletin*. **38**, 134-138 (2013)

[5] Shumiya, N., Hossain, M.S., Yin, JX. et al. Evidence of a room-temperature quantum spin Hall edge state in a higher-order topological insulator. *Nat. Mater.*. **21**, 1111–1115 (2022), https://doi.org/10.1038/s41563-022-01304-3

[6] Grossi, M., Crippa, L., Aita, A., Bartoli, G., Sammarco, V., Picca, E., Said, N., Tramonto, F. & Mattei, F. A Serverless Cloud Integration For Quantum Computing. (arXiv,2021), https://arxiv.org/abs/2107.02007

[7] Bourassa, J., Alexander, R., Vasmer, M., Patil, A., Tzitrin, I., Matsuura, T., Su, D., Baragiola, B., Guha, S., Dauphinais, G., Sabapathy, K., Menicucci, N. & Dhand, I. Blueprint for a Scalable Photonic Fault-Tolerant Quantum Computer. *Quantum*. **5** pp. 392 (2021,2), https://doi.org/10.22331/q-2021-02-04-392

[8] Bravyi, S., Dial, O., Gambetta, J., Gil, D. & Nazario, Z. The future of quantum computing with superconducting qubits. *Journal Of Applied Physics*. **132**, 160902 (2022),https://doi.org/10.1063/5.0082975

[9] Moll, N., Barkoutsos, P., Bishop, L., Chow, J., Cross, A., Egger, D., Filipp, S., Fuhrer, A., Gambetta, J., Ganzhorn, M., Kandala, A., Mezzacapo, A., Müller, P., Riess, W., Salis, G., Smolin, J., Tavernelli, I. & Temme, K. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science And Technology*. **3**, 030503 (2018,6), https://dx.doi.org/10.1088/2058-9565/aab822

[10] Qiskit contributors Qiskit: An Open-source Framework for Quantum Computing. (2023), 10.5281/zenodo.2573505

[11] IBM-Quantum Qiskit-extensions/quantum-serverless: A programming model for leveraging quantum and classical resources. *GitHub*., https://github.com/Qiskit-Extensions/quantum-serverless

[12] Benedetti, M., Lloyd, E., Sack, S. & Fiorentini, M. Parameterized quantum circuits as machine learning models. *Quantum Science And Technology*. **4**, 043001 (2019,11), https://doi.org/10.1088/2058-9565/ab4eb5

[13] Ostaszewski, M., Grant, E. & Benedetti, M. Structure optimization for parameterized quantum circuits. *Quantum*. **5** pp. 391 (2021,1), https://doi.org/10.22331/q-2021-01-28-391

[14] Arrasmith, A., Cincio, L., Somma, R. & Coles, P. Operator Sampling for Shot-frugal Optimization in Variational Algorithms. (2020) https://doi.org/10.48550/arXiv.2004.06252

[15] Kübler J.M., Arrasmith, A., Cincio, L., & Coles, P. An Adaptive Optimizer for Measurement-Frugal Variational Algorithms. *Quantum*. **4** 263 (2020) https://doi.org/10.22331/q-2020-05-11-263

[16] Balles, L., Romero, J. & Hennig, P. Coupling Adaptive Batch Sizes with Learning Rates. *Thirty-Third Conference on Uncertainty in Artificial Intelligence (UAI)* (2017), https://doi.org/10.48550/arXiv.1612.05086

[17] Khaneja, N. & Glaser, S. Cartan Decomposition of SU(2^n), Constructive Controllability of Spin systems and Universal Quantum Computing. (2000), https://doi.org/10.48550/arXiv.quant-ph/0010100

[18] Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum*4 79 (2018), https://doi.org/10.22331/q-2018-08-06-79

[19] Dhand, I. & Sanders, B. Stability of the Trotter–Suzuki decomposition. *Journal Of Physics A: Mathematical And Theoretical*. **47**, 265206 (2014,6),https://doi.org/10.1088/1751-8113/47/26/265206

[20] Farhi, E., Goldstone, J. & Gutmann, S. A Quantum Approximate Optimization Algorithm. (2014), https://doi.org/10.48550/arXiv.1411.4028

[21] Magann, A., Rudinger, K., Grace, M. & Sarovar, M. Feedback-Based Quantum Optimization. *Physical Review Letters*. **129** (2022,12), https://doi.org/10.1103/PhysRevLett.129.250502

[22] Cong S., Meng, F.A Survey of Quantum Lyapunov Control Methods. *The Scientific World Journa*. Article ID 967529 (2013), http://dx.doi.org/10.1155/2013/967529

[23] Matteo, O., Izaac, J., Bromley, T., Hayes, A., Lee, C., Schuld, M., Száva, A., Roberts, C. & Killoran, N. Quantum computing with differentiable quantum transforms. (2022), https://doi.org/10.48550/arXiv.2202.13414

[24] Giurgica-Tiron, T., Hindy, Y., LaRose, R., Mari, A. & Zeng, W. Digital zero noise extrapolation for quantum error mitigation. *2020 IEEE International Conference On Quantum Computing And Engineering (QCE)*. pp. 306-316 (2020), https://doi.org/10.1109/QCE49297.2020.00045

[25] Koczor, B. & Benjamin, S. Quantum analytic descent. *Physical Review Research*. **4** (2022,4),https://doi.org/10.1103/PhysRevResearch.4.023017

[26] Stokes, J., Izaac, J., Killoran, N., Carleo, G. Quantum Natural Gradient. *Quantum*. **4** pp. 269 (2020,5), https://doi.org/10.22331/q-2020-05-25-269

[27] Schuld, M., Bergholm, V., Gogolin, C., Izaac, J. & Killoran, N. Evaluating analytic gradients on quantum hardware. *Physical Review A*. **99** (2019,3),https://doi.org/10.1103/PhysRevA.99.032331

[28] g, D., Izaac, J. Wang C. & Yen-Yu Lin C. General parameter-shift rules for quantum gradients. *Quantum*. **6**, 677 (2022),https://doi.org/10.22331/q-2022-03-30-677

[29] Roffe, J. Quantum error correction: an introductory guide. *Contemporary Physics*. **60**, 226-245 (2019,7), https://doi.org/10.1080/00107514.2019.1667078

[30] Aguado, R., Kouwenhoven L. Majorana qubits for topological quantum computing *Physics Today*. **73**, 44 (2020,6), https://doi.org/10.1063/PT.3.4499

[31] Kiefer, J., Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*. **23(3), 462–466.** (1952),http://www.jstor.org/stable/2236690

[32] Spall, J. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions On Automatic Control*. **37**, 332-341 (1992)

[33] Gacon, J., Zoufal, C., Carleo, G. & Woerner, S. Simultaneous Perturbation Stochastic Approximation of the Quantum Fisher Information. *Quantum*. **5** pp. 567 (2021,10), https://doi.org/10.22331/q-2021-10-20-567

[34] Schuld, M., Bocharov, A., Svore, K. & Wiebe, N. Circuit-centric quantum classifiers. *Physical Review A*. **101** (2020,3),https://doi.org/10.1103/PhysRevA.101.032308

[35] Amari, S. Natural Gradient Works Efficiently in Learning. *Neural Computation*. **10**, 251-276 (1998,2), https://doi.org/10.1162/089976698300017746

[36] Yamamoto, N. On the natural gradient for variational quantum eigensolver. (arXiv,2019), https://arxiv.org/abs/1909.05074

[37] Mondal, D. Generalized Fubini-Study Metric and Fisher Information Metric. (arXiv,2015), https://arxiv.org/abs/1503.04146

[38] Bergholm et al. PennyLane: Automatic differentiation of hybrid quantum-classical computations. (arXiv,2018), https://arxiv.org/abs/1811.04968

[39] Fellous-Asiani, M. The resource cost of large scale quantum computing. (arXiv,2021), https://arxiv.org/abs/2112.04022

[40] Jaschke, D. and Montangero, S. Is quantum computing green? An estimate for an energy-efficiency quantum advantage. *Quantum Science and Technology*. (2022)

[41] Wiersema, R., Lewis, D., Wierichs, D., Carrasquilla, J. & Killoran, N. Here comes the SU(N): multivariate quantum gates and gradients. (2023), https://doi.org/10.48550/arXiv.2303.11355

[42] Grant, E., Wossnig, L., Ostaszewski, M. & Benedetti, M. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*. **3** pp. 214 (2019,12), https://doi.org/10.22331/q-2019-12-09-214