Algorithm selection and configuration for Noisy Intermediate Scale Quantum methods for industrial applications

Moussa, C.

Algorithm selection and configuration for Noisy Intermediate Scale Quantum methods for industrial applications

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op woensdag 11 oktober 2023
klokke 10:00 uur

door

Charles Moussa
geboren te Saint-Claude, Guadeloupe, France
in 1993

Promotores:
Dr. V. Dunjko
Prof.dr. T.H.W. Bäck

Promotiecomissie:
Prof.dr. A. Plaat
Prof.dr. M.M. Bonsangue
Prof.dr. H. Trautman (University of Münster - University of Twente)
Dr. H. Wang
Dr. M. Möller (Delft University of Technology)

To my dear family and friends.

"Great things are done by a series of small things brought together."

Vincent Van Gogh, 1882

Contents

Contents

Chapter 1

Introduction

1.1 Background

The field of quantum computing has gotten increased attention as a different paradigm than classical computing for solving complex problems [142]. Quantum computers are defined as computers relying on principles from quantum mechanics to perform computation, and many implementations are being proposed. However, currently available quantum computers are very unstable, noisy, and hard to scale up. Hence, quantum algorithms that could run on current noisy hardware, in the so-called noisy intermediate-scale quantum (NISQ) era [156], which are also relevant for industrial applications, are the subject of many research topics.

The quantum community has come up with a large number of proposals for how relevant problems could be tackled with such devices. One of the main directions is the usage of hybrid quantum-classical schemes [37, 55, 40, 18, 134, 57]. A part of the computation is then carried out on a quantum computer and the classical device handles the rest. Such hybrid algorithms are most often employed as *heuristics*, i.e. approaches to solving a problem of interest whose proposed solution is not guaranteed to be optimal. Thus it is not clear whether they genuinely outperform current classical state-of-the-art algorithms in relevant domains of application. Additionally, such hybrid algorithms come with many internal components (i.e. the classical optimization algorithm used, many hyperparameters of a quantum neural network, etc.) or can be extended with new ones, which can affect their performances. Hence, they should be chosen well for achieving optimal performance.

Consequently, we will face the issues of *algorithm selection and configuration* [89].

Given many instances of a problem, the algorithm selection problem concerns selecting the best-performing algorithm for a given instance. The configuration problem is about finding how to set well the internal component of an algorithm in general. For instance, finding the best-performing classical optimization algorithm for a hybrid quantum-classical algorithm is a configuration problem.

In this thesis, we tackle two main research questions related to *algorithm selection and configuration* for hybrid quantum-classical algorithms:

> **RQ1** The first one is about understanding when a hybrid quantum-classical algorithm can be used against a classical counterpart.
>
> **RQ2** The second research question is about identifying the key internal components of hybrid quantum-classical algorithms and how to set them well.

We perform many algorithm selection and configuration studies on several common NISQ algorithms that can be applied to industrial problems from different fields. We contribute to understanding and improving their performances, toward helping the design of better hybrid quantum-classical algorithms for practice. Our contributions are summarized in more detail in the next section.

1.2 Contributions

The work described in this thesis has been performed within a collaboration between Leiden University and TotalEnergies which funded this Ph.D. The Ph.D. started with the aim to study the application in relevant domains of a few hybrid quantum-classical algorithms commonly found in the quantum literature named *variational quantum algorithms* (VQAs) [40], which will be presented in the next chapter. VQAs have many possible applications, from the more natural application of simulating quantum systems [134, 166, 71] (primarily to chemistry use cases) to optimization use cases [57, 201, 79] and machine learning [18]. These applications are all relevant to industrial problems in the energy sector.

The new hybrid quantum-classical algorithms we study are evolving to improve their performances toward becoming more practical and competing with classical counterparts. Hence, to achieve these goals, they are going through empirical studies and domain-specific enhancements. As research questions, we are interested in developing new methods and identifying key components to improve the design of hybrid quantum-classical algorithms relevant to applications in the above-mentioned fields.

In this thesis, we study several common VQAs from the literature and contribute to the issues of *algorithm selection and configuration* VQAs face. We do so through many empirical studies and with the usage of many benchmarking techniques. The dissertation is outlined as follows:

- Chapter 2 introduces the basic concepts of quantum computing and VQAs. We also review different VQA workflows related to quantum applications with a focus on applications related to the energy sector. The applications which we are interested in belong to the fields of chemistry, combinatorial optimization, and machine learning.

- In Chapter 3, we introduce ideas from the algorithm selection domain to a quantum algorithm designed to tackle combinatorial optimization problems. The latter is called Quantum Approximate Optimization Algorithm (QAOA) and is benchmarked against a classical counterpart. The contents of this chapter are (partly) based on [135] (item 1 in the next list of the author's contribution).

- Chapter 4 demonstrates an example of algorithm configuration problem with the usage of unsupervised machine learning techniques to set the internal parameters of QAOA. The contents of this chapter are (partly) based on [138] (item 4 in the next list of the author's contribution).

- In Chapter 5, we present a hybrid quantum-classical algorithm combining QAOA and a classical heuristic named tabu search. Such a combination allows us to tackle larger problems than the number of qubits would allow running only a quantum algorithm. The contents of this chapter are (partly) based on [139] (item 3 in the next list of the author's contribution).

- For chemistry and material science applications, we identify the best-performing classical optimization algorithm for VQAs. A benchmarking of classical optimization algorithms is carried out in Chapter 6. The contents of this chapter are (partly) based on [24] (item 2 in the next list of the author's contribution).

- VQAs for machine learning have many hyperparameters and their importance is assessed in Chapter 7. We apply the functional ANOVA framework for hyperparameter importance on VQAs for machine learning, namely quantum neural networks. The contents of this chapter are (partly) based on [137] (item 5 in the next list of the author's contribution).

- Finally, Chapter 8 concludes with the design of a resource frugal optimizer for quantum machine learning tasks. Benchmarking of this optimizer against previous existing frugal methods is performed to demonstrate better performances for similar resource counts. The contents of this chapter are (partly) based on [136] (item 6 in the next list of the author's contribution).

1.3 Author's Contributions

Below we list publications the author has contributed to and presented in this thesis in chronological order:

1. Charles Moussa, Henri Calandra, and Vedran Dunjko. To quantum or not to quantum: towards algorithm selection in near-term quantum optimization. *Quantum Science and Technology*, 5(4):044009, 2020.

2. Xavier Bonet-Monroig, Hao Wang, Diederick Vermetten, Bruno Senjean, Charles Moussa, Thomas Bäck, Vedran Dunjko, and Thomas E. O'Brien. Performance comparison of optimization methods on variational quantum algorithms. *Phys. Rev. A*, 107:032407, Mar 20232023.

3. Charles Moussa, Hao Wang, Henri Calandra, Thomas Bäck, and Vedran Dunjko. Tabu-driven quantum neighborhood samplers. In Christine Zarges and Sébastien Verel, editors, *Evolutionary Computation in Combinatorial Optimization, Evocop 2021*, pages 100–119, Cham, 2021. Springer International Publishing.

4. Charles Moussa, Hao Wang, Thomas Bäck, and Vedran Dunjko. Unsupervised strategies for identifying optimal parameters in quantum approximate optimization algorithm. *EPJ Quantum Technology*, 9(1), 2022.

5. Charles Moussa, Jan N. van Rijn, Thomas Bäck, and Vedran Dunjko. Hyperparameter importance of quantum neural networks across small datasets. In Poncelet Pascal and Dino Ienco, editors, *Discovery Science*, pages 32–46, Cham, 2022. Springer Nature Switzerland.

6. Charles Moussa, Max Hunter Gordon, Michal Baczyk, M. Cerezo, Lukasz Cincio, and Patrick J. Coles. Resource frugal optimizer for quantum machine learning. *arXiv:2211.04965*, 2022 (submitted to journal Quantum Science and Technology).

Chapter 2

Background

This chapter serves as a brief introduction to the basic concepts in quantum computing with the gate-based model of computation. We continue by presenting the general workflow of variational quantum algorithms (VQAs). Finally, we present three domains of applications of these algorithms relevant to the energy sector: chemistry, combinatorial optimization, and machine learning. The aim of this chapter is to familiarize the reader with the necessary background, especially on VQA applications, but not to offer an exhaustive overview. Such VQAs and their specificities for applications are then studied in the next chapters of this thesis.

2.1 Quantum computing basics: the gate-based model

In contrast to classical computing which operates by manipulating bits, computation is carried out by the manipulation of quantum bits or qubits. A quantum computer is made of physical systems acting as qubits. Just like a bit, a qubit is an abstraction and can be any quantum object which has two states such as an electron or a photon. A classical bit has two distinct configurations or states representing values 0 and 1. A qubit can be in two states representing 0 and 1 values but differ from classical bits by its possibility to be in a *superposition* of those two states. The state of a qubit can change by applying quantum operations which will constitute *quantum computation*. In the conventional Copenhagen interpretation of quantum mechanics [59], by applying a measurement operation, a *superposition* state *collapses* to states representing for

instance the classical 0 and 1 values. Hence, we can obtain either 0 or 1 with their respective probabilities p_0, p_1 summing to 1.

The framework of quantum mechanics provides a mathematical and conceptual formulation for the description of quantum systems. The framework is expressed in the language of linear algebra and comes with postulates, providing a fundamental theory to study such systems. Firstly, we will enumerate the four postulates as presented in [143] and provide along more explanations. We then connect them to the so-called gate-based model, a representation of *computation* in a quantum computer.

2.1.1 Postulates of quantum mechanics

The first postulate, the state vector representation, is as follows:

Postulate 1. *Associated to any isolated physical system is a complex vector space with an inner product (that is, a Hilbert space) known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space.*

From this postulate, we understand that we can represent the state of a qubit (the isolated physical system) by a 2-dimensional complex vector with two components, $c_0 \in \mathbb{C}$ and $c_1 \in \mathbb{C}$, subject to the constraint $|c_0|^2 + |c_1|^2 = 1$. This can be generalized to a system of n qubits, represented by a 2^n-dimensional complex vector in the Hilbert space $\mathcal{H} = \mathbb{C}^{2^n}$. Hence, the description of such systems on classical computers grows exponentially with the number of qubits.

In quantum computing, the bra-ket notation/convention is used. The vector describing the state of the system is denoted $|\psi\rangle \in \mathcal{H}$, their conjugate transpose $\langle\psi| = |\psi\rangle^\dagger$ and inner-products $\langle\psi|\psi'\rangle$ in $\mathcal{H}$, and is of unit norm $\langle\psi|\psi\rangle = 1$. Such a state can change with time, leading to the second postulate describing this *evolution*:

Postulate 2. *The evolution of a* closed *quantum system is described by a* unitary transformation. *That is, the state of the system at time t_1 is related to the state $|\psi\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2, $|\psi\rangle' = U|\psi\rangle$.*

Note that *closed* quantum system here refers to no interaction of the quantum system with other systems. The quantum state is modified with unitary operations U acting on $\mathcal{H}$. This change can be also thought of as the realization of *computation* in a quantum computer. We will see in the next subsection that such a computation can be represented by a *quantum circuit*.

Eventually, a quantum system can be *observed* or *measured* (by action or interaction of an experimentalist and/or experimental equipment), making them no longer *closed*. We previously mentioned that states *collapse* and this leads to the third postulate on *measurement*:

Postulate 3. *Quantum measurements are described by a collection of measurement operators* $\{P_m\}$*. These are linear operators acting on the state space of the system being measured. The index* m *refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is* $|\psi\rangle$ *immediately before the measurement then the probability that result* m *occurs is given by* $p(m) = p_m = \langle\psi| P_m^\dagger P_m |\psi\rangle$*. The measurement operators satisfy the completeness equation:* $\sum_m P_m^\dagger P_m = I$*.*

This postulate can also be rewritten with a special case of measurements known as projective measurements as they are orthogonal projectors, i.e. $\{P_m\}$ are Hermitian and $P_{m'} P_m = \delta_{m,m'} P_m$ where $\delta_{m,m'} = 1$ if $m = m'$ and 0 otherwise. A projective measurement is described by a Hermitian operator O called an observable. Its spectral decomposition $O = \sum_m \lambda_m P_m$ in terms of eigenvalues λ_m and orthogonal projections P_m defines the outcomes of this measurement: a measured state $|\psi\rangle$ gives the outcome λ_m and gets projected onto the state $P_m |\psi\rangle / \sqrt{p(m)}$ with probability $p(m) = \langle\psi| P_m |\psi\rangle = \langle P_m \rangle_\psi$. The expectation value of the observable O with respect to $|\psi\rangle$ is $\mathbb{E}_\psi[O] = \sum_m p(m)\lambda_m = \langle O \rangle_\psi$.

As mentioned above, for a qubit, a measurement operation changes its state to states representing the classical $m = 0$ and $m = 1$ values. The probabilities of the $0, 1$ outcomes are obtained by using the square of the amplitude on the components of the state vector when the state is expressed in the eigenbasis of the observable. Hence, for $|\psi\rangle = \binom{c_0}{c_1}$, we get $p_0 = |c_0|^2$ and $p_1 = |c_1|^2$. The special cases $|0\rangle = (1, 0)^T, |1\rangle = (0, 1)^T$ (T denotes the complex conjugate transpose for real numbers) are called the computational basis states of the qubit state and correspond to obtaining the $0, 1$ outcome with probability 1.

For systems made up of n qubits, the possible outcome by measurement (with respect to the computational basis) is then one of the 2^n possible bitstring values. Such *composite quantum systems* made up of $n \geq 2$ distinct physical systems are described by the last postulate:

Postulate 4. *The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. In particular, if we have systems numbered* 1 *through* n*, and system number* i *is prepared in the state* $|\psi_i\rangle$*, then the joint state of the total system is* $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$*.*

Hence, when dealing with many qubits, the state of the system is obtained by the tensor product of the state of each individual qubit, The tensor product of single-qubit computational basis states (given by $|0\rangle = (1,0)^T, |1\rangle = (0,1)^T$) describes general computational basis states, e.g., $|10\rangle = |1\rangle \otimes |0\rangle = (0,0,1,0)^T$. Hence, each of the 2^n possible bitstring values is associated with a basis state.

We refer to [143] for more basic concepts of quantum computing and introduce the gate-based model of computation in the next subsection.

2.1.2 Quantum circuits

Similar to classical circuits used for classical computing, one model of computation used to describe quantum computation is the model of quantum circuits (see Fig. 2.1 for an example). Considering a system of n qubits (represented by wires in Fig. 2.1), unitary operations referred to as *quantum gates* are applied to the system, making the circuit a unitary operation. Mathematically, the new state is obtained with $U|\psi\rangle$ (as seen with the third postulate on *evolution*) where U is the unitary operation associated with the circuit. When a circuit U acts non-trivially only on a subset $S \subseteq [n]$ of qubits where $[n] = \{1, \cdots, n\}$, we denote such operation $U \otimes \mathbb{1}_{[n]\setminus S}$.

Common gates applied on a single qubit include the Hadamard gate H, the single-qubit Pauli gates X, Z, Y, and their associated rotations R_X, R_Y, R_Z:

$$\begin{gathered} H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, R_Z(w) = \exp\Big(-i\frac{w}{2}Z\Big), \\ Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, R_Y(w) = \exp\Big(-i\frac{w}{2}Y\Big), X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, R_X(w) = \exp\Big(-i\frac{w}{2}X\Big), \end{gathered} \tag{2.1}$$

The rotation angles are denoted $w \in \mathbb{R}$. We also have gates acting on 2 qubits such as the 2-qubit controlled-Z gate $\bullet\!\!-\!\!\bullet = \mathrm{diag}(1,1,1,-1)$ and the CNOT gate given by the matrix

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.2}$$

Measurements specified by observables are carried out at the end of a quantum circuit to obtain bitstrings.

It is worth mentioning that any quantum computation can be expressed exactly by a quantum circuit using only single-qubit and two-qubit gates (in particular the

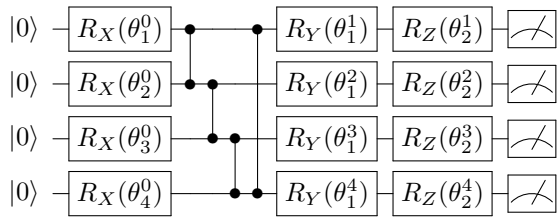


Figure 2.1: An example of quantum circuit with 4 qubits. Rotations are applied with arbitrary angles denoted by θ symbols. A layer of R_X rotations is first applied, followed by controlled-Z gates and R_Y, R_Z rotations. Finally, measurements are carried out to get a bitstring as an outcome.

CNOT or any entangling gate). We again refer to [143] for more details.

There exist many physical systems for designing quantum computers (superconducting, ion traps, neutral atoms...). However, they are very limited as their constructions do not scale well (in terms of the number of qubits) and the computation is heavily affected by (environmental) noise. Hence we do not have perfect (fault-tolerant, unaffected by noise) quantum computers. Thus, running large quantum circuits is impractical as, due to the increased effect of noise the longer the circuit and the greater the number of qubits, the calculations become less accurate and reliable up to the point of being useless. Hence, to tackle applications in the NISQ era, approaches based on *parameterized quantum circuits* were proposed, which can be used to construct resource-limited quantum computation, and are at the heart of many research studies.

2.2 Parameterized quantum circuits - Variational quantum algorithms

A parameterized quantum circuit (PQC - also called an *ansatz*) [18] is a quantum circuit with adjustable real-valued parameters $\boldsymbol{\theta}$. The latter is denoted by a unitary $U(\boldsymbol{\theta})$ that acts on a fixed n-qubit state (e.g., $|0^{\otimes n}\rangle$). The ansatz can be problem dependent, or generic (problem-independent). *Hardware-efficient Ansätze* [96] are examples of the latter and the corresponding circuit can be made shallow enough to run on limited hardware. PQCs have been widely used to tackle applications in different fields such as chemistry [134], optimization [57], and machine learning [18].

2.2. Parameterized quantum circuits - Variational quantum algorithms

PQCs can be trained or optimized to tackle a problem of interest by specifying a cost function. The optimization of the parameters within a PQC is generally done within a loop of calls between a classical optimization algorithm and the quantum computer. The output from a quantum circuit is processed by the classical optimization algorithm. The latter then proposes new parameters, defining a new quantum circuit. The optimization loop is carried out until a set of stopping criteria is met (maximum number of calls exceeded, time-out, etc). Such a hybrid quantum-classical procedure is common in research motivated by the near-term constraints of current quantum computers in the NISQ era [156]. The general term used in the quantum computing community of such hybrid approaches is *variational quantum algorithms* (VQA). Fig. 2.2 illustrates such a workflow.

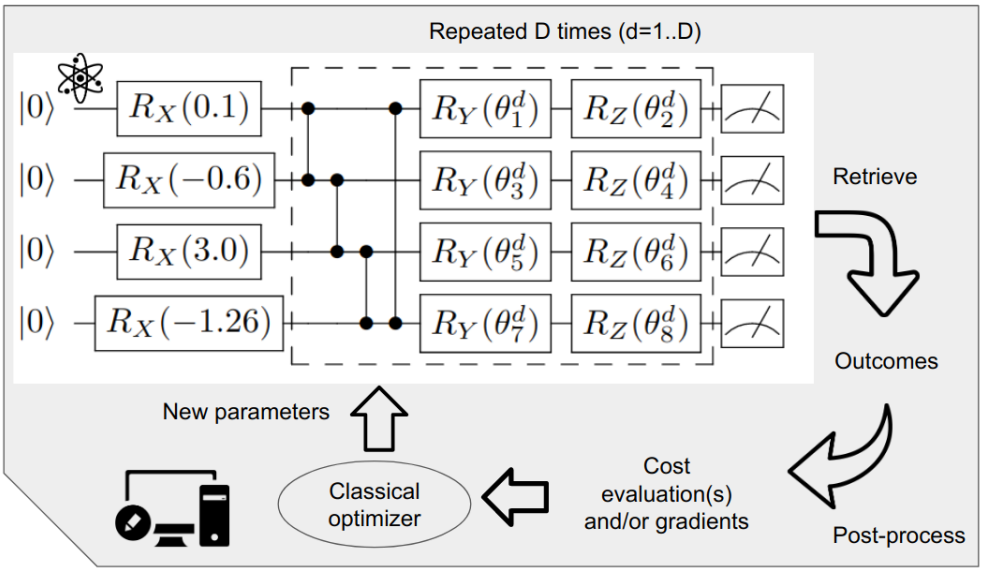


Figure 2.2: Diagram of a variational quantum algorithm. The user submits the algorithm where a quantum computer and a classical one work together. The user defines a quantum circuit with a set of parameters θ_f^d, $f = 1 \cdots 8$, $d = 1 \cdots D$. The dashed part corresponds to repeating the quantum operation a user-specified D times with different parameter values. Outcomes are retrieved from running a quantum circuit on a quantum computer. The outcomes are then converted into cost evaluation(s) and/or gradients. Such information is then used by an optimizer to get new parameters. The procedure is then repeated until convergence.

VQAs can be adapted to tackle different problems of interest in various domains or

industries. In the next section, we will review several examples of VQA applications in each of the previously-mentioned domains of chemistry, combinatorial optimization, and machine learning.

2.3 Ground state problems in quantum chemistry

In quantum computing, one of the problems from the chemistry domain is to prepare the ground state of a given molecule. The latter is specified by a molecular Hamiltonian denoted O and the task boils down to finding a PQC and its associated parameters approximating the ground state. The latter is a state vector that represents the most stable configuration of the system - the lowest energy state. Algorithms that find this ground state energy using variational approaches are called variational quantum eigensolvers (VQEs) [134]. The VQE approach has been applied to problems in the energy sector. For instance, at TotalEnergies, the problem of CO_2 adsorption in Al-fumarate metal-organic frameworks was tackled with such an approach [71].

The unitary representation of the PQC where the parameters $\boldsymbol{\theta} \in [0, 2\pi]^d$ will be modified in the process of the VQE algorithm. Starting from an initial state $|\Phi\rangle$, we get the final state $|\Psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})\,|\Phi\rangle$. The cost function to optimize is then:

$$\mathcal{C}(\boldsymbol{\theta}) = \langle O \rangle = \langle \Psi(\boldsymbol{\theta}) | O | \Psi(\boldsymbol{\theta}) \rangle.$$

By the variational principle, such cost is assured to always be greater than or equal to the ground state energy for all parameter values. In order to measure the expectation value of O on a quantum computer, it is typical to write O as a linear combination of easy-to-measure operators, i.e., the Pauli operators $\hat{P}_i \in \{\mathbb{I}, X, Y, Z\}^{\otimes N}$, which is always possible thanks to mappings from fermionic to qubit systems such as Jordan-Wigner's [144, 115]. By linearity of the expectation operator, we get:

$$O = \sum_i c_i \hat{P}_i \implies \mathcal{C}(\boldsymbol{\theta}) = \langle O \rangle = \sum_i c_i \langle \hat{P}_i \rangle. \tag{2.3}$$

The same VQA workflow illustrated in Fig. 2.2 applies to VQE. The only difference is that many circuits may be run to evaluate the cost $\mathcal{C}(\boldsymbol{\theta})$ (at most the number of Pauli-based operators to measure). Indeed, each Pauli-based operator $\hat{P}_i$ may require measuring the circuit differently (as with a different basis or projection), which is typically done by adding an extra layer of gates before measurement. The weighted sum of each expectation value is then computed.

One of the main research directions of VQEs is the performance and suitability of the classical optimizer for such tasks. Given that we estimate the cost value by many measurements, the question of the accuracy an optimizer can achieve arises. As part of the research reported in this thesis, we tackled these directions of benchmarking different optimizers for the task of finding the approximate ground-state energy of several chemistry and material science problems.

2.4 Combinatorial Optimization

In this section, we give the necessary background on another application of quantum computing: combinatorial optimization (CO). Firstly, we present an important formulation of CO problems when tackling them with a quantum computer. Then, we give a few examples of CO problems having industrial applications. Finally, we present a well-known VQA designed to tackle CO problems.

In CO, we optimize an objective function defined over discrete variables. Many combinatorial optimization problems can be expressed by a quadratic unconstrained binary optimization (QUBO) formulation [106, 204]. It is specified by the optimization problem $\min_{x\in\{0,1\}^n} \sum_{i\leq j} x_i Q_{ij} x_j$ where n is the dimensionality of the problem and $Q \in \mathbb{R}^{n\times n}$. This formulation is connected to the task of finding configurations of binary labels $\{1, -1\}$ (also called ground states) minimizing the energy of spin or Ising Hamiltonians, commonly tackled in statistical physics and quantum computing, i.e.:

$$\min_{s\in\{-1,1\}^n} \sum_i h_i s_i + \sum_{j>i} J_{ij} s_i s_j, \tag{2.4}$$

where $h_i \in \mathbb{R}$ are the biases and $J_{ij} \in \mathbb{R}$ the interactions between spins. The solutions from Ising to QUBO are connected by a linear transformation: $x_i = \frac{1+s_i}{2}$. Many industrial applications have been tackled using this formulation on quantum hardware [204].

Let us mention a few examples of CO problems falling under the QUBO formulation and which have numerous applications to industrial problems. A well-known example is the traveling salesperson problem (TSP): given N different cities and their distances, find the shortest possible path that visits each city exactly once. For the energy sector, we can take as an example of TSP application the optimization of delivery routes. The MaxCut problem is another one, with applications such as network design or data clustering [155]. Consider a graph G over a vertex set V, and edge set E, with

N vertices. The problem is to find a subset $S \subset V$ such that the number of edges between S and $V \setminus S$ is maximized. The set of edges between S and $V \setminus S$ is called a *cut*, and we will denote the size of the maximal cut with C_{max}. The problem naturally extends to weighted graphs, where the objective is to maximize the total weight of the edges of the cut. Note that it is also possible to reformulate a QUBO of n variables into a weighted MaxCut of $n+1$ variables. Finally, in graph coloring, we aim to color a given graph such that no two adjacent vertices are of the same color. As an example of an application, we can mention timetable/resource scheduling. All the previous problems are $\mathcal{NP}$-complete, and thus mutually reducible.

In terms of quantum algorithms, one approach stands out using the gate-based model to tackle CO problems: the Quantum Approximate Optimization Algorithm (QAOA) [57]. It constitutes one of the often-mentioned candidates expected to yield a quantum boost in the era of near-term quantum computing [58]. Firstly, the classical cost function given in Eq. 2.4 is encoded in a quantum Hamiltonian defined on N qubits by replacing each variable s_i in Eq. 2.4 by the single-qubit operator σ_i^z (which has the same matrix representation as the Z gate in Eq.2.1 but we write so to maintain consistency with standard notation in different literature), for example:

$$H_C = \sum_i h_i \sigma_i^z + \sum_{j>i} J_{ij} \sigma_i^z \sigma_j^z. \tag{2.5}$$

Here, H_C corresponds to the target Hamiltonian, and any bitstring corresponding to the ground state of H_C also minimizes the cost function. Secondly, a so-called *mixer Hamiltonian* $H_B = \sum_{j=1}^{N} \sigma_j^x$ is used during the procedure. σ_j^x is the same operator as the X gate in Eq.2.1. These Hamiltonians are then used to build a layer of a quantum circuit with real parameters [1]. This circuit is initialized in the $|+\rangle^{\otimes N}$ state, corresponding to all bitstrings in superposition with equal probability of being measured. Then, applying the layer p times sequentially yields the following quantum state:

$$|\gamma, \beta\rangle = e^{-i\beta_p H_B} e^{-i\gamma_p H_C} \cdots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |+\rangle^{\otimes N},$$

defined by $2p$ real parameters $\gamma_i, \beta_i, i = 1...p$ or *QAOA angles* as they correspond to angles of parameterized quantum gates. An example of depth 1 QAOA circuit is provided in Fig. 2.3. The output state corresponds to a probability distribution over all possible bitstrings. The classical optimization challenge of QAOA is to identify the sequence of parameters γ and β so as to minimize the expected value of the cost func-

[1] As Hamiltonians H_B, H_C are Hermitian operators, e^{-iH_B} and e^{-iH_C} are unitary operators and can be converted into circuits.

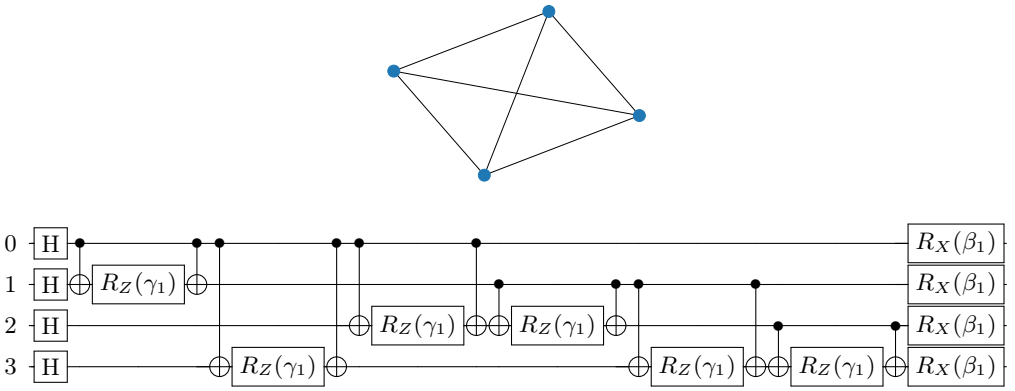


Figure 2.3: An example of QAOA circuit for MaxCut on a graph with 4 nodes. Here $p = 1$, corresponding to 2 parameters γ_1, β_1. The register of qubits is initialized as the $|0\rangle^{\otimes N}$ state. The first operation, the Hadamard transform, will prepare the state $|+\rangle^{\otimes N}$. Then the unitary $e^{-i\gamma_1 H_C}$ is decomposed into CNOT and R_Z gates. All possible edges of the graph are represented in the circuit with the presence of two CNOT gates and one RZ gates. Finally, the mixer $e^{-i\beta_1 H_B}$ is translated as a layer of R_X rotations.

tion from the measurement outcome. It is known that, in the limit of infinite depth, for optimal angles, the distribution of bitstrings will converge to the optimum [57].

While QAOA is an interesting candidate for tackling CO problems, several issues remain and we address a few in this thesis. Firstly, QAOA will have to compete with classical counterparts and we will face an algorithm selection issue, which we address in Chapter 3. Secondly, the quality of its output depends on procedures to set the values of the *QAOA angles*. In Chapter 4, we design and study different unsupervised learning approaches for setting these parameters without optimization. Finally, QAOA faces many practical challenges considering (some of the) limitations of real devices, which severely limits the size of problems we can tackle with it. In Chapter 5, we present a combination of QAOA with a classical heuristic, namely tabu search, allowing us to tackle larger problems than the number of qubits would allow running only QAOA.

2.5 Machine Learning

In 1959, the American pioneer in the field of computer gaming and artificial intelligence Arthur Lee Samuel published a paper [165] popularizing the term *machine learning*

(ML), defined as the field of study that gives computers the capacity to solve problems without being explicitly programmed. The definition of *learning* can be taken from [132]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." Hence, a machine learning algorithm uses experiences and (raw) data to improve its performance. ML has been applied to many fields/industries in order to extract actionable information from raw data. Such is the case in the energy sector. At TotalEnergies, ML has been applied to solve partial differential equations more efficiently [2], detect oil slicks (a film or layer of oil floating on an expanse of water) from satellite images [3], and model the injection of CO_2 in porous media [54].

Using quantum computing techniques for ML (or conversely ML for quantum) is a simple description of the field of quantum machine learning (QML) [21, 173]. In QML, PQCs can be used as ML models [18, 172, 45, 1, 141], allowing to tackle ML applications on near-term quantum computers. From Section 2.2, one understands that the VQA workflow with a PQC as a ML model resembles training a neural network (the optimizer can be a gradient descent method, commonly used in ML). Hence, they have also been named *quantum neural networks* (QNNs). The applications concerned mainly *classical data* but also *quantum data*. For instance, quantum experiments can produce interesting sets of quantum states to be used as datasets for machine learning tasks [90, 91]. QNNs have been used to tackle regression [130], classification [83], generative adversarial learning [208], and reinforcement learning tasks [94, 180]. Quantum models can exhibit clear potential in special datasets where we have theoretically provable separations with classical models [94, 117, 163, 188]. In this thesis, two chapters will treat different benchmarking studies, one with classical data and another with datasets of quantum states.

Let us explain more about the inner workings of QNNs. In the case of classical data, an input data instance $x \in \mathbb{R}^d$ is passed into a part of the PQC: the *data encoding part* (e.g. as the angles of a layer of rotations). We also have a parameterized part specified by a trainable vector $\boldsymbol{\theta}$, following the encoding part. Many circuit architectures exist but hardware-efficient PQCs [97] with an alternating-layered architecture are often considered in QML when no information on the structure of the data is provided [157].

This architecture is depicted in an example presented in Fig. 2.4 and essentially consists of an alternation of encoding unitaries U_{enc} and variational unitaries U_{var}. In the example, U_{enc} is composed of single-qubit rotations R_X, and U_{var} of single-qubit rotations R_z, R_y and entangling Ctrl-Z gates, represented as in Fig. 2.4, forming the

entangling part of the circuit denoted U_{ent}. The same circuit architecture can be used for quantum data coming as quantum states.

Finally, the user can define the observable(s) and the post-processing method to convert the circuit outputs into predictions (e.g. a probability of belonging to a certain class or a real value for a quantity of interest related to the ML task). Commonly, observables based on the single-qubit Z operator are used. When applied on $m \leq n$ qubits, the observable is represented by a $2^m - 1$ square diagonal matrix with $\{-1, 1\}$ values and is denoted $\mathcal{O} = Z \otimes Z \otimes \cdots \otimes Z$. The predictions will be used to evaluate a cost or loss function to train the model via an optimization algorithm.

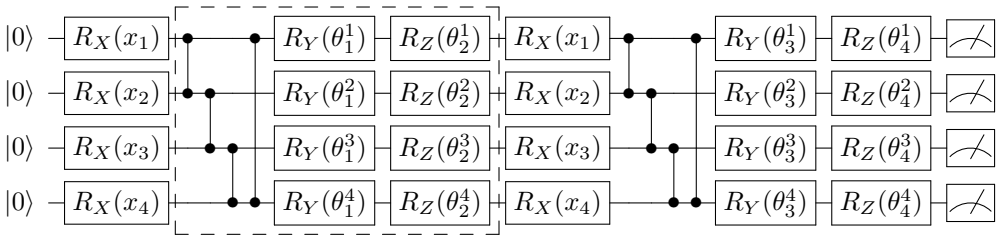


Figure 2.4: Parameterized quantum circuit architecture example with 4 qubits and ring connectivity (qubit 1 is connected to 2, 2 to 3, 3 to 4, and 4 to 1 makes a ring). The first layer of R_X is the encoding layer U_{enc}, taking a data instance $x \in \mathbb{R}^4$ as input. The entangling part with Ctrl-Z gates follows it. Finally, a variational layer U_{var} is applied. Eventually, we perform measurements. The dashed part can be repeated many times to increase the expressive power of the model. The encoding layer U_{enc} can also be repeated, a technique known as data reuploading [151]. Here we show an example of both repeated twice.

However, QNNs face various sorts of trainability issues such as exponentially vanishing gradients, known as barren plateaus [125, 42, 87, 88, 176, 122, 190, 9, 153], as well as the prevalence of local minima [23, 7], which can impact the complexity of the training process. Quantum hardware noise also impacts trainability [197, 184]. Finally, applications can require a large measurement-count overhead due to the large datasets involved. Hence, QNNs require tailored optimization procedures to address such trainability issues. Chapter 8 of this thesis is dedicated to the design of a resource frugal optimizer for QML applications. We benchmark it against state-of-art optimizers and apply it on a few datasets of quantum states.

Another costly procedure is hyperparameter optimization, given that many ways exist to design quantum circuits for ML tasks. A user can tweak many hyperparame-

ters of a QNN, affecting its performance. For instance, the parameterized layer can be repeated multiple times, which increases its *expressive power* (the ability to approximate a wider range of functions) [179]. The data encoding strategy (such as reusing the encoding layer multiple times in the circuit - a strategy called *data reuploading*) also influences their expressive power [151, 174]. However, there is currently limited intuition as to which hyperparameters are important to optimize and which are not for their *predictive performance*. Such insights can lead to much more efficient hyperparameter optimization [31, 60, 133]. This research question is addressed in Chapter 7, where we apply QNNs on a few open-source classical datasets.

Chapter 3

Towards algorithm selection in NISQ era

The Quantum Approximate Optimization Algorithm (QAOA) constitutes one of the often-mentioned candidates expected to yield a quantum boost in the era of near-term quantum computing. In practice, quantum optimization will have to compete with classical heuristic methods, which have the advantage of decades of empirical domain-specific enhancements. Consequently, we will face the issue of algorithm selection. In this chapter[1], we introduce this problem to the quantum optimization domain.

Specifically, we study the problem of detecting those problem instances where QAOA is most likely to yield an advantage over a conventional algorithm. As our case study, we compare QAOA against the well-understood approximation algorithm of Goemans and Williamson (GW) on the Max-Cut problem. As exactly predicting the performance of algorithms can be intractable, we utilize machine learning to identify when to resort to the quantum algorithm. We achieve cross-validated accuracy well over 96%, which would yield a substantial practical advantage.

In the process, we highlight a number of features of instances rendering them better suited for QAOA. While we work with simulated idealized algorithms, the flexibility of the ML methods we employed provides confidence that our methods will be equally applicable to broader classes of classical heuristics, and to QAOA running on real-world noisy devices.

[1] Contents of this chapter are based on [135];Charles Moussa, Henri Calandra, and Vedran Dunjko. To quantum or not to quantum: towards algorithm selection in near-term quantum optimization. *Quantum Science and Technology*, 5(4):044009, 2020.

3.1 Introduction

The QAOA algorithm consists of a quantum circuit of a user-specified depth p, inducing $2p$ *QAOA parameters* which are tweaked to solve a combinatorial optimization problem. To make the chapter more self-contained, we refer to Chapter 2.4 for more background on QAOA. QAOA is interesting from the perspectives of approximation, and heuristic optimization.

Approximation with QAOA - Initially, the QAOA algorithm was designed as an *approximation algorithm*, i.e., an algorithm intended to provide an efficient, but approximate solution to an otherwise intractable problem. Approximation algorithms differ from *heuristic* methods in that they provide mathematically provable a-priori guarantees on the quality of the solution, whereas heuristics utilize expert knowledge, and domain-specific operators to achieve good performance in practice, with less concern regarding theoretical worst-case bounds. For instance, one of the most natural applications of the QAOA algorithm is to solve so-called MaxCut problems, where the task is to identify a bipartition of a given graph, which maximizes the number of edges (the *cut*) crossing the two sets. Finding the actual maximum of the cut C_{max} is NP-hard [2], but the problem allows non-trivial polynomial-time approximation. The best known classical approximation algorithm, of Goemans and Williamson (GW) [69], guarantees an *approximation ratio* of $\alpha \approx 0.878$ (referred to as the GW bound). In the case of randomized algorithms like GW, this means that the value of the output cut is at least αC_{max} with high probability. Under which conditions better efficient approximations can be achieved is a hard open question, and depends on details. For instance, it is known that achieving an approximation ratio above $16/17 \approx 0.94$ for general graphs is NP-hard [80] (so as hard as finding the exact solution), and it is possible that the actual NP-hard bound may be α. For more special graphs the GW can do better, e.g, for 3-regular graphs, the ratio of 0.932 is achieved. To use QAOA for optimization, a critical step is finding the optimal classical circuit parameters for a given depth. For a constant depth, this can be done in polynomial time, yielding a poly-time approximation algorithm [3]. The question of whether QAOA can improve over GW for any constant depth was open for a number of years and has been resolved very recently, in the negative [28]. However, already at depth $p = 1$, it achieves the ratio of 0.6924. While falling short of the GW bound, this is a highly non-trivial result,

[2] Technically, the decision variant of the problem, deciding whether the cut is larger than some integer k, is NP-complete.

[3] Without constant depth limits, it is known that QAOA can achieve exact solutions, but the classical parameter optimization then can take exponential time.

as prior to GW, the best algorithms achieved only $1/2 + o(1)$[4], even after decades of efforts. This makes QAOA an interesting algorithm and allows the possibility that QAOA performs better than GW (or some other classical algorithm) on some classes of instances. Setting the theoretical approximation aspects aside, this suggests the substantial practical potential of QAOA as a *heuristic optimizer*.

Heuristic optimization with QAOA - The above-mentioned theoretical results motivate studying how to best use QAOA for practical optimization problems via numerical (and real-device) experiments. As mentioned previously, to actually use QAOA one needs to find the optimal classical parameters. Although finding exact optima for any constant depth can be done in polynomial time [57], the actual overheads are prohibitive, requiring billions of evaluations already for small graphs and depth in low single digits. Almost as much research has been dedicated to developing heuristics and analytical techniques to find good circuit parameter values for QAOA [206, 26, 177, 193], as has been to the actual evaluation of the performance of QAOA [46]. This is justified as any evaluation of QAOA will be in part limited by the limitations of the classical optimizer. The early studies are promising, showing, e.g., that QAOA with depth $p = 8$ significantly outperforms GW on a set of Erdős-Rényi random graphs (with edge probability 0.5) [46]. A somewhat related theoretical study identifies that there exist instances where QAOA will outperform quantum and simulated annealing methods [186].

Perhaps the first main issue with QAOA is that many open questions still remain, e.g., regarding the comparison of QAOA with other heuristic optimization methods, on other graphs, with different optimizers, and with varying levels of experimental (or simulated) noise – in part as simulating, or actually running it on a real device QAOA is computationally costly, for larger graphs, preventing large-scale analyses. We note that the issue of noise was addressed e.g. in [73, 5, 203].

The second issue is more subtle and pertains to the importance of predicting when to use QAOA, on an instance-to-instance basis. This capacity would allow us to build better hybrid solvers, with possible implications in combined approximation algorithms (see e.g. [68]) or in the heuristic setting (see e.g. [158]). Note that despite these early successes of QAOA on, realistically speaking, very small instances, QAOA cannot beat even the bare GW algorithm on all instances at any constant depth [81, 28]. This holds even when assuming ideal parameters and zero noise. We note that this result was not unexpected, as previously known results imply that, under the so-called unique games conjecture (UGC [101, 102]), QAOA could not beat GW unless BQP contains NP.

[4] In other words, the achieved approximation ratio is below $1/2 + \epsilon$ for any constant $\epsilon > 0$.

QAOA should thus not be expected to universally win even against GW, let alone all standard heuristic methods, optimized for certain classes of problems. Further, given the expected substantially higher expenses of running actual quantum hardware, in practice, we will need to leverage this additional cost against the actual improvement (potentially) offered by QAOA over less costly classical algorithms. Thus one should tackle the challenge of *algorithm selection* in quantum optimization, which in the classical realm is mostly concerned with using the fastest algorithm for a given instance [108]. As we clarify presently, detecting which instances are "quantum-suitable" is likely itself a hard problem, requiring a heuristic treatment. Investigating whether such detection is possible using machine learning techniques is the key theme of this work.

Here, we investigate such a detection method for the case of the QAOA versus the GW algorithm. A thoughtful critic may point out that a) since the overall approach is heuristic, and b) since QAOA is most likely going to impact the heuristic domain, detection-of-advantage strategies against classical heuristic algorithms would be more relevant. While we absolutely agree with this sentiment, there are important reasons we opted nonetheless to focus this first investigation of its type on the QAOA/GW setting. The main reason is pragmatic: at present QAOA can only be run on small instances – as defined in classical literature, those with fewer than 100 nodes in [51]. In our case, we are indeed limited to 24 nodes. Mainstream classical heuristics can quite easily attain optima in these cases, which prevents any reasonable analysis. Comparing against purposefully bad algorithmic choices would have been equally uninformative. In contrast, the GW algorithm provides the right combination of solid, yet imperfect performance on reachable instance sizes for our purposes (despite the fact that it has strong provable worst-case performance bounds rendering it, technically, an approximation algorithm). As a potential secondary advantage, the ML techniques employed may provide hints for further theoretical insights into the performances of these two well-studied algorithms.

We highlight that since our actual object of study is the capacity of ML methods for algorithm selection (where one algorithm is QAOA), the choice of the second algorithm is not central. Further, thanks to the nature of ML approaches, our technique is likely flexible enough to be extended to settings with genuine heuristic algorithms, as soon as instance sizes allow for meaningful training sets.

Contributions - In this work, we contributed in these directions, as follows:

- we compare the performance of QAOA with GW on a different family of graphs, namely 4-regular graphs;

- we specify two pragmatically motivated criteria specifying when QAOA should be used over GW;
- we design a machine learning model which achieves over 96% and over 82% balanced accuracy in predicting which MaxCut instances satisfy the above criteria, respectively, which can be used for algorithm selection;
- we highlight a number of graph properties strongly correlated with a QAOA advantage, which may help guide further theoretical analyses.

The structure of the chapter is as follows. Section 3.2 provides the necessary background on GW and QAOA. In section 3.3, we detail the simulation performed, comparing QAOA and GW, and which we used to fix the criteria when one algorithm should be chosen over another, and to generate the datasets needed for machine-learning-based algorithm selection. In section 3.4, we describe the algorithm selection model and discuss the critical features (characteristics) of MaxCut instances that influence which algorithm does better. We conclude this chapter with a discussion in section 3.5.

3.2 Background

The MaxCut problem is one of the famous 21 NP-complete problems identified by Karp [98], and it naturally occurs in computer science and physics. Consider a graph G over a vertex set V, and edge set E, with N vertices. The problem is to find a subset $S \subset V$ such that the number of edges between S and $V \setminus S$ is maximized. The set of edges between S and $V \setminus S$ is called a *cut*, and we will denote the size of the maximal cut with C_{max}. The problem naturally extends to weighted graphs, where the objective is to maximize the total weight of the edges of the cut. A common phrasing of the MaxCut problem is as follows. Let w_{ij} be the weight associated to the edge $(i,j) \in E$ (1 in the case of an unweighted graph), then the MaxCut problem is to identify the bitstring $\mathbf{z} = (z_1, \ldots, z_N) \in \{-1, 1\}^N$ where $z_i = 1 \Leftrightarrow i \in S$ maximizing

$$C(\mathbf{z}) = \sum_{(i,j)\in E} w_{ij}(1 - z_i z_j)/2. \tag{3.1}$$

This quadratic binary optimization (QUBO) formulation connects the MaxCut problem with the task of finding ground states of classical spin Hamiltonians. As phrased, the MaxCut problem is to identify the cut set ($S(\mathbf{z}) = \{i | z_i = 1\}$ s.t. $C(\mathbf{z}) = C_{max}$),

but, even the problem of identifying the cut size (distinguishing the *max* versus the *argmax* problem for $C(\mathbf{z})$) is computationally difficult. The MaxCut problem is in the APX class [149], the set of NP optimization problems that allow polynomial-time approximation algorithms with approximation ratio bounded by a constant).

3.2.1 Goemans-Williamson Algorithm for MaxCut

The best classical approximation algorithm for MaxCut is that of Goemans and Williamson [69] (GW). It is based on solving a related relaxation of the problem – intuitively, we solve an optimization problem in the continuous domain $[-1, 1]^N$ as a semi-definite program, giving a real-valued solution with the associated cost C_{rlx}. Then a sampling process that depends on the probabilities specified by the entries in the real-valued solution (a *random projection routine*) generates a feasible bitstring solution. Note that the cost of the found bitstring is upper bounded by C_{rlx}, and in fact, it can happen that $C_{max} < C_{rlx}$.

Nonetheless, the expected value attained by the cost function using this procedure is provably lower bounded by αC_{max} with $\alpha \approx 0.878$, for all input graphs - in other words, it is an $\alpha-$approximation algorithm.

While better approximation bounds are possible for special graphs (e.g. for 3-regular graphs [80], or for graphs with large cuts [69]), assuming the UGC [101], GW provides the best possible polynomial-time approximation, unless P=NP.

3.2.2 Quantum Approximate Optimization Algorithm

The QAOA approach, presented previously in chapter 2.4, yields a quantum state defined by $2p$ parameters or QAOA angles $\gamma_i, \beta_i, i = 1...p$. Such a quantum state when measured yields a probability distribution over all possible bitstrings. The classical optimization challenge of QAOA is to identify the sequence of parameters γ and β so as to maximize the expected value of the cost function from the measurement outcome, i.e. the quantity $r = F_p(\gamma, \beta) = w - \langle \gamma, \beta | H_C | \gamma, \beta \rangle$.

If we denote the optimal parameters maximizing F_p with γ^*, β^*, the approximation ratio of QAOA *achieved on the given instance* is given by:

$$r^* = \frac{F_p(\gamma^*, \beta^*)}{C_{max}}. \tag{3.2}$$

For this value to be computable, C_{max} for the given graph must be known. Note that it is known that if $p \to \infty$, then we have $r^* \to 1$ [57]. This quantity – the expected

achieved approximation ratio – is equally defined for the GW and for the QAOA algorithm, and we use it as our central figure of merit regarding the performance of the two algorithms on MaxCut instances. In the next section, we analyze the performance of both algorithms on a new domain – that of 4-regular graphs, complementing previous analyses done on 3-regular graphs and Erdős-Rényi random graphs. The obtained dataset will be used to define reasonable criteria when considering QAOA to have done "significantly better", and also to train a machine learning classifier to predict this criterion.

3.3 Performances on 4-regular graphs

In order to systematically compare QAOA against other algorithms, certain choices regarding the QAOA set-up (hyperparameters), the problem test set, and regarding the exact notion of "better performance" need to be made.

Regarding the problem test set, as mentioned earlier, we chose to focus on 4-regular graphs, in part because they are relatively easy to generate, yet have not been considered in the literature in the context of comparison to GW. Although in practice both GW and QAOA would be used by selecting the best performance out of a number of runs, in this work we opted to compare averaged performances as they come with a number of practical and theoretical advantages. From the theory perspective, both QAOA and GW have been studied theoretically with respect to average performance and this theoretical insight allows us to have a better grasp on our empirical findings. From a pragmatic perspective, due to the fact we can only investigate comparatively small instances, the best cut value over a reasonable number of M (user-defined) samples would end up in ratios very often equal to optimal performance. Considering average performances allow us to have a more robust comparison (combined with best performances if relevant). Note also that in the event that we could explore significantly larger instances, this extreme behavior of seeing mostly optimal results would cease, and observing averages would be more representative.

Along the same lines, we note that measuring averages, or best-out-of-M-shots actually yields, technically speaking, distinct algorithms, with different run-times (due to M). Which algorithm is more relevant depends on the point of view of a practitioner (time to run on real hardware, number of function evaluations allowed during optimization...). If the performance measure is best-out-of-M-shots, this adds another hyperparameter – a degree of freedom – in the comparison. Technically, our setting is the averaged $M = 1$ setting. Using higher M values would involve other aspects

of the output distribution (which can also be subject to optimization), yielding a significantly more complicated process. Finally, we highlight that the main objective of this work is to investigate the capacity of ML methods to detect when QAOA could yield better performances compared to a classical algorithm. The actual specification of the algorithm is of course important, but still secondary for our study; we note that, from the onset, we had to make a number of modeling selections, including choosing the optimization algorithm, fixing the depth, and in the same vein, choosing to study $M = 1$.

Regarding these other important choices in the QAOA algorithm details/hyperparameters, the depth of the circuit we will allow, and the optimizer used is obviously very relevant. The basis of our optimization is the Nelder-Mead (NM) algorithm, with some QAOA-specific tweaks, detailed shortly. NM is a common choice due to the relatively small dimensionality of the parameter space. For instance, NM was used in [73] to give a computational time cost for running QAOA against a classical heuristic. Perhaps the most important choice, the allowed circuit depth, is non-trivial. Note that it is known that if the circuit is deep enough, relative to the instance size, we will find true optima (given that the QAOA circuit parameters themselves are chosen optimally). The exact functional scaling of the depth required for this is not known, but as clarified, having the depth depending on the instance size would lead to intractable optimization demands regarding the QAOA parameters – and defeat the idea of NISQ-friendly constant-depth constructions. However, since we are bounded to relatively small instances due to the cost of quantum simulation, already depths 7-9 may allow us to achieve better performance than what is asymptotically possible for any constant depth[5]. On the other hand, keeping the depth too low may prevent any advantage in the upper range of the graph sizes and test-set sizes that we could tackle. To help us with the choice here, we analyzed the performance of QAOA on a range of depths, as described next.

Finally, this brings us to the choice of the appropriate definition of "better performance". Relating to previous works, in [46] (although the objective was not to formally define such a criterion), an average improvement of the approximation ratio of 2% was

[5]More precisely since QAOA finds the exact optimum in the limit of infinite depth, there exists some function $d(n)$, for which QAOA of depth $d(n)$ can achieve performance approximating the optimum arbitrarily well, e.g. better than the GW bound. However, this does not yield an efficient algorithm, even if $d(n)$ grows very slowly, as the finding of the ideal parameters is in principle exponentially expensive in $d(n)$. Nonetheless, for the values $n \leq 22$ it may be the case that the depth of 9 which we study is larger than $d(22)$, allowing us to nearly always, or always beat GW in our experiments. Thus this is a finite-size effect, but of the kind, we may care about in practical computing.

considered a significant improvement. Our choice specified in the next section is also guided by the data acquisition stage, i.e., the performance analysis on our graph test sets.

3.3.1 Simulation methods

We built a set of randomly generated 4-regular graphs from where we varied the number of nodes $N = 11, \ldots 24$, with 20 instances generated per size, i.e., in total 280 instances. For each graph, the C_{max} value was found by brute-force so that approximation ratios can be computed. We evaluated both the GW and QAOA algorithms in each instance. Regarding GW, we ran the standard algorithm and computed the effective performance by dividing the expected cost (estimated using 1000 projections) by the actual C_{max}.

Regarding our evaluation of QAOA, more details are warranted. We run QAOA on a quantum simulator, starting from $p = 1$ and increasing the depth up to $p = 10$. The algorithm we use for the QAOA parameter optimization is based on the Nelder-Mead (NM) method, whose performance dramatically depends on the choice of initial values/angles. We embed the NM algorithm in an optimization procedure outlined in [26] for finding good near-optimal angles for maximizing $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$. The procedure is based on the observation that the evaluation of the QAOA objective function for fixed parameters on graphs generated from a "reasonable" distribution concentrates with little fluctuation. This suggests that optimal angles for QAOA from one instance can be used as a starting point for other instances, which is then improved via local search. Specifically, for a given depth, we find candidates for optimal angles starting from random "seed" positions. The newly found angles are then used as starting points for NM in the next instance. After passing through all graphs in the set, we attempt to improve the best-achieved ratios for a given p. This is done by trying out all angles found for each graph as seeds for further optimization with NM, but only in the case that the achieved performance was not already better than what GW achieves. This is motivated by the fact that we are predominantly interested in the performance of QAOA assuming the capacity to achieve optimal parameters/angles, and identifying when that performance surpasses GW [6].

[6] As we explain shortly, we will investigate another more stringent criterion which requires an objectively higher performance quality and a larger gap over GW. In principle, we could have forced further optimizations on all instances that failed that criterion (but were already better than GW). We opted not to do so, because such a process would not be feasible in practice. Note that we can guide our optimizations of QAOA relative to the performance of GW (as we can run GW in polynomial time), but we cannot do the same in practice for any criteria which evaluate the actually

In those cases where QAOA still underperformed relative to GW, we also run also QAOA with depth 11 and 12, but ultimately, even this did not improve the performance. In the following subsection, we present the results obtained using the method above.

3.3.2 Results

We used simulations to first identify the depth of QAOA we wish to focus on. Intuitively we looked for a depth when using QAOA is obviously interesting relative to the performance of GW. Such a decision will also depend on the choice of the criterion of what makes one algorithm heuristically better than another. In our case, we looked at three figures of merit: the minimum and median of the performance ratio, and the overall percentage of instances where QAOA outperformed GW. Additionally, we address the more practically motivated criteria where we aim to identify settings where QAOA does not just outperform GW but does so with a significant margin, and in the regime where the performance is already much better than the GW lower bounds. Here we selected the threshold of 98% for practical and pragmatic reasons: it may be high enough that no poly-time approximation algorithm can do better even on our restricted family of graphs (regardless of the UGC conjecture). This would make it a "genuinely heuristic" feature. Yet, it is low enough that we have sufficiently many YES and NO instances for a meaningful analysis.

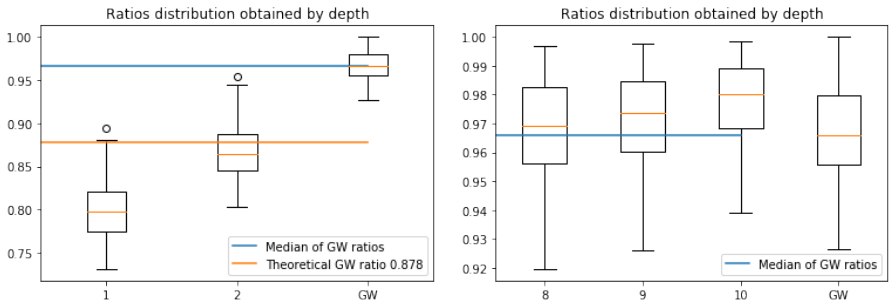


Figure 3.1: Boxplot of ratios for $p = 1, 2$ and $8, 9, 10$ as well as the ratio of GW.

Fig.3.1 shows boxplots, representing the quartiles of achieved ratios, presented according to depths from 1 to 10. Interesting depths for QAOA start at $p = 8$. Indeed, for $p = 8, 9, 10$, QAOA yields a better-expected cost than GW on respectively

achieved ratio of QAOA, as this would require computing C_{max}.

Table 3.1: Median and minimal ratios achieved by the QAOA algorithm for a given depth. For comparison, we also provide the values achieved by the GW algorithm. We underline the first value where QAOA surpasses GW.

Depth	Minimal ratio	Median ratio
1	0.7312	0.7979
2	0.8026	0.8641
3	0.8488	0.9050
4	0.8777	0.9302
5	0.8993	0.9446
6	0.9031	0.9523
7	0.9209	0.9607
8	0.9195	<u>0.9692</u>
9	0.9259	0.9737
10	<u>0.9390</u>	0.9800
GW	0.9265	0.9658

57.5%, 67.8%, and 92.14% of the instances. Table 3.1 displays the obtained minimum and median ratios (averages yield similar conclusions but are generally more sensitive to outliers) on the generated set. At depth 10, we see a clear advantage of QAOA with respect to the minimal ratio criterion.

In addition to the expected cost, we also investigated the standard deviation of the output of the algorithms. This quantity is particularly relevant when randomized algorithms are used as subroutines in other algorithms (e.g. as steps in local search), in which cases more stable performance (with a worse mean) may be preferred over on-average-better, but much less reliable performance. We took the optimal angles obtained at depths 9 and 10, from which we sample the circuit 1000 times and compute the MaxCut cost. For GW, we use the 1000 random projections obtained for computing the expected ratios. From Fig.3.2, we observe that the QAOA output is more spread out around the average than GW, and would require higher circuit depth to decrease it. In [57] it was shown that the upper bound on the spread (specifically, the variance) of the output distribution is dictated by a term of the form $(v-1)^{2p+2}$, where v is the constant degree of the graph (and p is the depth). Due to this form, we applied a logarithmic fitting to the found variances, to estimate at which depths QAOA spread is guaranteed to concentrate more than GW. Numerically, we estimate this to occur after depth 20. However going to these higher depths did not yield better-expected ratios (and did make optimization significantly more costly and unstable), so we restricted the depths to $p \leqslant 10$ in the subsequent steps of our study.

Last but not least, we are interested in defining when QAOA is a good heuristic. In [46], the average approximation ratio (given on graph instances of similar size) at depth 8 was between 0.98 and 0.96, with an improvement close to 2% against GW. We then choose arbitrarily to define QAOA as a significantly good heuristic when QAOA yields a ratio higher than 0.98 and an improvement against GW by at least 2%. This occurred on 24 instances (8.5%) and gives rise to potentially good situations when QAOA is more suited to be advantageous. Indeed, those will be the cases where QAOA is nearing perfect performance. At most the returned cut values will differ by 1 from C_{max} (on small instances) on average. Although this criterion may be pretty strict, it is interesting for algorithm selection. Having defined labels for our generated set of instances, we inferred when QAOA is more suited for applications.

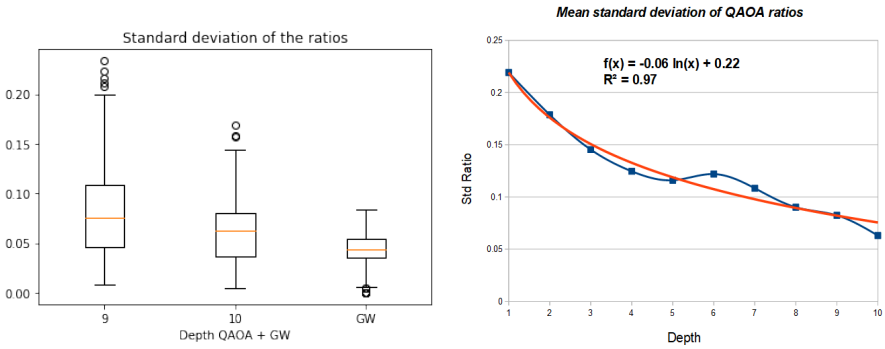


Figure 3.2: Standard deviations of the ratios over the generated graphs obtained by sampling 1000 times for $p = 9, 10$. The mean standard deviations are respectively: $0.0824, 0.0631$, and 0.0432. The second plot is a logarithmic fitting of the mean deviation by depth.

3.4 Characterizing QAOA advantage

As indicated in the previous section, we identify two basic regimes of interest. The first criterion (1) – the *GW vs. QAOA* criterion, is the basic question: can we identify those instances where the ratio achieved by GW is surpassed by that of QAOA at all. The second is a question of whether QAOA performs remarkably well in a more general sense – whether it is a "good heuristic", so to say. Note that what values an algorithm has to attain to be considered a "good heuristic" is of course arbitrary, and our decision of setting this threshold at 0.98 is motivated by three main factors:

this is the highest value where we obtain non-trivial sample sizes where QAOA also outperforms GW by a substantial margin (2%); this is also likely in the NP-hard regime (if the performance could be attained for *all* graphs), even when restricted to 4-regular graphs. Finally, achieving objectively high-performance values intuitively reduces the chances that other heuristic algorithms will generically be better at those instances.

Thus our second criterion (2) – *the good heuristic criterion* is: does QAOA attain performance above 0.98 with an advantage margin of at least 0.02 over GW?

Note that deciding the first criterion is trivial if we are allowed to actually run QAOA. The second criterion may be more problematic since it implies we can guess whether QAOA will do better than 0.98.

Since we are interested in developing a methodology that helps us decide whether to run QAOA at all, we are only interested in detection methods that can be run efficiently on a classical device. So one may wonder whether characterizing instances with respect to the two criteria can be done algorithmically in polynomial time on a classical computer.

Regarding the two criteria, although we do not make any hard claims specifically for our choices, we point out that it is not difficult to see that finding exact predictors of performance is in general exceptionally difficult. For instance, deciding if GW does better than some threshold $\alpha < \beta$ (where α is the GW bound) is already NP-hard under the UGC[7]. Given access to an algorithm deciding this, and to the cut value r returned by the GW algorithm, in the case the output is YES, we have a cut value above βC_{max}, where C_{max} is the true optimum. In the NO case, we can conclude that there exists a cut with value $(r/\beta) \geqslant (\alpha/\beta)C_{max} > \alpha C_{max}$ (since $\alpha C_{max} < r < \beta C_{max}$). This results in an overall $\min(\beta, \alpha/\beta)$-approximation algorithm, beating GW[8]. Similar arguments can be applied to other algorithms and criteria as well. For instance, assuming QAOA is an α'-approximation algorithm (we know $\alpha' \leqslant 5/6 + \delta$ for constant p and any $\delta > 0$) deciding if QAOA does better than the GW bound $\alpha + \epsilon$ by any margin ϵ is likely hard. YES results give an $\alpha + \epsilon$ approximation, whereas NO yields an α'/α approximation, which beats GW if $\alpha' > \alpha^2 \approx 0.77$. This is significantly lower than the plausible bound of 5/6 [28]. Thus, unless QAOA is weaker than a $0.77-$approximation algorithm for all constant depths, deciding whether it beats the GW bound on a given instance would imply $BQP \subseteq NP$, under the UGC.

[7] Under the unique games conjecture (UGC), the NP-hard bound for approximation coincides with α achieved by the GW algorithm [101, 102].

[8] Note, already estimating the cut size (rather than outputting the cut itself) better than the GW bound is NP-hard, under the UGC [101, 102].

In the more general case when we compare QAOA against more complicated heuristics, for which bounds may be unknown, proving formal claims may be even more difficult, and as we explain shortly, arguably less useful.

3.4.1 Machine learning for performance prediction

The reasons above are important factors why we resort to a machine learning (ML) approach, specifically supervised learning, to decide our criteria (1) and (2). Another one is that machine learning methods are also more robust, and flexible. We note that exactly the same method we employ here for characterizing the performance of the idealized QAOA relative to GW can be used with any other classical heuristic, and with real-world, noisy QAOA implementations.

Our method is inspired by the approach in [51] where the authors inferred a ranking between different classical MaxCut heuristics, gathering instances from many MaxCut and QUBO libraries. However, due to the need for quantum simulation, we could only provide a more modest sample set and instance sizes.

In essence, we prepare a dataset of instances for which we compute the criterion value (NO/YES, or 0/1 for each criterion) – in supervised learning, this is called a *label*. We then train a ML model to fit this value, using a subset of data – the training set; the performance reported is obtained by applying the model to the testing set. Using 4-fold cross-validation in our case gives an idea of the generalization and robustness of the model.

When relying on machine learning, a key step is the identification of the *features*, that is, the pieces of information (i.e. properties of the instance) based on which we will be training the model and making a decision. While the graph itself is a feature implicitly containing all the information we may care about, it is almost always useful to pre-compute many other derived properties (e.g. the graph density, or some property of the spectrum of the graph adjacency matrix) and uses them directly in training the ML classifier. Note that in principle the performance itself could be a feature, however, we limit ourselves only to use features that can be computed classically, and efficiently relative to the running of the QAOA algorithm – the entire idea is to decide *whether* to employ the quantum device at all.

In general, our approach incorporates the 3 standard phases when employing ML: preparing the dataset (incl. pre-computation of the features, the running of the QAOA to compute the true criterion value, etc.), training the ML model, and evaluation. This process was iterated a number of times before the most informative features were

identified. In our analysis we also discuss which features contributed the most to prediction accuracy; this may also be informative for theoretical analyses of QAOA performance.

The learning model - In our study we have initially considered a number of models to perform prediction. We have experimented with traditional ML models such as tree-based gradient boosting, random forests, LightGBM and XGBoost [32, 43, 99] – which are simple and would have high interpretability (meaning we can infer something about why the model makes certain decisions). However, accuracies were very low (less than 63%). This presumes that more complicated ML machinery is needed, especially when dealing with small datasets.

As we are predominantly interested in high accuracy for these computationally hard predicates, vital for a practical advantage, we opted to deal with relatively complex models, the hyper-parameters of which were optimized using automated (auto-ML) techniques. Specifically, we built our models using the TPOT library [146], which builds the models using an evolutionary algorithm. A clear advantage of such auto-ML methods is that they offer a significantly enhanced level of flexibility for the user, as much of the vital hyper-parameter optimization is taken care of automatically. This suggests that, with very few interventions from the user, the same techniques we provide could be used for noisy QAOA or comparisons against very different types of heuristics. Such flexibility is a key desired feature of automated algorithm selection.

Features utilized for prediction - In our analysis we do not employ the raw graph description as a feature as it carries a lot of irrelevant information (e.g. both algorithms are invariant under graph isomorphisms). We used some of the features proposed in [51] suited for unweighted graphs while generating others for our results via a trial and error process. In the final analysis, we investigated the potential 20 features, listed in appendix 3.5. These are grouped into three classes: (i) graph spectral properties, (ii) subgraph characteristics (e.g., maximum independent set size), and (iii) certain GW performance features (e.g., the normalized value of the relaxed problem). This third class of features is of course GW-specific, but it should not be surprising that the performance of GW itself carries a lot of information about whether QAOA will do better. We note that some of the features in group (ii) are actually NP-hard to compute, so we cannot expect their exact values to be used in practice to guide our decisions. While we nonetheless investigated their value as predictors (we show they are absolutely unnecessary), the best results we report only utilize efficiently computable features. Criterion (1) is tackled before (2).

3.4.2 Predicting Criterion (1): QAOA vs. GW

We computed the features for each of our 280 graph instances and set the label to 1 when GW outperforms QAOA (at depth 10) in terms of the approximation ratio, and 0 otherwise. Note that to compute this particular label, the computation of the actual optimum is not necessary, but the running of QAOA is.

We then built a classifier by 4-fold stratified cross-validation and optimized the average balanced accuracy over the validation sets; the data was split into 4 subsets, and the model is trained on 3 of them while being tested on a fourth. Stratified sampling is used in order to maintain the ratios of the labels in the sets. This procedure is performed 4 times and performances are averaged. k-fold cross-validation is a commonly used method to build ML models on small datasets while still giving a robust idea of performance and generalization features.

Since our data is highly biased (QAOA outperforms GW significantly, more than half of the time), the relevant measure of performance is the so-called balanced accuracy [33], which uniformly averages the performances for YES and NO instances.

In our analysis, we initially used all 20 features and then proceeded to prune out a smaller number with the highest impact on accuracy. In the end, we have identified two features that alone enable the same balanced accuracy, as all features combined, and these are:

- expected_costGW_over_sdp_cost: the expected cost over the 1000 random projections per instance divided by the cost of the relaxation C_{rlx},
- std_costGW_over_sdp_cost: the standard deviation of the cost approximated using 1000 random projections divided by C_{rlx}.

In other words, to achieve the +96% accuracy we report, we only need to use these two features.

A few comments are in order; first, note that both of these features exclusively characterize the output of GW – this is not too surprising as the criterion "better than GW" strongly depends on GW performance. For instance, the first of these two features is likely closely correlated to the actual performance of GW in terms of the approximation ratio (it is the exact value when $C_{rlx} = C_{max}$). However, we can also understand the performance analysis of GW as a feature extraction mechanism, which identifies the properties of graphs that also make them suitable for QAOA. This is indeed the case, as our classification algorithm correctly predicts QAOA advantage even in many instances where GW did exceptionally well, and also, correctly predicts

that QAOA underperformed also when GW did "badly" as well – in other words, we capture more than the quality of performance of GW. Also importantly, this shows that NP-hard features are really unnecessary to decide criterion (1).

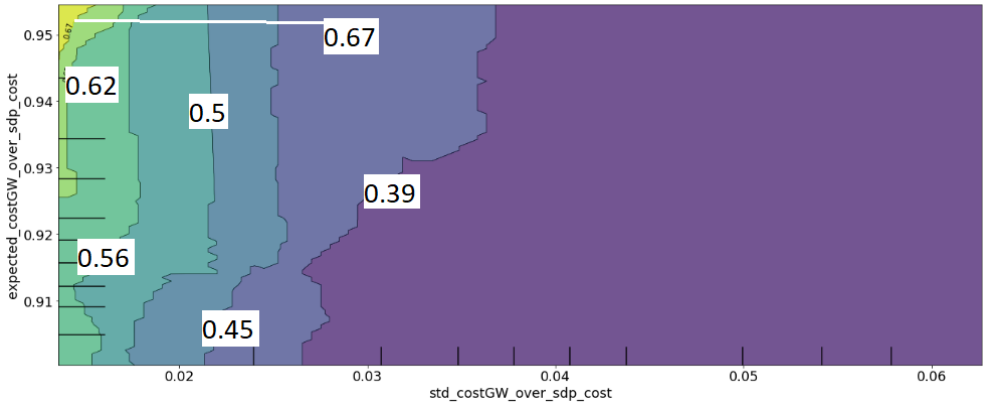


Figure 3.3: Partial dependence plot for the classifier built on two features. The contour lines from left to right represent probabilities thresholds: $0.67, 0.62, 0.56, 0.5, 0.45, 0.39$.

We analyzed these two features and their relationship in more detail. By looking at the partial dependence plot of the classifier in Fig.3.3, we quantified how the predicted probabilities evolve according to those two features. In more detail, our classification model outputs probabilities of assignments, and not the labels themselves. The closer the probability is to 1, the more likely the output label will be 1. From the plot, we observe that the smaller the standard deviation of the cost applied on the random projections, and the closer the expected cost to C_{rlx} is, the higher the probability that GW outperforms QAOA. But this also means that the GW performance is not far from an actual optimum since C_{rlx} upper bounds C_{max}. In order to understand the importance/impact of these features, we also performed a so-called permutation importance analysis, in essence, based on permuting the values of a column in the dataset corresponding to one feature and measuring the impact on the overall classifier performance. When this measure was applied to the two features above, we noted that performances decrease by 0.4283 ± 0.1843 for the standard deviation feature and 0.0533 ± 0.0743 for the expected cost, indicating the prior is more informative.

While showing that GW performance influences whether QAOA outperforms it is unsurprising, it does open the obvious question of whether it is possible to find other sufficiently informative features. We would especially be interested in finding features

to be used in place of the standard deviation feature, which requires costly random projection evaluations.

To this end, we have rerun the analysis from scratch, running through a new cycle of automated hyper-parameter optimization, trying to match the performance without using these two GW-specific features – including using the NP-hard features. However, we have failed to achieve performance close to when those features were available. To understand this further, we have applied a regression approach, trying to predict the values of the expensive feature std_costGW_over_sdp_cost from the other features. Note that if this were possible, then the other features indeed did contain sufficient information, even without these expensive features being explicitly present. We analyzed the explained variance of the constructed regressor and obtained a low value of 0.7693. With this result, it became apparent that we should not expect to be able to recover the same levels of accuracy without these costly (but still efficiently computable) features, using our machine learning model. In principle, of course, all features should be distillable from raw graph data, and in future work, it will be valuable to achieve as good or better performance relying on better-tuned models, and cheaper features.

To summarize the results of this section, we can report that the prediction performance for criterion (1), which can be achieved when many NP-hard graph-theoretical features are available, along with the less costly features, can easily be matched using just properties derived from the tractable GW performance. It is worth noting that the generation of the datasets for this prediction is also tractable (albeit, expensive), as it requires only the running of QAOA and of GW.

3.4.3 Predicting Criterion (2): QAOA as a High-Performing Heuristic

For this analysis, we considered the same dataset but the label was changed. We labeled with 1 the instances where the QAOA ratio exceeded 0.98 and where the gain was at least 2% over the performance of GW. Note that to generate this label, we did need the true C_{max} values which we computed by brute-force. We will discuss this issue presently. We built a classifier that utilized all the features we discussed previously, including the features relying on the random projections and the NP-hard graph-theoretical features, which lead to 0.8255 in balanced accuracy. The analysis was done again using 4-fold cross-validation. This performance provides a benchmark that we would ideally like to achieve using only those features which can be computed

in polynomial time.

To do so, we performed importance analyses of the features and built a classifier using a subset of the features, first disregarding the GW-dependent features. With this collection of features, the obtained accuracy was essentially as good: 0.8236 and with a recall (intuitively the ability of the classifier to find all the class 1 instances) of 0.7917. We analyzed the feature importance shown in Fig.3.4, which shows that not only the GW-specific features can be dropped, but also that the NP-hard graph-theoretic features are unnecessary. So we can discard features (ii) and (iii) in the second regime, keeping the green-highlighted ones – and these are all computationally efficiently computable features, yielding an efficient classifier. For completeness, we did train and test the classifier using only these features, resulting in an accuracy of 0.8236, as expected.

Further, we note that the reported performance is very good from a ML/classification point of perspective (in an absence of other well-established benchmarks).

In more detail, we report that the density and the ratio between the largest eigenvalues of the adjacency matrix of the graph seem to be most important for deciding criterion (2), while it seems that demanding the gap of 2% between GW and QAOA significantly diminished the explicit dependence on the GW features, which was so prominent for criterion (1). To further investigate this, we constructed a partial dependence plot, which revealed that regular graphs with more nodes are over-represented in the high-performance regime.

Further, those cases correspond to the case when the largest eigenvalue of the Laplacian matrix is at least 1.05% times the second largest eigenvalue.

In summary, from our 82% accurate ML model, we found interesting graph properties which influence whether a QAOA approach will result in a high-performing heuristic. This is mainly dependent on the Laplacian spectrum and the density. Using ML explainability, we quantified the best significantly handled graph instances. Specifically, the density and the ratio between the largest eigenvalues of the adjacency matrix of the graph were the most informative features in this regime. This highlights a possible direction for further analytic and experimental investigations of QAOA performances.

3.5 Discussion

In this chapter, we raise the problem of algorithm selection when quantum approaches, specifically the QAOA algorithm, are considered. We were interested in detecting in-

Weight	Feature
0.2036 ± 0.2481	density
0.0816 ± 0.2094	log_laplacian_ev_ratio
0.0367 ± 0.1394	log_norm_laplacian_ev4
0.0296 ± 0.0926	log_norm_laplacian_ev5

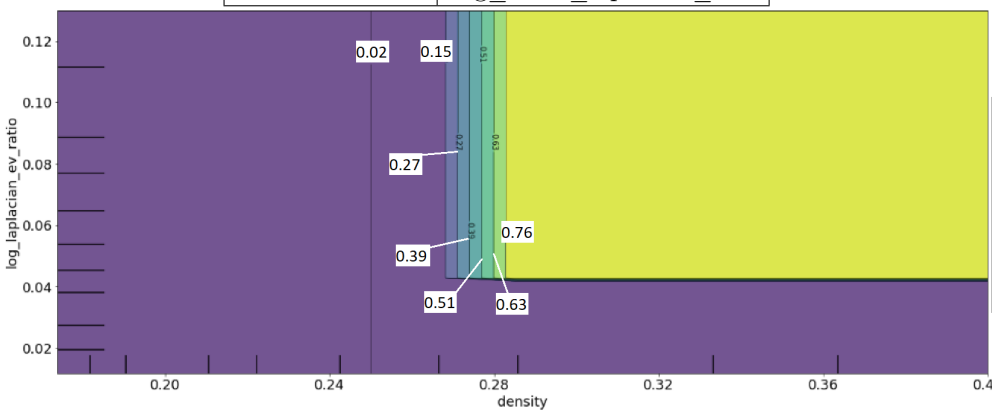


Figure 3.4: Partial dependence plot for the discriminator of significant gain with QAOA with its permutation importance. The model used the 4 features displayed in the table with a larger weight meaning a more important feature. The yellow box corresponds to probabilities higher than 76%, so likely to be labeled 1 or be in the second regime. The contour lines from right to left represent probability thresholds as follows: $0.76, 0.63, 0.51, 0.39, 0.27, 0.15, 0.02$.

stances where quantum advantage over a classical algorithm is obtainable, as characterized by two regimes: when QAOA simply outperforms the classical algorithm, in this case, the Goemans-Williamson algorithm run on the MaxCut problem (the first regime); and a practically more significant regime, where the quantum heuristic yields a "very-high-quality" output, while significantly beating the classical algorithm (second regime). Our choices of criteria were guided by practical and pragmatic considerations, based on our analysis of a dataset of graphs.

The algorithm selection problem boils down to the design of a classification algorithm, which can efficiently detect whether a given graph satisfies the first, and/or the second criterion. To this end, we developed a machine learning approach for these types of tasks. Our choice to resort to machine learning was motivated by a number of considerations; first, machine learning methods are flexible, and offer confidence that our approach can equally be applied when using QAOA (or other quantum algorithms, for that matter) in other regimes (e.g., when considering real-world noise), when com-

pared other classical algorithms. Second, we provided simple complexity-theoretical arguments about why exact decisions are likely computationally intractable. Third, using ML methods allowed us to employ explainability techniques to identify features of graphs that make them better suited for a quantum treatment, which may guide other more theoretical research.

Finally, to our knowledge before this work, the use of ML techniques to analyze the performance of quantum methods has only previously been employed in [128] (albeit for a different purpose); yet, we believe ML will play an ever-increasing role in helping us identify interesting quantum heuristics, so we hope our work may motivate more studies in this direction. For instance, machine learning methods of the type we propose could conceivably be used to identify and characterize what types of datasets are best suitable for quantum-variational-circuit-based classification, which is arguably one of the key questions in the NISQ-oriented quantum-enhanced machine learning domain.

In the process of analyzing QAOA and GW performance, we found further evidence that QAOA can provide genuine advantages over the GW algorithm– specifically, we have shown that at depth $p = 10$, QAOA outperforms GW on most instances of 4-regular graphs up to size 24. With respect to predicting advantages, we constructed a model yielding an accuracy of 96% for the first regime – the key features here depend on the output of GW, and other features were less significant. For the second regime, we achieved a model with 82% balanced accuracy and further used explainability methods to elucidate which graph features influence QAOA performance the most. In this regime, spectral properties of the graph, and basic graph density were most influential. We note that already the accuracy of 82% (4-fold cross-validated) would yield substantial optimizations in the use of quantum resources in any larger-scale optimization effort, of the type we may expect in, e.g., industrial applications.

We see numerous possibilities for future work. First, it would be interesting to perform a similar analysis on different graph families and identify when to use QAOA. Second, we could consider using Graph Sparsification [8], which converts a weighted graph into a sparser one preserving all cuts up to the multiplicative error. This can be done classically in near-linear time, and quantumly even in sublinear time. We would then study if the performances of both GW and QAOA are improved on the sparser instance. Third, an obvious limitation to this type of study is the size of the graph we can handle; to this end, it would be interesting if divide-and-conquer type methods explored in [62] can be utilized to increase the size of datasets we can consider. Additionally, in our work, we did not tackle the important question of how

the choice of QAOA optimization procedures influences the advantage gained. It is also important to consider how realistic, or real, noise effects affect performance. Note that since we employ general machine learning methods, we expect that a similar level of results could be obtained. Finally, more recent contributions also provide new modifications of QAOA to make it more suitable for standard real-world optimization objectives [114, 14]. We believe our methods can equally be applied in those settings.

Features used

Common features related to regular graphs and the spectrum of the Laplacian matrix

- density: percentage of the number of edges in a complete graph with the same number of vertices. For regular graphs, this is strongly correlated to the number of vertices.
- log_norm_laplacian_ev1: logarithm of largest eigenvalue of the laplacian normalized by the degree.
- log_norm_laplacian_ev2: logarithm of the second largest eigenvalue of the laplacian normalized by the degree.
- the same for the third, fourth and fifth largest eigenvalues.
- log_laplacian_ev_ratio: logarithm of the ratio between the two largest eigenvalues.
- spectral_gap: the second smallest eigenvalue of the laplacian.

(ii) Set numbers for graphs

- independence_number_over_number_edges: cardinality of a largest independent set of nodes in the graph normalized by the number of edges.
- matching_number_over_number_edges: cardinality of a maximum matching (size of a maximum independent edge set) normalized by the number of edges.
- diameter_over_number_edges: diameter is the maximum eccentricity normalized by the number of edges.

- domination_number_over_number_nodes: number of vertices in a smallest dominating set divided by number of nodes.
- zero_forcing_number_over_number_nodes: minimum cardinality of a zero forcing set divided by number of nodes.
- power_domination_over_number_edges: minimum cardinality of a power dominating set divided by number of nodes.

(iii) Features related to the relaxed solution of the semi-definite program in GW and the random projection routine

- percent_cut: ratio between C_{rlx} and the number of edges.
- percent_positive_lower_part_relaxation_solution: percentage of elements that are positive in the relaxed solution after Cholesky factorization.
- percent_close1_lower_part_relaxation_solution: percentage of elements that are less than .1 in absolute value in the relaxed solution after Cholesky factorization.
- percent_close3_lower_part_relaxation_solution: percentage of elements that are less than .001 in absolute value in the relaxed solution after Cholesky factorization.
- expected_costGW_over_sdp_cost: the expected cost over the 1000 random projections per instance divided by the cost of the relaxation C_{rlx},
- std_costGW_over_sdp_cost: the standard deviation of the cost approximated using 1000 random projections divided by C_{rlx}.

Chapter 4

Unsupervised strategies for setting optimal parameters in Quantum Approximate Optimization Algorithm

QAOA stands out as a promising approach to tackling combinatorial problems. In Chapter 3, we assumed we found good parameters by optimization. However, finding these appropriate parameters is in general a difficult task [23]. In this chapter[1], we study a few unsupervised machine learning approaches for setting these parameters without optimization. These angle-finding strategies can be used to reduce calls to quantum circuits when leveraging QAOA as a subroutine. We showcase them within a recursive algorithm featuring QAOA as a subroutine called Recursive-QAOA. We applied it for MaxCut over 200 Erdős-Rényi graphs limiting the QAOA depth to 3 where the number of QAOA parameters used per iteration is limited to 3. We obtain similar performances to the case where we extensively optimize the angles, hence saving numerous circuit calls.

[1]Contents of this chapter are based on [138];Charles Moussa, Hao Wang, Thomas Bäck, and Vedran Dunjko. Unsupervised strategies for identifying optimal parameters in quantum approximate optimization algorithm. *EPJ Quantum Technology*, 9(1), 2022.

4.1 Introduction

QAOA exhibits a few properties that make it interesting for combinatorial optimization such as a perfect theoretical performance at infinite depth [57], a sampling advantage [58] and the concentration of parameters [26]. The latter suggests that optimal parameters found for one instance can be reused in another. Most importantly, this means we can reduce the classical optimization loop and number of calls to a quantum device (saving runtime of QAOA-featured algorithms).

Many works have studied or illustrated this concentration property [26, 207, 100, 111, 61, 167, 4, 187, 46]. However, in many algorithms which feature QAOA as a subroutine [113, 74, 139, 178, 29], many distributions of instances are generated and several areas of parameter concentrations may arise. Hence, balancing between finding good QAOA parameters and reducing circuit calls will be key to QAOA-featured algorithms.

In this chapter, we apply unsupervised learning for setting QAOA angles, namely clustering. Our main contributions are as follows:

- We consider different approaches for the problem of setting QAOA angles with clustering: using directly the angle values, instance features, and the output of a variational graph autoencoder as input to the clustering algorithm.

- We analyze our methods by comparing them on two types of problems: Max-Cut on Erdős-Rényi graphs and Quadratic Unconstrained Binary Problems on random dense matrices.

- We demonstrate that our techniques can be used to learn to set QAOA parameters with respectively a less than $1 - 2\%$ reduction (in relative value) in approximation ratio in cross-validation while reducing circuit calls.

- We show that leveraging instance encodings for angle-setting strategies yields better results than using angle values only.

- Finally, we demonstrate their usage in Recursive-QAOA (RQAOA) [29] up to depth 3 on the Erdős-Rényi graphs. We limit the number of QAOA circuit calls per iteration to 3 (in contrast to a de novo optimization which would require many more calls), and achieve a 0.94 median approximation ratio. With our approaches, we obtain similar performances to the case where we extensively optimize the angles, hence saving numerous circuit calls.

The structure of the chapter is as follows. Section 4.2 provides the necessary background and related works. Section 4.3 analyses the optimal angles found in both problems, pointing to concentration effects and the suitability of clustering. Section 4.4 shows different unsupervised learning strategies using different data encoding for clustering and the comparison between them. Section 4.5 sums up our experiments on RQAOA. We conclude with a discussion in Section 4.6.

4.2 Background

4.2.1 QUBO and QAOA

Quadratic Unconstrained Binary Optimization (QUBO) problems are specified by the formulation $\min_{x\in\{0,1\}^n} \sum_{i\leq j} x_i Q_{ij} x_j$ where n is the dimensionality of the problem and $Q \in \mathbb{R}^{n\times n}$. QUBO can express an exceptional variety of combinatorial optimization (CO) problems such as Quadratic Assignment, Constraint Satisfaction Problems, Graph Coloring, and Maximum Cut [106]. The QAOA algorithm [57] was designed with the goal to tackle CO problems as seen in chapter 2.4.

An interesting property of the algorithm is the concentration of the QAOA objective for fixed angles [26] due to typical instances having (nearly) the same value of the objective function. Additionally, the QAOA landscape is instance-independent when instances come from a «reasonable» distribution (with the number of certain types of subgraphs of fixed size themselves concentrate, which in turn implies the values concentrate). Hence, we can focus on finding good parameters on a subset of instances that could be re-applied to new ones, with a few extra calls to the quantum device in order to refine. As stated earlier, in the most general case, characterizing distributions which are «reasonable» may be involved, or even characterizing the distribution at hand may be hard. Previous work [26, 187, 46] referenced [4] reported concentrations over optimal parameters even when QAOA is applied on random instances. These distributions over optimal parameters are empirically shown to behave non-trivially with respect to n. [4] pointed out this problem as «folklore of concentrations» .

Hence, even though angles concentrate in many settings asymptotically, for finite-size problems, different areas of concentration may rise. Therefore, choosing good angle values is challenging, especially when considering the runtime of quantum algorithms. As such, some studies built on this property and resorted to using Machine Learning (ML) or characterizing instances by some properties for finding good QAOA parameters. We present a few of them in the next subsection.

4.2.2 Related Work

Many previous works have extensively employed the concentration property [207, 100, 111, 61, 167, 4, 187, 46]. Among them, a few employed ML or designed strategies for setting good QAOA parameters for different objectives. In [100], a simple kernel density model was trained on the best angles and instances solved by QAOA to exhibit better QAOA optimization than the Nelder-Mead optimizer. Parameter fixing strategies for QAOA are also studied in [111, 207] where the best-found angles at depth p are used as starting points for depth $p+1$ before using a classical optimizer.

[187] present a strategy to find good parameters for QAOA based on topological properties of the problem graph and tensor network techniques. [61] point out that the success of transferability of parameters between different problem instances can be explained and predicted based on the types of subgraphs composing a graph. Finally, meta-learning is used in [167] to learn good initial angles for QAOA. They focused on initialization-based meta-learners in which a single set of parameters is used for a distribution of problems as initial parameters of a gradient-based optimizer. The meta-learner is a simple neural network that takes as inputs some meta-features of the QAOA circuit to predict the angles to apply (depth and which angle to output the value). However, no instance-related features are involved in their work.

In our case, we focus on clustering with the goal of proposing many parameter values to try for new QAOA circuits. In contrast to all the approaches we discussed above, we do not use a classical optimization loop after setting them. Hence, our approaches allow balancing between circuit calls of small quantum computers and performances. Such settings for instance naturally occur in divide-and-conquer-type schemes to enable smaller quantum computers to improve optimization [113, 74, 139, 178], or in Recursive-QAOA [29] as we demonstrate later.

4.3 Revisiting the concentration property

In contrast to previous related works, we propose unsupervised approaches that also exploit these concentration effects. We take a data-driven approach where from examples of good angles, we will infer new good angles for new instances. Namely, we use clustering in order to obtain clusters that can be used to reduce calls to the quantum device to small numbers (in our case, less than 10) when applying QAOA on new instances, without further optimization.

We take a usual ML approach to this problem. First, from generated instances,

we apply *exploratory data analysis* [85] (EDA) that suggests clustering may be a good approach for recommending good angles to new instances. Namely, we look at the density of angle values and apply t-distributed stochastic neighbor embedding (t-SNE) [191] for visualizing concentration effects. t-SNE is a nonlinear dimensionality reduction technique for mapping high-dimensional data to a lower d-dimensional space (typically $d \in \{2, 3\}$). Briefly, this method constructs a probability distribution to measure the similarity between each pair of points, where closer pairs are assigned with a higher probability. Then, in the lower-dimensional space $\mathbb{R}^d$, we use a Student t-based distribution to quantify the similarity among the embeddings of the original data points. Finally, the optimal embeddings are chosen by minimizing the Kullback–Leibler divergence between the similarity distributions in the original and the lower-dimensional spaces. We follow by explaining how clustering is used in order to recommend angles for new instances. The approaches we outline differ in input to the clustering algorithm. We consider clustering from the angle values directly but also from instance encodings. Finally, we compare these approaches allowing us to provide recommendations for their usage.

4.3.1 Data generation

We generated two datasets that show different concentration behavior. The first one consists of 200 Erdős-Rényi graphs for MaxCut problems. The graphs have $10, 12, 14, 16$, and 18 nodes. We utilized the following probabilities of edge creation: $0.5, 0.6, 0.7, 0.8$. We have generated 10 graphs per number of nodes and probability. The second dataset consists of 100 instances of QUBO problems, specified by their weight matrix Q (20 per aforementioned number of nodes). Their coefficients are sampled uniformly in $[-1, 1]$. For the purpose of computing approximation ratios, we are interested in C_{opt} – the maximal value of the MaxCut (or QUBO) – over all possible bit configurations, and as a reference, this was computed using brute-force. Our experiments were achieved using a classical simulator.

We then obtained for each problem the best set of angles by running the BFGS optimizer [35] 1000 times for $p = 1, 2, 3$, and selecting the ones which achieve the best QAOA objective. BFGS with random restarts is deemed a very good optimizer for continuous differentiable functions [77]. These angles are saved as a database and apply unsupervised approaches to learn to set optimal angles for unseen instances. Our approach is clearly optimization method-specific but can be applied to other state-of-the-art optimizers. Different optimizers would give different data (as the optimizers

could fail to find the optimal QAOA parameters) but they can be combined and one would select the best set of angles found among all considered.

4.3.2 Exploratory Data Analysis

Having obtained the optimal angles, we apply EDA to observe concentration effects. We look at their corresponding performance ratios using the average cost yielded by QAOA for angles γ, β denoted with $E_{\gamma,\beta}(C)$. For MaxCut on unweighted Erdős-Rényi graphs, we compute approximation ratios as $\frac{E_{\gamma,\beta}(C)}{C_{opt}}$. This value is upper-bounded by 1, which is the optimal value. For QUBOs, we compute optimality gaps $\frac{C_{opt}-E_{\gamma,\beta}(C)}{C_{opt}}$ as the optima were all negative and the closer to 0, the better. We show boxplots in Fig 4.1 the ratios wrt depth. Increasing depth results in better ratios.

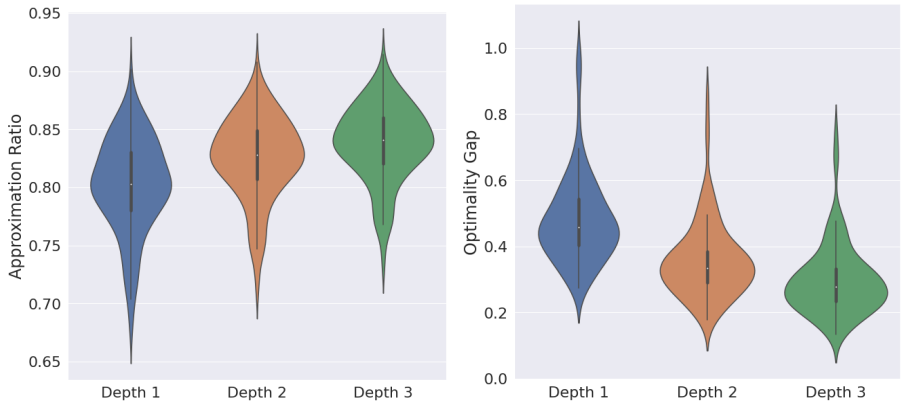


Figure 4.1: Violin plots of ratios on MaxCut and optimality gaps over QUBOs (bottom plot) for $p = 1, 2, 3$. The respective median by depth is $0.802954, 0.827901, 0.840478$ for MaxCuts and $0.457434, 0.335144, 0.278984$ for QUBOs, illustrating improved performances with increased depth.

Next, we looked at the distribution of γ_i, β_i values. Fig 4.2 shows that the concentration per each parameter is significant since their corresponding density functions are quite peaky. Also, we also observed multiple clusters of angles as the density functions are multimodal. Finally, we applied t-SNE with two components to visualize the angle values in $2D$ for $p = 2, 3$. This highlights potentially a number of clusters for each depth and problem. Note that it may be possible that we may not obtain global optima with these angles, or know if they are unique.

We notice that the probability of edge creation, represented by a different color, does not seem to influence the clusters. For dense QUBOs, we observe one important

cluster and a few instances that start to form another. Finally, in the case of the dense instances, we witness a more important spread in angle values at depth 1. This can be explained by differences between instances. Although the concentration effect is present, such an order of magnitude will impact the performances of parameter-setting strategies, and make an interesting playground to benchmark them.

Using clustering techniques can then reveal potential areas of QAOA angle values where good angles can be found to try on new instances. The angle values related to clusters can be used as recommendations for new instances. This becomes interesting as this enables lowering runtime and allow comparing based on function evaluations, or on the number of quantum circuit calls, in algorithms where QAOA would be used as a subroutine.

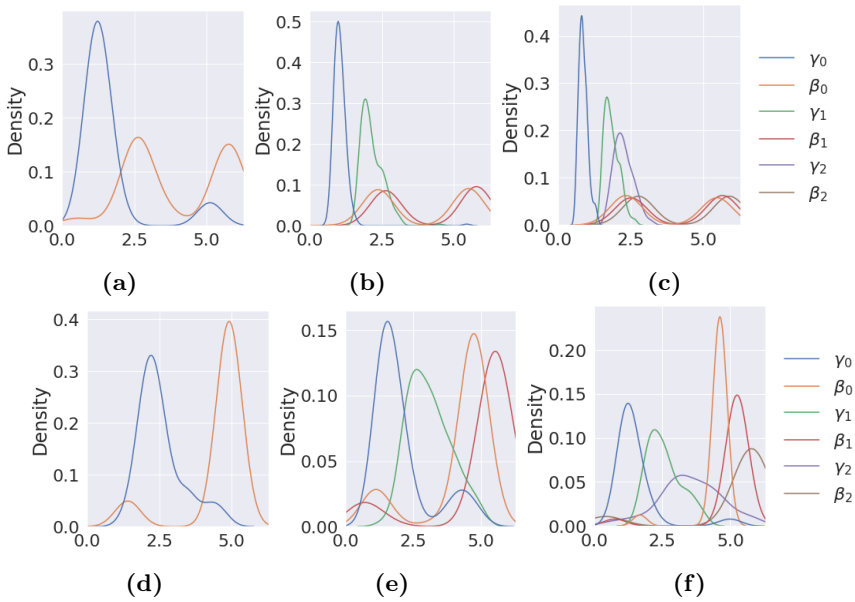


Figure 4.2: Distribution of angle values γ_i, β_i for each depth. Plots a), b) and c) concern MaxCut problems while the others refer to the dense QUBO matrices. We witness concentration effects of the angle values, suggesting the suitability of clustering as an angle-setting strategy.

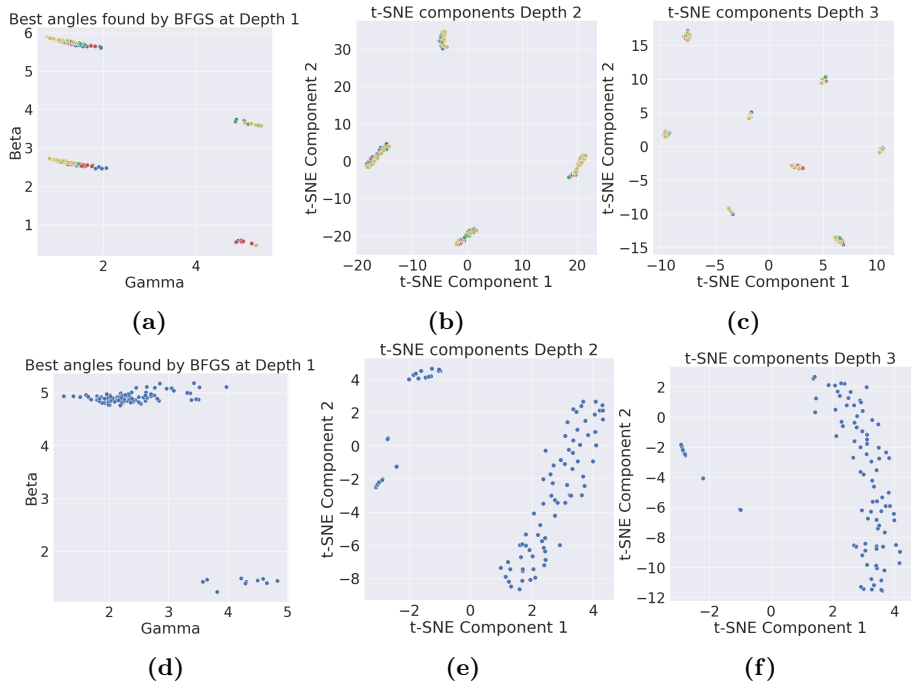


Figure 4.3: 2D angles visualization γ_i, β_i for each depth. Plots a), b) and c) concern MaxCut problems while the others refer to the dense QUBO matrices. For $p = 2, 3$, t-SNE is applied for projecting the angle values to 2D. Different areas of concentration are revealed again. We use different colors for differentiating the probability of edge creation of the Erdős-Rényi graphs, showing no correlation with clusters.

4.4 Clustering-based (unsupervised) learning for angles

As the EDA highlights a clustering effect, we propose different clustering approaches that use different data for angle recommendations. Namely, we describe first using the angle values directly for building clusters serving as angles to try. Then, we switch to using instance-related features. Finally, for the unweighted case, we use graph auto-encoders whose outputs can be used for clustering instead of computing graph features. In the following, we detail each clustering approach for flexible angle recommendation.

4.4.1 Identifying clusters of angles or problem instances

We first considered clustering using angle values. Given a database of optimal angles for Q problem instances $\{I_1, \cdots, I_Q\}, \{(\gamma^*, \beta^*)_1, \cdots, (\gamma^*, \beta^*)_Q\}$, this can be seen as computing or selecting a good set of angle values the database to apply on new instances. In this case, we do not use the problem instances during clustering. Given a user-specified number of angles to be tested K, this set of angle values is then applied to new QAOA circuits. To specify them, we can use a clustering algorithm on the database $\{(\gamma^*, \beta^*)_1, \cdots, (\gamma^*, \beta^*)_Q\}$. For instance, K-means [118] will output centroids to use directly as angle recommendations for QAOA on new instances. The K-means algorithm aims to partition a set of n data points x_i into K disjoint clusters C, characterized by the mean/centroid of the points within a cluster, denoted μ_j. The partition $P = \{P_1, P_2, \ldots, P_K\}$ $(\forall i \neq j \in [1..K], P_i \neq \emptyset, P_i \cap P_j = \emptyset, \cup_i P_i = \{x_i\}_{i=1}^n)$ is chosen by minimizing the within-cluster sum of squares, i.e., $\arg\min_P \sum_{i=1}^K \sum_{x \in P_i} ||x - \mu_i||^2$, where the centroid $\mu_i = |P_i|^{-1} \sum_{x \in P_i} x$. The algorithm iteratively updates the centroids by assigning each data point to its nearest centroid and computing the mean, until convergence.

To incorporate knowledge from instances when recommending angles, we change the data fed to the clustering algorithm. We distinguish computing instance features from learning an embedding, that is a user-defined $F-$dimensional representation or encoding of the instances as data. We denote an encoding of an instance I_t as $f(I_t)$. The angle recommendation framework using a clustering algorithm for such instance representation is presented in Alg. 1. First, clusters are learned from the encodings extracted from training data. Then, we find the instances in the database that are the closest in distance to the clusters, and their corresponding optimal angles [2]. The latter are then used for QAOA circuits on new instances, from which we keep the best QAOA output.

4.4.2 Instance encodings

In this work, we show two main approaches to encoding the instances for clustering. First, we computed a set of features following [52, 139]. Such features were used in [52] to decide among classical heuristics to solve MaxCut and QUBO problems. Inspired by [52], the features were also used for choosing when to apply QAOA against a classical approximation algorithm [139]. For Erdős-Rényi graphs, we took the graph

[2] Since the clustering algorithm outputs encodings that do not contain QAOA angle information, we use the QAOA angles of the closest training instances to the clusters.

Algorithm 1: K-angle recommendation framework for QAOA. Comments are provided in curly brackets to explain the key concepts with the associated pseudocode.

Input: Clustering algorithm, number of clusters K,
Training Data: $\{I_1, \cdots, I_Q\}; \{(\gamma^*, \beta^*)_1, \cdots, (\gamma^*, \beta^*)_Q\}$,
Testing Data: $\{I_1^{'}, \cdots, I_R^{'}\}$,

Initialize $anglesToRecommend = [], encodings = []$
for $t = 1$ **to** Q **do**
  Compute $f(I_t)$ and append to *encodings*. {First, we compute the encodings from the training instances before applying the clustering algorithm.}
end for
Apply Clustering algorithm on *encodings*
for $c = 1$ **to** K **do**
  Get encoding of cluster f^c from the clustering algorithm
  Get closest point in *encodings* to f^c and extract index i_c {Here, we find the instances in the database that are the closest in distance to the clusters, and their corresponding optimal QAOA angles.}
  Append $(\gamma^*, \beta^*)_{i_c}$ to *anglesToRecommend*
end for
{Finally, the QAOA angles in *anglesToRecommend* are used for QAOA circuits on the test instances, from which we keep the best QAOA output.}
for $t = 1$ **to** R **do**
  $bestOutput_t = \inf$
  for $c = 1$ **to** K **do**
    Apply QAOA on $I_t^{'}$ with $anglesToRecommend[c]$ {We can then compute the QAOA expectation for the test instance $I_t^{'}$ using the QAOA angles $anglesToRecommend[c]$, denoted $E_{anglesToRecommend[c]}(C_{I_t^{'}})$.}
    if $E_{anglesToRecommend[c]}(C_{I_t^{'}}) < bestOutput_t$ **then**
      $bestOutput_t = E_{anglesToRecommend[c]}(C_{I_t^{'}})$
    end if
  end for
end for

density, the logarithm of the number of nodes and edges, the logarithm of the first and second-largest eigenvalues of the Laplacian matrix normalized by the average node degree and the logarithm of the ratio of the two largest eigenvalues. For QUBOs, we reduced them to the MaxCut formulation and used the logarithm of the number of nodes, and the weighted Laplacian matrix eigenvalues-based features.

We also show how to use graph embeddings using Graph Neural Networks (GNNs) [205], avoiding the need for the user to compute the features. We employ

the Variational Graph Auto-Encoders (VGAE) [104]. This technique only works on unweighted graphs by its design principle. Consequently, we only applied it to the MaxCut instances later in this work. A VGAE learns latent embeddings $\mathbf{Z} \in \mathbb{R}^{N \times F}$ where F is the dimension of the latent variables and N the number of nodes. Given the adjacency matrix A and nodes feature vector X, the model outputs the parameters of a Gaussian distribution μ, σ for the latent representation generation. We feed to the model the Erdős-Rényi graphs, and we add as node features the degree of the nodes. Once learning is completed, we compute the embeddings by a common average readout operation [205, 196]. The latter operation can be defined as averaging the node embeddings for a graph with vertex set $\mathcal{V}$ $\frac{1}{|\mathcal{V}|}\sum_{n\in\mathcal{V}} Z_n$. This allows having a fixed dimension F for the encoding to be used by a clustering algorithm.

Having defined different strategies for clustering, we apply them to the data we generated and compare their performances. In the following section, we present our results obtained by taking a Machine Learning approach, starting from a simple baseline and cross-validating each method.

4.4.3 Results

In this section, we apply the above-mentioned proposed strategies to the generated data where EDA revealed different areas of concentration. As the first baseline for the angle-setting strategy, we experiment with simple aggregation of angle values (median and average). Then we follow this up with K-means applied on angle values by varying the number of clusters from 3 to 10 as the underlying clustering algorithm. Finally, we change the K-means data to cluster based on instance encodings instead of angle values. We computed first a set of graph features that were used in a previous study [135]. Then we investigate graph autoencoders to learn the encodings of the Maxcut instances. We cross-validate each method using 5-fold cross-validation where we report the ratios $\frac{(C_{opt}-E_{\gamma,\beta}(C))}{(C_{opt}-E^{cluster}_{\gamma,\beta}(C))}$ on test instances. A value higher than 1 would mean that the average cost yielded by clustering has improved over the one found by optimization. We also consider the case where one trains on smaller instances to apply to the bigger ones.

From angle values

As a simple baseline, we compute the average and the median of the optimal angles from the database $\{(\gamma^*, \beta^*)_1, \cdots, (\gamma^*, \beta^*)_Q\}$. From depth-aggregated results, averaging the angle values yielded a median ratio of 0.524 for MaxCut and 0.672 for QUBOs,

while taking the median values increased it to respectively 0.950 and 0.941. This can be explained by the fact that the median value is statistically more robust than the mean when handling data sets with large variability.

As expected with K-means, increasing the number of clusters yielded better median ratios. With $K = 10$, the median ratios are 0.998 and 0.985 on each dataset, a less than $1 - 2\%$ reduction in performances w.r.t. the optimal angles. Fig. 4.4 shows the improvement with an increased number of clusters. We observe also that with increased depth, median ratio performances are reduced. We conjecture that, when the dimension of the parameter space increases, more clusters are naturally needed to ensure a sensible recommendation.

Also, such a deterioration of performance w.r.t. circuit depth is more substantial on the QUBO instances than on the MaxCut ones, which can be explained by the clustering patterns in the MaxCut scenario being more significant and regular (Fig. 4.3). In addition, this observation suggests that for future work, for dense QUBO instances where the cluster center is not representative of all points pertaining to it, it is more reasonable to take a supervised learning method, which takes the problem instance as input and predicts the optimal angle values.

We also observed that, for the MaxCut problem, the cluster centroid of K-means can be quite distant from the data points when the number of clusters is small and the circuit depth is high. Particularly, this phenomenon deteriorates the median ratio by ca. 30% for 3 and 4 clusters with $p = 3$. Hence, we decided to take the closest data point to the centroid in each cluster as the recommendation, which solves this issue. For QUBOs, using the cluster centroids directly yields better results.

Overall, increasing the number of angles attempted will improve the quality of the QAOA output. Clearly, the results with less than 4 clusters present examples where the ratio is low, worsening the median performances. For instance, with 3 clusters on QUBOs, the median ratio is 0.915. In the context where the budget of quantum circuit calls is very limited, this could be problematic and call for more robust approaches. To this end, we consider using instance features for clustering.

From instance encodings

To witness whether using instance features can improve the quality of clustering, we divided the ratios obtained with instance features by the ones using angle values. We show these results in Fig. 4.5 and Fig. 4.6 where we can clearly see better ratios with less than 4 clusters, and similar results on average otherwise.

As for learned encodings or embeddings with auto-encoders, the GNN model con-

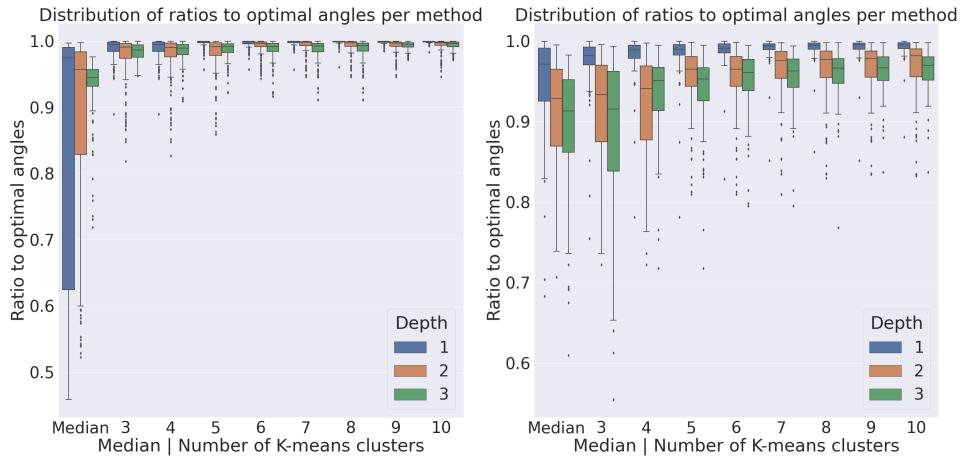


Figure 4.4: Boxplot visualization of ratios to optimal angles' expectation value per clustering method and depth on MaxCut (a) and dense QUBOs (b), when using the angle values. We show also the boxplots when computing the median angle values, yielding a median ratio of 0.524307 for MaxCut and 0.671572 for QUBOs. The median ratios are respectively 0.990579, 0.995610 and 0.998293 for 3, 5 and 10 clusters. For QUBOs, we get 0.956099, 0.970842, and 0.984787 taking the same number of clusters. With reference to the optimal angles' expectation value, this corresponds on average to a less than $1-2\%$ reduction in performances when using 10 clusters. For MaxCut, we had to use the closest data point in the dataset to the cluster, as it results in better performances. For instance, with 3 clusters at $p = 3$, the median ratio was 0.618275.

figuration we use is the same two-layer graph convolutional layer as [104]. Namely, the first one has 32 output-dimension using the ReLU activation function. This is followed by two 16-dimensional output layers for the generation of the latent variables. We train using Adam with a learning rate of 0.01 for 100 epochs and batch size set to the dataset size. Our implementation uses the Deep Graph Library (DGL) [196]. The embeddings obtained by averaging are of dimension $F = 16$. This allows having a fixed dimension for the encoding as input of the same K-means strategy described above. We observe in Fig. 4.7 that the results are similar to the ones obtained using instance features. Yet, in some instances, we see better results. Hence, many clustering results can be combined to improve the performances in ratios canceling each other's weaknesses at the cost of trying more angles to find the best ones. As future work, we could also decide which heuristic to use depending on a given test instance by using a ML model.

Finally, our approaches can save numerous circuit calls compared to de novo op-

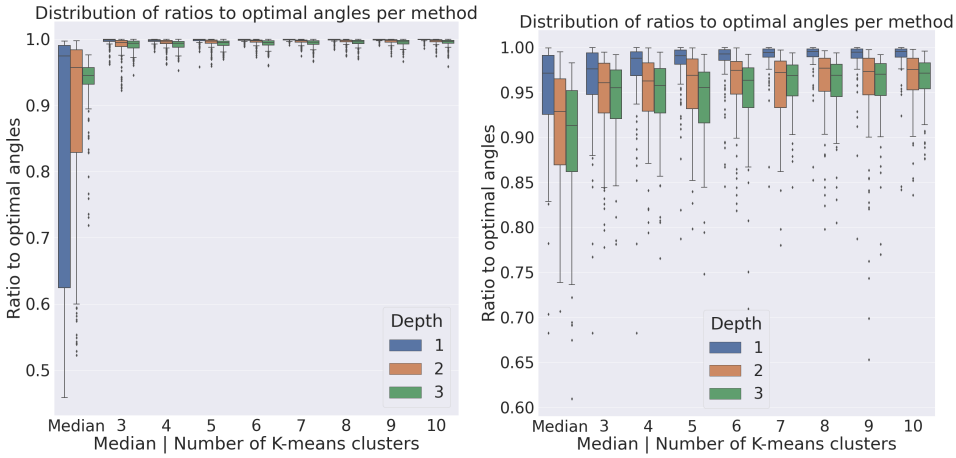


Figure 4.5: Boxplot plot visualization of ratios to optimal angles per clustering method and depth on MaxCut (a) and dense QUBOs (b), when using instance features. For Erdős-Rényi graphs, K-means yielded ratios $0.996214, 0.996368$ with $3, 4$ clusters and 0.998429 with 10. On dense QUBOs, we obtained respective median ratios of $0.963129, 0.971778$ and 0.982964. With reference to the optimal angles' expectation value, this corresponds on average to a less than $1 - 2\%$ reduction in performances when using 10 clusters.

timization. The median numbers of circuit calls for the BFGS runs giving the best QAOA angles were $56, 150, 320$ for each depth respectively on MaxCut and $44, 132, 252$ for QUBO, while in the cluster approach, the number of calls is always the cluster size, which is considerably smaller than the cost of BFGS. Instance size does not seem to affect the number of circuit calls by BFGS. In our approaches, we limited circuit calls to 10 and we do not need multiple restarts.

4.4.4 Aggregating results

Following the presentation of the different clustering approaches, we compare their performances to determine which approach works best. We propose to take the Empirical Cumulative Distribution Functions (ECDF) of the ratios as the performance measure to compare those different approaches. Given a sample $\{r_i\}_{i=1}^{R}$ of the ratios and a value of interest $t \in [0, 1]$, ECDF is the fraction of the sample points less or equal to t: $F(t) = \frac{1}{R}\sum_i \mathbb{1}_{[0,r_i]}(t)$, where $\mathbb{1}$ denotes the indicator function, which returns one only if $t \in [0, r_i]$ and zero otherwise. They enable us to aggregate the results of the different numbers of clusters and depths. A better method will have more proportion

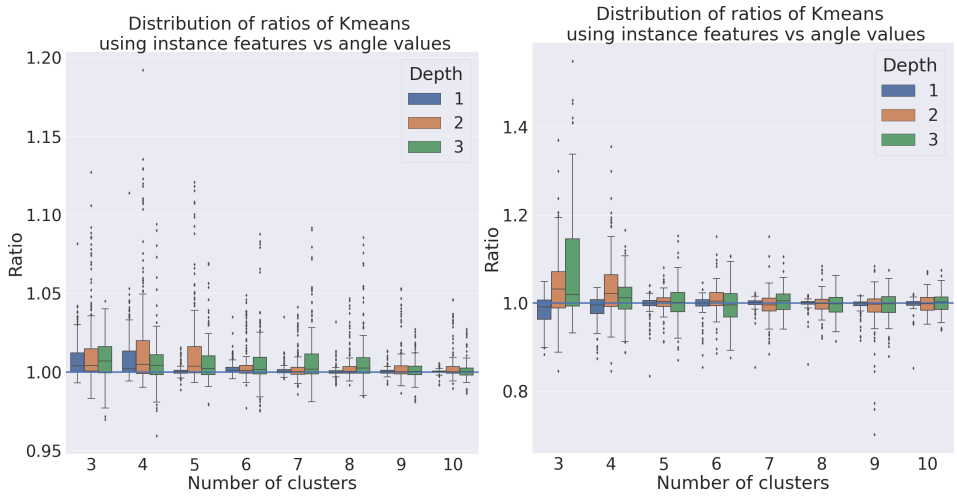


Figure 4.6: Boxplot of ratios comparing K-means with instance features against angle values on MaxCut (a) and QUBOs (b). A value higher than 1 (highlighted by a horizontal line) means using instance features results in better QAOA objective. We see an overall improvement with $3, 4$ clusters mainly at $p = 3$.

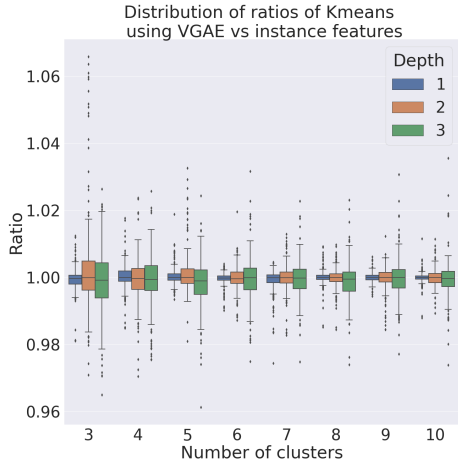
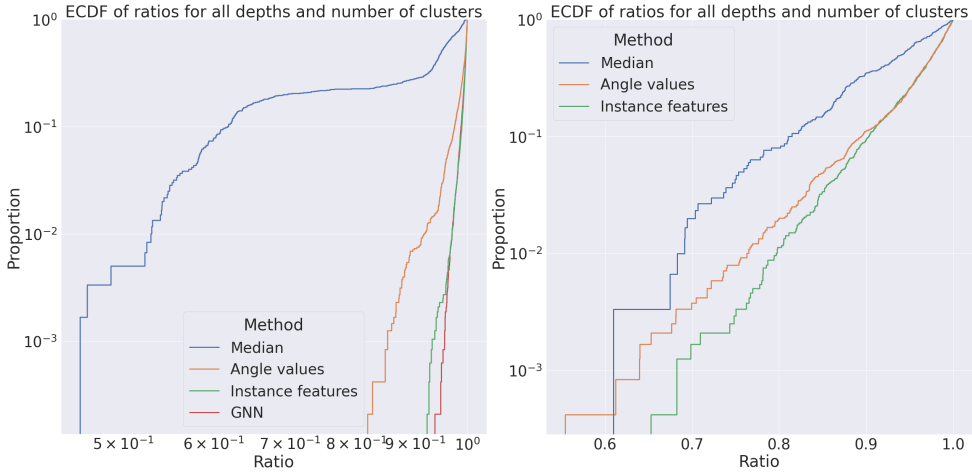


Figure 4.7: Boxplot visualization of ratios on MaxCut obtained using Variational Graph Auto-Encoders compared to using instance features. A value higher than 1 means using VGAE results in a better QAOA objective. Overall, performances are similar as the ratios are close to 1 on average.

of higher ratios, resulting in an ECDF curve located more to the right. From Fig. 4.8,

we observe that *using instance encodings is more successful in yielding better angles than using the angle values*. This is also witnessed in Fig. 4.9 with increased depth and a low number of clusters. Also, VGAE seems to be slightly better than instance features on the MaxCut problems. However, these methods can complement each other, especially as we do not need to increase the dataset size. Hence, combining them at the cost of circuit calls becomes an option for running QAOA, as we showcase with RQAOA in the next section.



(a) **(b)**

Figure 4.8: Empirical cumulative distribution functions of ratios to optimal angles' for all depths and number of clusters. A lower curve for an approach means better results when using it aggregating depths and number of clusters. We see for MaxCut (a) and QUBO problems (b) that instance features achieve better results, VGAE being competitive with instance features. When using 3 clusters, using VGAE on MaxCut instance and instance features for QUBOs leads to better ratios.

4.4.5 Case when test instances are bigger than training instances

One important consideration of these methods is to analyze scaling. This is relevant in settings where one is interested in solving larger instances given small ones. In our case, we apply these approaches in the case $K = 3$ by a $60-40\%$ train-test split. From Fig. 4.10 and 4.11, we find similar conclusions with respectively VGAE on MaxCut and instance features on the QUBO problems yielding better results. Note that we did not use the logarithm of the number of nodes and edges as features when using

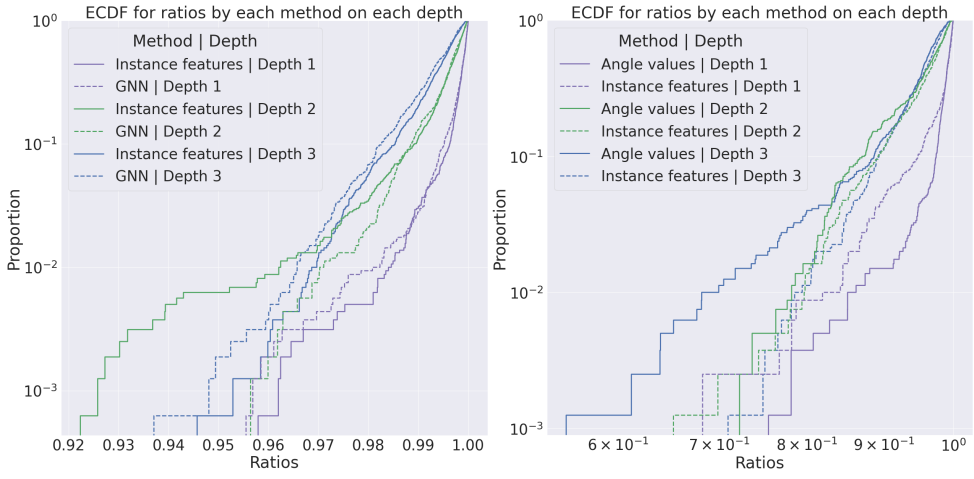


Figure 4.9: Empirical cumulative distribution functions of ratios to optimal angles per method and depth. The lower the curve, the better the method. In most cases, the curve corresponding to instance features was lower (except for QUBOs (b) at $p = 1$, and VGAE's curve was more competitive at $p = 2$ for MaxCut (a)). This was also the case when using 3 clusters.

instance features as the values between training and test are too different.

4.5 Demonstration with RQAOA

RQAOA [29] is a recursive algorithm where, given an Ising problem $\sum_{i,j} w_{ij} Z_i Z_j$, one starts by applying QAOA on the former. The quantum state output $|\gamma, \beta\rangle$ is then used to compute correlations $M_{ij} = \langle \gamma, \beta | Z_i Z_j | \gamma, \beta \rangle$. Then, variable elimination is carried out by selecting a pair of variables satisfying $(i_l, j_l) = \arg\max |M_{ij}|$, and substituting Z_{j_l} with $\text{sign}(M_{i_l,j_l}) Z_{i_l}$ in the Ising formulation. This reduces the number of variables by 1. We then get a new reduced problem and we reiterate the procedure for a user-defined number of iterations. The choice of iteration fixes the size of the final instance which is then solved using a brute-force (or some other classical) approach, and the substitutions are used onto it to obtain a final solution.

As RQAOA requires optimizing many QAOA instances that iteratively shrink in size, we demonstrate the application of our clustering approaches in this context. We do so for the MaxCut problems where we limit the number of iterations to half of the size of the Erdős-Rényi graphs. We do not consider the dense QUBOs as RQAOA

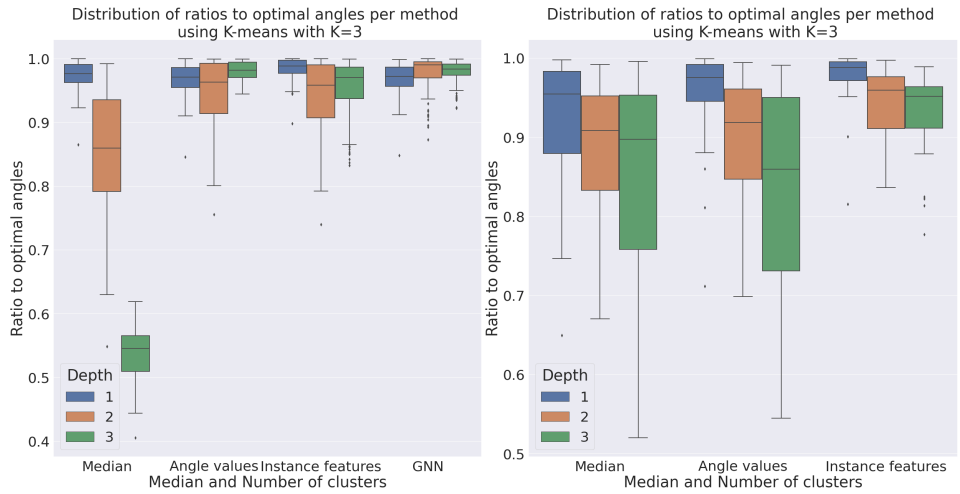
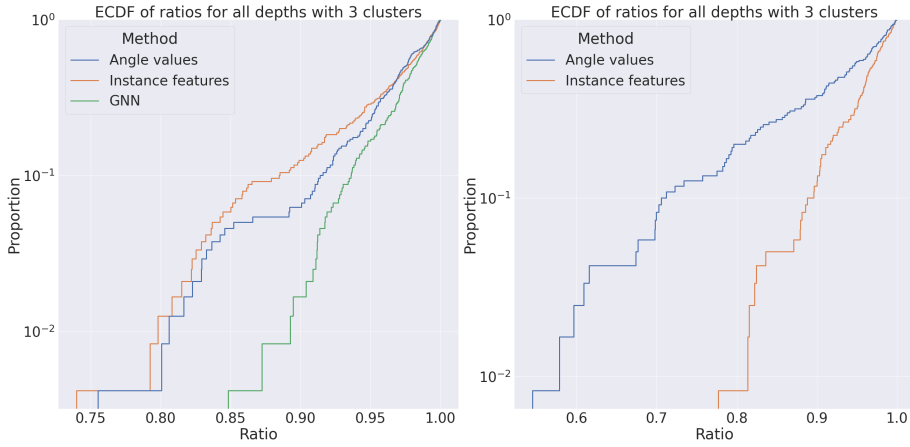


Figure 4.10: Boxplot of ratios comparing K-means $K = 3$ using instance features against angle values on MaxCut (a) and QUBOs (b). The ratios are obtained from 40% of the instances with the highest number of nodes. From depth-aggregated results, on MaxCut, using the median values gives a median ratio of 0.859316, 0.928519 with angle values, 0.976959 with instance features, and 0.981618 using VGAE. On QUBOs, we obtained respectively 0.926136 for the median of angle values, 0.936679 clustering with angle value and 0.963677 with instance features.

would reduce an original dense graph to non-dense intermediate subproblems not part of the database. As per the number of QAOA parameters attempted per iteration, we limit it to 3 and apply the three clustering approaches: angle-value, instance features, and VGAE-output based. We do so by using our previous database and training each method on all instances to get 3 QAOA parameter recommendations. The latter are then used for QAOA on the RQAOA-generated instances.

Fig. 4.12 shows that with the three approaches, we obtain a median 0.94117 approximation ratio with RQAOA. The minimal ratio obtained is 0.8367 and the optima were found on 33 instances. When looking at each method independently, we observe that the angle-value clustering performances at $p = 3$ are lower than the others. This is due to the fact that we use the K-means clusters directly as it allowed us to find more instances with a ratio of 1. Graph features and VGAE seem similar in performance, with a small advantage at depth 2 for VGAE. Looking at the frequencies where the best ratio by instance was obtained, VGAE is more successful. Respectively, each method achieves the best-found ratios over 88, 118, and 165 instances. Finally, we also tried

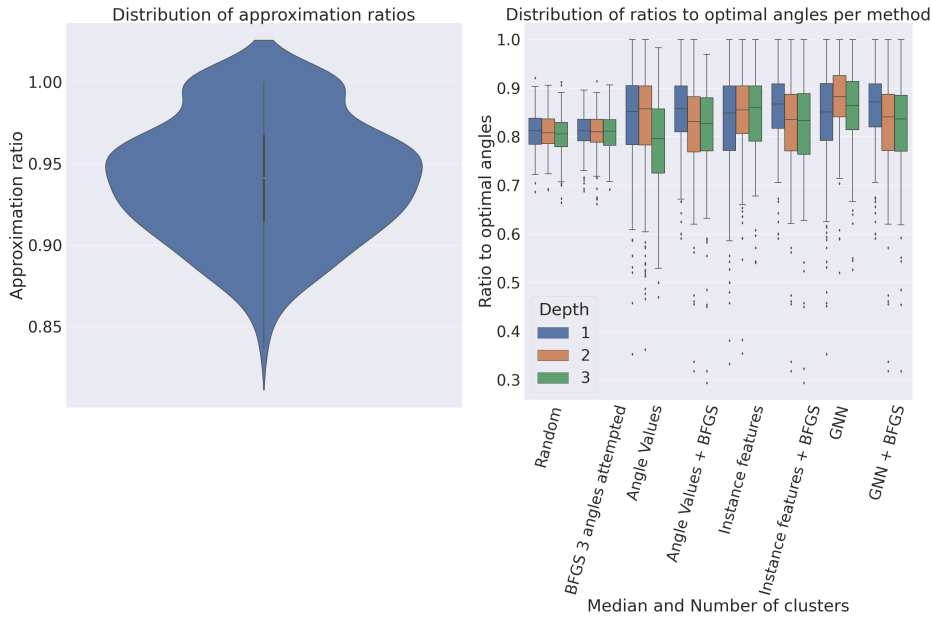


(a) **(b)**

Figure 4.11: Empirical cumulative distribution functions of ratios to optimal angles. The ratios are obtained from 40% of the instances with the highest number of nodes. Similar results to Fig. 4.8 and Fig. 4.9 are obtained.

using random angles, by sampling uniformly values in $[0, 2\pi]^p$ and optimizing further the angles from each approach with BFGS up to 100 iterations maximum. We clearly see better performances with clustering approaches compared to random angles. This is also the case when using BFGS (starting with random angles) limited to 3 circuit calls when optimizing, the same budget as our clustering-based approaches. Dividing the MaxCut ratios obtained with BFGS with the ones without further optimization yielded a median value of 1. Hence, the results were similar to the BFGS-optimized approaches, saving many circuit calls.

To conclude, our unsupervised approaches can be used to run quantum algorithms where QAOA is used as a subroutine. They are then considered as hyper-parameters that can be tweaked to achieve better performances for QAOA-featured algorithms, depending on a user-defined budget definition. In our RQAOA showcase, the maximal depth of QAOA, as well as the number of parameters to try at each iteration, was set to 3, and optimizing further did not improve. For MaxCut on Erdős-Rényi graphs, leveraging VGAE in RQAOA achieved the best ratios over 82.5% of the instances.



(a) **(b)**

Figure 4.12: The violin plot visualization represents ratios obtained on MaxCut using all three unsupervised approaches, using just 3 circuit calls per RQAOA iteration, without optimizing further with BFGS. A median ratio of 0.94117 was obtained. The boxplots represent the MaxCut ratios obtained using each approach. We added using random angles per iteration as a baseline as well as using BFGS (starting with random angles) with a budget of 3 angle values attempted during optimization, and we witness clustering approaches yielded better results. When dividing the ratios of the methods with the ones obtained by adding BFGS, we obtain a median ratio of 1, meaning we saved many circuit calls for similar results with clustering.

4.6 Discussion

In this chapter, we study different strategies for fixing the parameters of QAOA based on unsupervised learning. We focused on clustering given previous works highlighting the concentration property and exploratory data analysis of the best angles found for MaxCut on Erdős-Rényi graphs and dense QUBOs. We however use a methodology closer to machine learning by cross-validating compared to related work.

Furthermore, we demonstrated that these techniques can be leveraged to restrict the number of QAOA circuit calls to small numbers (less than 10) with a less than $1-2\%$ reduction in approximation ratio on average from the best angles found when cross-validating. We also showed how to compare different clustering strategies and

that leveraging instance encodings (by computing features or computing them with a model, in our case a VGAE) for angle-setting strategies yields better results than using angle values only. Although the VGAE embedding-based is quite competitive, we recommend using the simpler instance features in practice since the VGAE brings extra computation overhead. For generalization, in regard to the problem scale, both instance features- and VGAE-based clustering approaches manage to retain the performance for unseen problem instances larger than the training set. For dense QUBOs, increasing the clusters is less impactful compared to MaxCut, in which we conjecture that the clusters in QUBO are of large spread and less separable, hindering the performance of the clustering approach in higher dimensions. For both problems, it is necessary to increase the cluster number to retain a good performance when the circuit becomes deeper.

From an application perspective, we envision these techniques to be employed in algorithms where QAOA is run on a small part of the problem to solve such as divide-and-conquer [113, 74] and iterative algorithms [139, 178, 29]. Restricting to a few numbers of circuit calls will help decrease the runtime of quantum-featured or quantum-enhanced algorithms, making them closer to competing with classical heuristics. We showcased our approach in the context of Recursive QAOA as hyperparameters under a limited budget (QAOA depth and number of QAOA parameters per iteration limited to 3), where we were able to achieve a 0.94 median approximation ratio. With our approaches, we obtain quite comparable performance to the case where we extensively optimize the angles, hence saving numerous circuit calls.

For future work, other clustering techniques can be studied and extended to predict the angle values by instance in a semi-supervised approach, and for different problem instances. Plus, ML can be used to decide which heuristic to use depending on a given test instance. We also did not apply GNN to the dense QUBOs as graph autoencoders are mostly applied to unweighted graphs. Using VGAE that can reconstruct graph adjacency and node features is then another research direction. Since we use unsupervised methods, we expect the same methodology to be used on noisy hardware. Studying different approaches to resilience under different noisy settings would be also considered of main interest. Finally, these approaches can be studied within different QAOA-featured algorithms and under different settings (depth of QAOA, number of clusters, and instances properties to name a few).

Chapter 5

Tabu-driven quantum neighborhood samplers

Combinatorial optimization is an important application targeted by quantum computing. However, near-term hardware constraints make quantum algorithms unlikely to be competitive when compared to high-performing classical heuristics on large practical problems. The small size of instances used in previous chapters 3 and 4 illustrates such an issue. One option to achieve advantages with near-term devices is to use them in combination with classical heuristics. In particular, we propose using quantum methods to sample from classically intractable distributions – which is the most probable approach to attain a true provable quantum separation in the near term – which are used to solve optimization problems faster. In this chapter, we numerically study this enhancement by an adaptation of Tabu Search using QAOA as a neighborhood sampler[1]. We show that QAOA provides a flexible tool for exploration-exploitation in such hybrid settings and can provide evidence that it can help in solving problems faster by saving many tabu iterations and achieving better solutions.

[1]Contents of this chapter are based on [139];Charles Moussa, Hao Wang, Henri Calandra, Thomas Bäck, and Vedran Dunjko. Tabu-driven quantum neighborhood samplers. In Christine Zarges and Sébastien Verel, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 100–119, Cham, 2021. Springer International Publishing.

5.1 Introduction

While numerous works have been studying various theoretical and empirical properties of QAOA [26, 46, 207, 186, 135], many practical challenges remain. Indeed, only small-sized problems and very limited p can be run on real hardware, which severely limits the quality of the solution obtained empirically [79, 14, 114]. Many open questions still remain, e.g., regarding the comparison of QAOA with other heuristic methods on various cases of instances that stem from particular problem domains, with different optimizers, and with varying levels of experimental (or simulated) noise. One reason of why so many uncertainties remain is that classical simulation is computationally very expensive, and quantum devices are still scarce to prevent large real-world tests [127, 79].

In contrast to optimization problems, quantum advantage has been demonstrated in sampling [11]. Indeed, theoretical results establish a quantum advantage in producing samples according to certain distributions of constant-depth quantum circuits [27]. In this direction, it has been demonstrated that the sampling of the QAOA circuit, even at $p = 1$, cannot be efficiently simulated classically [58]. The above considerations point to a possibility of utilizing sampling features of QAOA for neighborhood explorations with the added benefit that, since the neighborhood may be limited to fewer variables, a smaller quantum device may already lead to improved performance of a large instance.

It is interesting to delve into sampling aspects in the domain of classical local search algorithms, where we seek the optimum in the vicinity of the current solution with respect to either the original optimization problem or a subproblem thereof, using a deterministic or stochastic sampling strategy [12]. Such a sampling-based local procedure is typically realized by the combination of some parametric distribution family for drawing local trial points (e.g., the binomial or a power-law distribution [48]) and a selection method for choosing good trial points, and hence the overall outcome of this procedure results in the family of sampling distributions [47, 199, 112].

In this chapter, we use QAOA circuits as local neighborhood samplers, having malleable support in (many) good local optima but still allowing a level of exploration (which is desirable since local optima may not lead to global optima). This introduces the topics of sampling and multiobjective aspects of QAOA that allow balancing between exploration and exploitation. To this end, we study its combination with tabu search (TS), a metaheuristic that has been successfully applied in practice for combinatorial optimization by local search. Moreover, to control the trade-off between

exploration and exploitation, we add this critical component in TS to the specification of the standard QAOA circuit.

Contributions - We construct an algorithm incorporating QAOA in TS with the usual attribute-based short-term memory structure (a.k.a the tabu list). With our approach, we kill two birds with one stone: we gain quantum enhancements, while the local properties of tabu search can make the required quantum computations naturally economic in terms of needed qubit numbers, which is vital in the near-term quantum era. We analyze and benchmark this incorporation with *small QAOA depths* against a classical TS procedure on QUBO problems of up to 500 variables. We also propose a penalized version of QAOA incorporating knowledge from a current solution. We find that QAOA is often beneficial in terms of saved iterations, and can find shorter paths toward better solutions. The structure of the chapter is as follows. Section 5.2 provides the necessary background on TS and interplay with quantum techniques. In section 5.3, we detail the TS procedure incorporating a short-term memory structure with QAOA. The results of our simulations are presented in section 5.4. We conclude with a discussion in section 5.5.

5.2 Background

Tabu Search (TS) [66] is a meta-heuristic that guides a local heuristic search procedure to explore the search space beyond local optimality. One of the main components of TS is its use of adaptive memory, which creates a more flexible search behavior. Such a framework allows using a quantum algorithm as a local search tool, for solving large instances with limited-sized quantum devices. Various works leveraged TS for solving QUBOs [105, 65, 147, 67, 148] using short-term and long-term strategies used during the search. We note also different hybrid settings that combine a basic TS procedure with another framework such as genetic search [120] and Path Relinking [198]. TS was also incorporated with quantum computers to tackle larger problems beyond their limitations. Indeed, finding methods to leverage smaller devices is of main importance. Many divide-and-conquer approaches have been designed for quantum circuits and algorithms [50, 159, 30, 150]. In this paper, the size of the QC comes into play more naturally as a hyperparameter defining the «radius» of the search space.

With respect to the interplay between TS and quantum techniques, to our knowledge TS has only been considered from the perspective of D-Wave quantum annealers, a different non-universal model of computation than the gate-based model. The first approach of this kind is an algorithm called qbsolv [25]. It starts with an initial TS

run on the whole QUBO. Then the problem is partitioned into several subproblems solved independently with the annealer. Subproblems are created randomly, by selecting variables. Non-selected ones have their values fixed (clamping values) from the TS solution. The subsolutions are then merged and a new TS is run as an improvement method. The second approach is an iterative solver designed in [162]. At each iteration, a subproblem is submitted to the annealer. The subproblem is obtained by clamping values from a current solution. A tabu list is used in which each element is a list of variables of length k. Each element is kept tabu for a user-defined number of iterations. In contrast in this work, we consider using QAOA in combination with TS.

5.3 Tabu-driven QAOA sampling

Inspired by the above-mentioned works, we use a simple TS procedure where QAOA is added in the neighborhood generation phase to solve QUBO problems. Note that we could also apply QAOA in more sophisticated frameworks, but a simpler approach is easier for understanding the benefits of QAOA with TS.

Local search algorithms explore a search space by generating sequences of possible solutions which are refined. At each step, we generate a so-called neighborhood from a current solution. In particular, if we denote the current solution x, a generated neighborhood corresponds to candidates x' that differ by at most k bits. We denote this set as $N_k(x) = \{x' \in \{0,1\}^n | \delta_H(x', x) \leq k\}$, where δ_H denotes the Hamming distance. For a simple one-bit-flip generation strategy, this corresponds to $k = 1$ and TS uses a modified neighborhood due to tabu conditions. Although increasing k could help exploration, the neighborhood generation comes at an exponential cost. But this could mean finding better solutions in fewer TS iterations, and thus also in principle overall faster if a fast good method for neighborhood exploration is devised.

This motivates the use of a quantum algorithm as a proxy for exploring $N_k(x)$. Specifically, we will use QAOA which $2p$ real parameters are tweaked in a continuous optimization scheme resulting in a probability distribution on N_k. Increasing p (assuming the optima are found over the parameters) will improve the quality of the output (likelihood of returning an actual global optimum).

However, in the case of local search, a greedy strategy that tends to select the best point in the neighborhood would not only lead to potential stagnation but also result in longer optimization time (unless the neighborhoods are already the size of the overall problem) [20]. Indeed, one may also consider modifications that impose (various) notions of locality, which are usually not considered in standard QAOA

uses where it is used for the entire instance, with the sole goal of finding optima. To this end, we first outline the basic TS procedure generally used to solve QUBO problems [67, 120, 198]. Then, we show how QAOA can be combined with the latter. Finally, we propose a modification of QAOA that balances between going for the global optimum and prioritizing local improvements relative to the current TS solution.

5.3.1 The basic TS algorithm

The basic TS procedure for solving a QUBO with objective function $f(x)$ is described in Alg. 2, for $k = 1$ excluding the green-highlighted part. It uses a simple *tabu list* recording the number of iterations a variable remains tabu during the search. A variable can be set tabu for a fixed number of iterations (denoted Tabu tenure TT) but also with a random tenure. Each iteration can be considered as updating a current solution denoted x, exploring a modified neighborhood N_1' due to the tabu considerations. Generally, x is chosen greedily when evaluating the objective function over candidates $x \in N_1'$.

For large problem instances, there exists an efficient evaluation technique for QUBO solvers leveraging one-bit flip move [64]. Let $\Delta_x = f(x') - f(x)$ be a move value, that is the effect in the objective of going to x' from x. For one-bit flip moves, we denote as $\Delta_x(i)$ the move value upon flipping the i-th variable, which can be computed using only the QUBO coefficients. The procedure records a data structure storing those move values, which is updated after each TS iteration.

Initially, all variables can be flipped (line 5). At each iteration, the tabu solution x is updated by flipping the variable that minimizes the objective over the neighborhood obtained by one-bit flip moves over non-tabu variables (lines 6-8). If the new tabu solution improves over the best-recorded solution, the aspiration is activated. In this case, the tabu attribute of the flipped variable is removed. The tabu list is finally updated (lines 18-23) and iterations continue until the stopping criterion is reached. This can be either a maximum number of TS iterations and/or a maximum number of TS iterations allowed without improvement of the best solution (*improvement cutoff*).

5.3.2 QAOA neighborhood sampling

In the usual TS algorithms, the neighborhood consists of candidates with Hamming distance one relative to the current tabu solution x. We note that sometimes considering also neighbors that are at most k-Hamming distance away from x helps in finding better solutions. The number k can be set in our case as large as the (limited) number

of available qubits in quantum hardware.

To study the exploration of such neighborhoods, a brute-force generation approach is initially tested, and thereafter replaced by QAOA. As stated before, getting the optimum for subproblems in TS may lead to getting stuck during the search. QAOA, by definition, is a flexible framework as an exploration-exploitation tool. On the one hand, QAOA generates better solutions the deeper the circuit (p) , and the better the classical optimization procedure within QAOA is. It is known it can have advantages over various standard algorithms, e.g. Simulated and Quantum Annealing [186]. To extract all advantages from the capacities of QAOA, we can further modulate the distribution of outputs it produces by limited depth or, as we present next, modifications to the QAOA objective to prioritize a more local behavior. Such flexibility is important, not only for the exploration-exploitation trade-off as it provides interesting ways to fine-tune the algorithm depending on the instance to solve.

First, the choice of variables to run QAOA on needs to be addressed. Considering the $\binom{N}{k}$ possibilities would be intractable. Variables can be chosen randomly but an approach incorporating one-bit flip move values can help in guiding toward an optimum. The k variables can be chosen amongst the non-tabu ones at each step. Plus, this means QAOA is an attempt at improving the solution one would get with the one-bit flip strategy outlined previously. One can either select the k variables greedily or add randomness by using the one-flip gains as weights for defining a probability to be chosen. For simplification, we consider the greedy selection based on one-bit flip move values. If we consider the chosen variables that were flipped, the update step of the incremental evaluation strategy can be applied. Let $l \leq k$ be the number of different bits. The newly generated candidate can be considered as a result of l sequential one-bit flips. Thus, l calls to the above-mentioned efficient procedure are required.

A second consideration concerns the tabu strategy for updating the tabu list. We choose to set as tabu the variables amongst the k chosen ones that were flipped. Choosing to flip all chosen ones could be problematic as it could lead to all variables being tabu very early during TS. An aspiration criterion can be used if the new candidate gives the best evaluation found during the search.

Finally, the question of how to run QAOA is of main importance. In our first scenario, QAOA will be run as a proxy for brute-force (with exploration properties) to optimize the subproblem defined over the k chosen variables. This is done by fixing in the QUBO the non-chosen ones from the current tabu solution x. The depth p of QAOA can be user-defined. In this work, we limit p to 2 to showcase sampling aspects

of QAOA at small depths.

Our QAOA-featured TS is outlined in Alg. 2 where QAOA addition is indicated in green shades. It starts with the same steps as with the standard Tabu search algorithm until line 8. The QAOA part kicks in from line 9 by first choosing a subset of k variables, and executing QAOA on the sub-QUBO problem where we optimize over the chosen k variables while keeping the remaining bits the same with the current point x. After obtaining the best point from QAOA (lines 11-13), we select the better one from the QAOA outcome $x^{k\text{-bit}}$ and the best one-bit flip point $x^{1\text{-bit}}$ and use it to update the current search point. The move values are then updated by l calls of the fast incremental method (line 14). Finally, if the best-so-far point is improved by the updated search point, we drop the accepted bit flips from the tabu list (line 21). Otherwise, we reset their tabu value to the sum of tabu tenure and a random tenure (line 23).

5.3.3 Enforcing locality with penalized QAOA

As a tool in local search algorithms, QAOA may be useful with modifications that impose notions of locality. We incorporate these notions in the cost hamiltonian so that they are captured during the QAOA evolution. This can be done through the cost function by adding a penalty term. A possibility is to consider the Hamming distance with a current tabu solution x. Hence, the objective for QAOA becomes:

$$\min_{x'}[f(x') + A\delta_H(x', x)], \tag{5.1}$$

where δ_H corresponds to the Hamming distance and A is a constant. The right-hand side additive term of Eq. (5.1) aims to encourage the output of candidates that differ by a few bits from the current solution if $A > 0$, and vice-versa. As a neighborhood sampler, the penalty may help in enforcing locality. However, setting the parameter A is non-trivial for activating the effect of the extra term.

There is also a possibility to add information about the fitness gain in differing from x. If switching a bit is an improved move, it would be prioritized. Our algorithm uses the one-bit flip move values Δ_x in order to select the variables to run QAOA on, from which we can construct a weighted penalty term:

$$-\tfrac{1}{2}\sum_{j=1}^{N}\Delta_x(j)(-1)^{x_j}x'_j. \tag{5.2}$$

For a minimization problem, $\Delta_x(j) < 0$ characterizes encouraging flipping the j-th

Algorithm 2: QAOA-featured Tabu Search for solving QUBO.

Input: An initial solution x_0, Cost function $f(x)$
Parameter : Tabu tenure TT, Random tabu tenure rTT, subproblem size k
Output: The best solution achieved x^*

$x^* \leftarrow x \leftarrow x_0$;
$Tabu(i) \leftarrow 0, \Delta_x(i) \leftarrow 0$ for $i \in [1..N]$;
while *stopping criterion not reached* **do**
  $x^{\text{pre}} \leftarrow x$;
  for $i \in [1..N] \cap Tabu(i) = 0$ **do**
    $x_i^{(i)} \leftarrow 1 - x_i, x^{(i)} \leftarrow x$;
    one bit-flip gain: $\Delta_x(i) = f(x^{(i)}) - f(x)$;
  $x^{\text{1-bit}} \leftarrow x^{(j)},\ j \leftarrow \arg\min \Delta_x(i)$;
  Select greedily or randomly a subset of variables $K \subseteq I$ s.t. $|K| = k$;
  Get a new QUBO by fixing the $N - k$ other variables in x;
  Run QAOA and get the best sample $\hat{x}$ minimizing the new QUBO;
  $x_i^{k\text{-bit}} \leftarrow x_i$ for $i \in I \backslash K$ and $x_i^{k\text{-bit}} \leftarrow \hat{x}_i$ for $i \in K$;
  $x \leftarrow$ the better out of $x^{\text{1-bit}}$ and $x^{k\text{-bit}}$;
  Update move values $\Delta_x(i)$ for $i \in [1..N] \cap x_i^{\text{pre}} \neq x_i$;
  aspiration $\leftarrow$ False;
  if $f(x) < f(x^*)$ **then**
    $x^* \leftarrow x$, *aspiration* $\leftarrow$ True;
  $Tabu(i) \leftarrow Tabu(i) - 1$ for $i \in [1..N] \cap Tabu(i) > 0$;
  for $j \in [1..N] \cap x_j^{pre} \neq x_j$ **do**
    if *aspiration* **then**
      $Tabu(j) \leftarrow 0$;
    else
      $Tabu(j) \leftarrow \text{TT} + \text{Random(rTT)}$;

variable in new candidates. For a candidate, in this case, $\Delta_x(j)$ is added to the cost. Conversely, $\Delta_x(j) > 0$ would result in penalizing candidates with the j-th bit value flipped. One can also multiply by a positive constant A for enforcing more the locality effects. The penalty translates in an additional operator term H_{penalty}, following an application of a usual cost operator in QAOA, where:

$$H_{\text{penalty}} = -\tfrac{1}{2} \sum_{j=1}^{N} \Delta_x(j)(-1)^{x_j} \sigma_z^j \tag{5.3}$$

The corresponding quantum circuit of depth one is very simple and given by $\bigotimes_i R_Z((-1)^{x_i}\gamma\Delta_x(i)), \gamma \in \mathbb{R}$.

5.4 Simulations

We performed extensive simulations over instances of QUBO problems publicly available in the well-known OR-Library [65, 15, 16]. In designing them, our objectives are 1) investigating whether exploring larger neighborhoods can facilitate faster convergence (in terms of TS iterations), 2) elucidating the effect of locality on QAOA output given by the introduced penalty in Eq. (5.3), and 3) studying the utility of QAOA as a proxy for brute-force.

5.4.1 Larger neighborhood exploration benefits

To study the first objective we replaced QAOA with brute-force search in Alg. 2. Starting from the all-zero initial solution, we first run the basic TS with different constant tabu tenures (rTT = 0). Then, we do the same with brute-force TS for different values of k up to 20. We study two regimes that differ in how TS search results, namely when k is not comparatively small to N on instances where $N = 20$, and when it is on instances of 100, 200, and 500 variables.

k/N relatively large

In the first regime, we assume that we can explore a large percentage (more than 25%) of the instance size greedily. As instances, we take the first eight instances of *bqpgka* (named 1a-8a consisting of 30-100 variables), for which we solve by TS. Then we select randomly 20 variables out of N and clamp values of the non-selected ones from the solutions. We do so five times per instance, resulting in 40 instances of size 20. Then TS with different k/N values for subproblems $(0.9, 0.75, 0.5, 0.25)$ were tried on this suite. When using brute-force, we set $TT = 2$ and tried many values for the basic TS.

Fig. 5.1 shows the proportion of (run, target value) pairs aggregated over all functions for 10 targets generated by linear spacing using the benchmarking and profiling tool IOHprofiler [49] for iterative optimization heuristics. The target values were normalized by the optimum of the problems. We observed that for $k/N \geq 0.5$, these instances are straightforward to solve (in 3 iterations). The case $k = 5$ required 8 iterations on one instance but managed to achieve optimality on all of them. However, the basic TS, run for 20000 iterations, failed to solve the same instance. Letting this instance aside, 5 iterations would be required for $k = 5$, and 22 for the classical TS procedure. Hence, we clearly observe, as expected, degrading performances as k/N decreases. This also enabled us to confirm numerically that a flip-gain-based approach

when considering subproblems is in general beneficial for solving QUBOs.

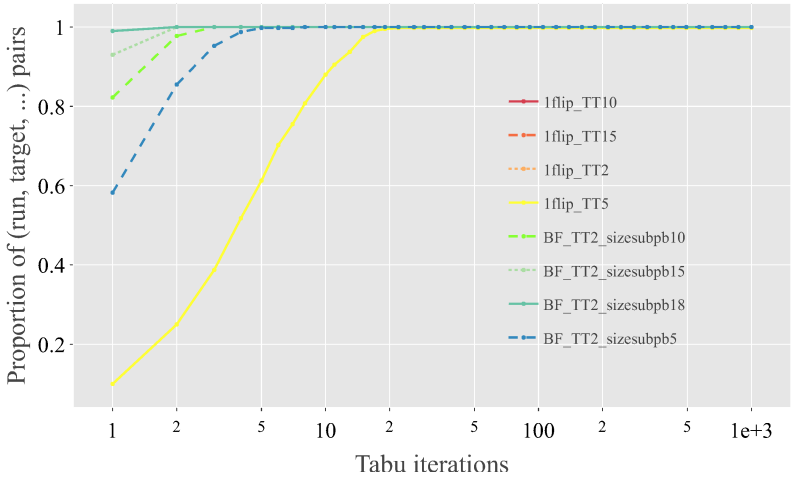


Figure 5.1: Empirical cumulative distribution function (ECDFs) of tabu iterations for each algorithm aggregated over all 20-variable problems with 10 target values evenly spanning the range of all observed function values.

k/N relatively small

In this regime, we study the case when k/N is relatively small (less than 20%). The simulations are carried out on 15 QUBO instances from bqpgka: 100-variable (named 1d-5d), 200-variable (1e-5e) and 500-variable problems (1f-5f). Algorithms are run until the best objective value found in [65] is reached or a maximum number of TS iterations is reached.

We run the algorithms with different TT values ranging from 2 to 10 and adding 15, and for $k = 10, 15, 20$. We limit the number of iterations to 20000 for the basic version and 1000 for the brute-force approach (200 for $k = 20$ though). Table 5.1 shows for the different values of TT which ones achieved the best performances in terms of target only. In these cases, we observe that we can find a k such that better solutions are found using fewer iterations, especially on the densest instances 4f and 5f. These instances, when considering the underlying graph given by the coefficients connecting different variables, have a density of respectively 0.75 and 1 (a non-zero coefficient for each pair of variables). For $k = 20$ the proposed approach achieved optimality where the basic TS failed. Again, we observe performances, in terms of target achieved, depending on setting well the tabu tenure in accordance with k. Intuitively, one could

think that the larger k, the smaller TT has to be to save iterations and achieve a better objective value. But we clearly observe counter-examples.

Table 5.1: Best values of tabu tenure (ranging from 2 to 10, adding 15) achieving the optimum obtained with the first TS iteration(s) to reach the corresponding maximum given in [65]. The best performances per instance are highlighted in bold. The mention All means all TT values reached the same solution. The NA mention means no run returns the optimum, with the best value obtained in parenthesis.

Algo.	1d (6333)	2d (6579)	3d (9261)	4d (10727)	5d (11626)
Basic	15 / 300	All / 71	<9 / 90	2 / 67	6 / 171
$k = 10$	All / 13	>2 / 61	6 / 39	2 / 31	All / 19
$k = 15$	All / 10	**2+5 / 26**	All / 8	3 / 18	5 / 42
$k = 20$	**All / 7**	5 / 37	**All / 5**	**All / 11**	**2 / 18**
Algo.	1e (16464)	2e (23395)	3e (25243)	4e (35594)	5e (35154)
Basic	9 / 419	6 / 493	10 / 190	15 / 170	6 / 238
$k = 10$	8 / 485	7 / 111	All / 21	6 / 55	All / 25
$k = 15$	**All / 13**	**6+7 / 31**	**All / 13**	3 / 41	**2 / 20**
$k = 20$	2 / 35	NA (23370) / 82	3 / 44	**3 / 25**	4 / 31
Algo.	1f (61194)	2f (100161)	3f (138035)	4f (172771)	5f (190507)
Basic	8 / 970	7 / 795	4 / 449	NA (172734) / 1148	NA (190502) / 647
$k = 10$	**15 / 213**	7 / 337	15 / 77	NA (172449) / 110	NA (190502) / 126
$k = 15$	8 / 225	**8 / 173**	15 / 139	NA (172734) / 46	NA (190502) / 383
$k = 20$	NA (61087) / 184	NA (100158) / 101	**4 / 57**	**4 / 57**	**9 / 150**

Larger k exploration, in this regime, turns out to not always be beneficial. This seems counter-intuitive when considering a target objective only, on an instance-to-instance basis comparison. Fig. 5.2 shows the proportion of (run, target value) pairs aggregated over all functions for 1000 targets generated by linear spacing. The target values were normalized by the optimum of the problems. Again, we observe that, with larger k, the proportion of successes is higher, when measured at the same number of iterations. Note that this can only be observed for $k = 20$ up to 200 iterations. The proportion for the basic TS was close to 0.9 while the brute-force approach was superior to 0.99. Hence, we can reach very good solutions with fewer iterations as k increases.

In summary, as opposed to the previous regime, the structure of the problems becomes very important and we have to look at performances in an aggregated way to

witness the benefits of exploring larger neighborhoods. Having outlined some performances given by the brute-force approach on subproblems, we switch to QAOA and study its sampling effect as a proxy.

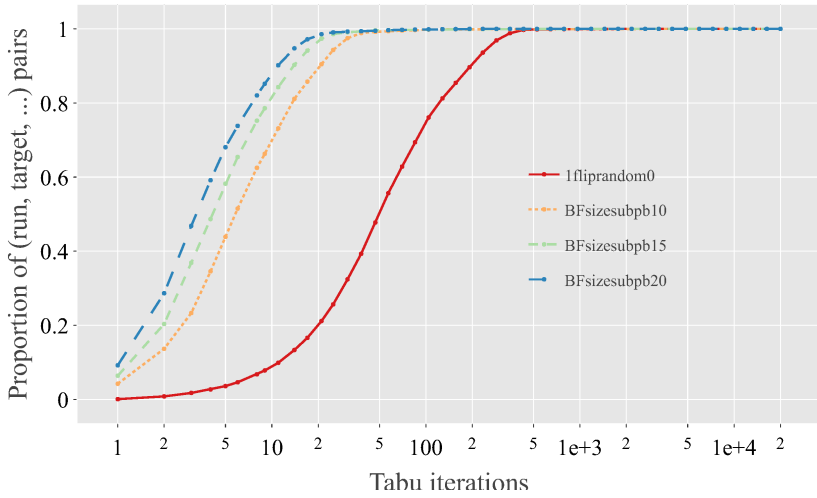


Figure 5.2: Empirical cumulative distribution function (ECDFs) of tabu iterations for each algorithm aggregated over all dimensions, all problems, and 1000 target values evenly spanning the range of all observed function values.

5.4.2 QAOA as a proxy for brute-force

The second part of our simulations studies the output of QAOA as a proxy for brute-force. To this end, we first study an example TS run from our previous simulations. We take the subproblem QUBOs obtained at each step (except the first one), and run QAOA at $p = 1$ and 2, and we study the distribution of the energy given by $|\gamma, \beta\rangle$, after optimization, with and without the penalty term.

Having outlined the properties of the QAOA output, we run Alg. 2 and study its performances in comparison to the basic TS. From the optimized angles, we try different sampling strategies to generate a candidate per iteration: just sampling once, sampling 10 times and choosing a candidate greedily, and finally considering all samples (even during optimization) greedily. The latter corresponds to a quasi brute-force (BF) approach.

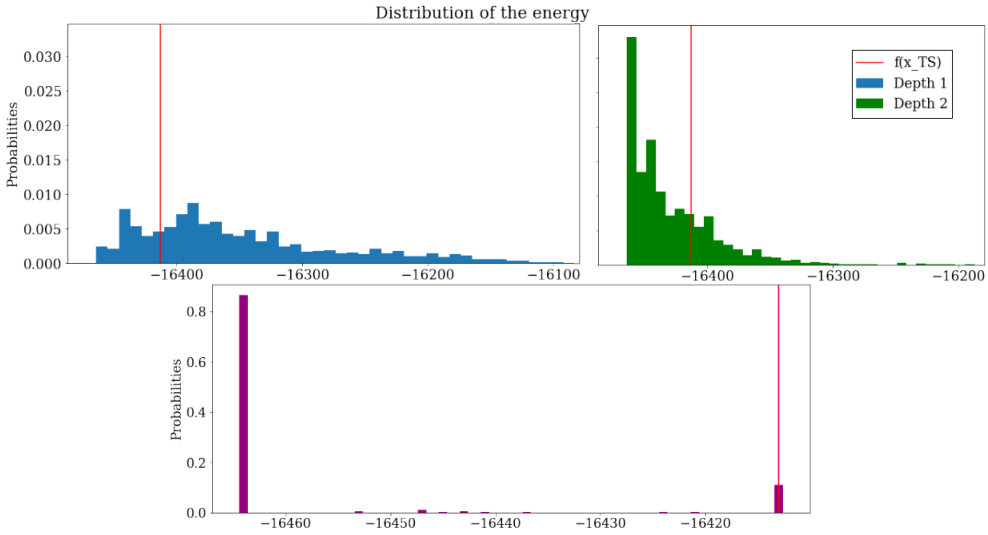


Figure 5.3: QUBO evaluation distribution given by sampling 10^5 times from the QAOA quantum state, and by running simulated annealing 1000 times (bottom), for the last TS iteration done on instance 1e.

Energy distribution of QAOA

As a first step, we study how the QAOA output distribution looks like at small depth, with the purpose of elucidating how it can help in avoiding detrimental greedy search behavior. We consider, as an example, instance 1e for which $k = 15, TT = 5$ used 13 iterations greedily. The subproblem QUBOs are kept and we run QAOA on them as previously stated. The third and last iterations are interesting when considering the penalty term. The former is a case where the optimum is obtained by flipping all bits except one and where all flip moves are favorable. The last iteration has flip moves from the current tabu solution discouraging flipping all bits. Plus, very few candidates (0.2%) improve over the tabu solution. For these iterations, we look at the quantum state given by QAOA and analyze the distribution of the QUBO evaluation (or energy in an Ising context).

Fig. 5.3 shows the distribution given by 10^5 samples from the last iteration's quantum state at $p = 1$ and 2. At $p = 1$, we observe a homogeneous spread with two major humps on the left and right sides of the tabu solution evaluation. There is a probability of 23.9% of improving from it by the quantum state. At $p = 2$, we see the distribution is shifted to lower energies, yielding an improved probability of 75%. The average energy is 16350 for QAOA $p = 1$ and 16428 for $p = 2$. Moreover, the standard

deviation of the output decreases from 79.2 to 42. This is expected as to the limit of infinite depth, QAOA converges to the optimum with less variance.

When running simulated annealing 1000 times with a temperature of 17.5 using 100 steps, we observed that, unlike QAOA, the energy spread is restrained to a few points, the optimum being most present. Decreasing slightly the temperature or the number of steps would always yield the optimum. However, in terms of exploration opportunities, QAOA could allow visiting different paths that may lead to fewer iterations required towards improved solutions.

Theory shows that increasing the depth would permit QAOA to find the optimum assuming optimal parameters are found. But by limiting the depth, we can control how other good candidates are spread from the optimum. This could engender new paths to solve a problem differently, where a suboptimal solution on a subproblem leads to an easier one for QAOA towards better candidates. Note this can be done in different ways. One way would be using brute-force and perturbating or mixing the solution, which is not efficient. We could also have the same effect with simulated annealing. But in many cases, we can fail to find the optimum and even less find a bunch of candidates around it. Finally, due to its flexibility, QAOA permits in leveraging modifications to introduce locality notions in a multiobjective scheme for local search, such as the previously mentioned penalty. In the following, we study its effect on the QAOA distribution.

Penalty effect

In section 5.3.3, we introduced a penalty term to impose notions of locality in QAOA as a local search tool. An extra operator based on the hamiltonian given in Eq. (5.3), translates to a circuit of depth one concatenated with a QAOA layer. We study its effect on the QAOA distribution obtained on the resulting sub-qubos at the third and last iteration.

On iteration 3, for both QAOA and its penalized version, the distribution tends to output candidates with the largest Hamming distances to the current point as expected. Also, the most likely candidate is the one that completely differs from the current point, which is more favored by the penalty effect. No significant changes were observed when penalizing at $p = 2$.

For the last iteration, Fig. 5.4 shows that the original QAOA at $p = 1$ results in many probability peaks compared to the penalized version, which evolves to a major peak with an increased depth. The penalized version demonstrates two major peaks, from which the optimum and a close candidate to x are preferred.

This characterizes the interplay between optimizing and penalizing. At $p = 1$, the penalized version has a better probability of improving x (0.34 vs 0.239) and a higher probability of finding the optimum (0.1 vs 0.02). However, the unpenalized $p = 2$ version was more likely to output the optimum (respectively 0.75 and 0.26, where the penalty at $p = 2$ yields 0.62 and 0.26).

In summary, using the penalty creates a balance between the greedy approach and the one-bit-flip gains knowledge from the current solution. This could result in smoothening the distribution while favouring interesting candidates for both objectives. This will modify the search path taken during TS depending on the outcome. Having studied numerically the output of the quantum state one can get with QAOA, and the penalty effect, we switch to less idealized simulations where the subproblems depend on the QAOA output during the TS search.

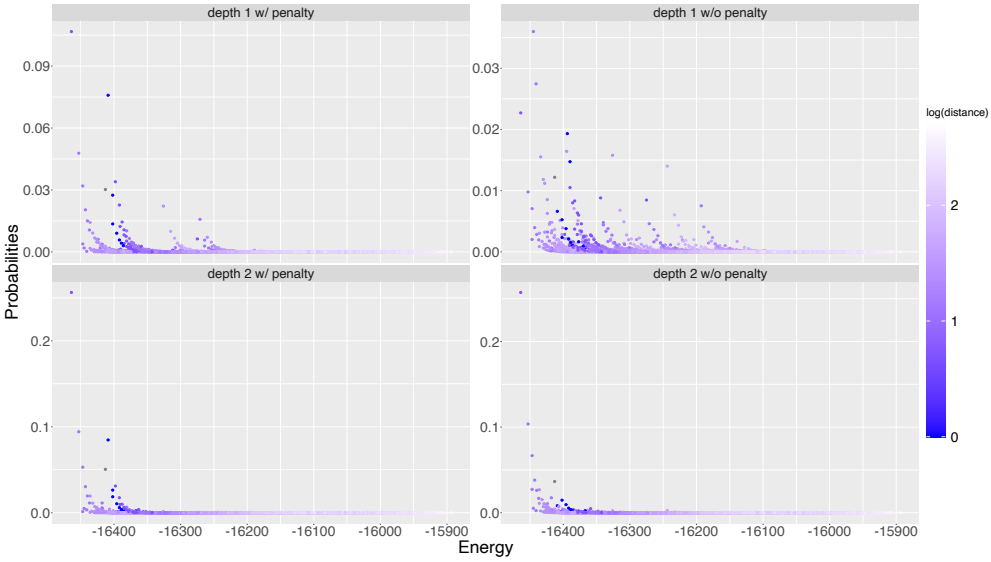


Figure 5.4: Distribution of the evaluations (or energies) for the last iteration obtained on instance 1e given by the QAOA output state, with and without the penalty term. A colormap is given for the Hamming distance with the current tabu solution.

QAOA exploration possibilities

After looking at examples of QAOA output and outlining a few possible exploration opportunities, we carried out a few extra simulations but not considering the subproblems obtained with brute-force. Hence, QAOA (and its penalized version) was called

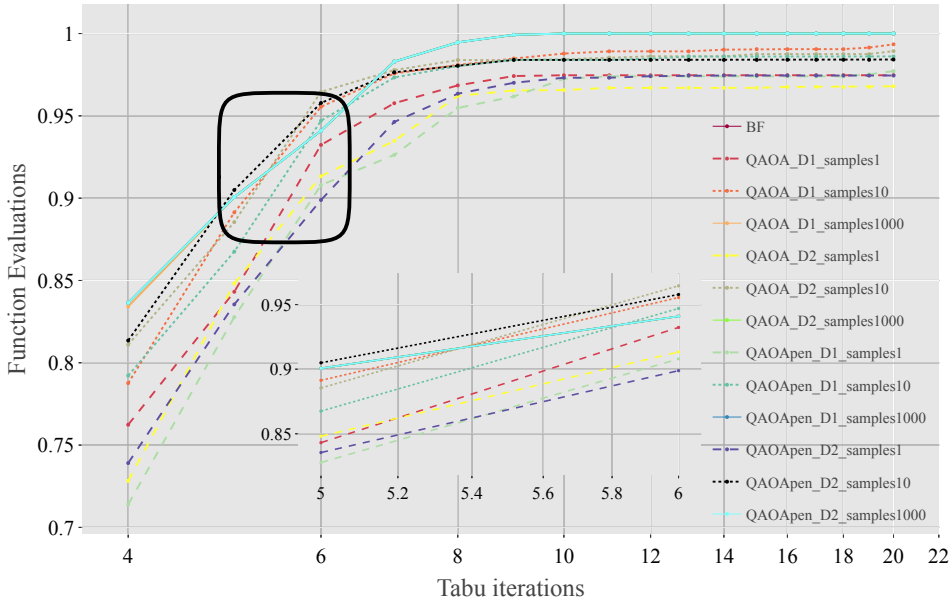


Figure 5.5: Median of best normalized evaluations achieved over TS iterations for instance 1d. The mentions D1, D2, mean respectively $p = 1, 2$ and the penalized version is indicated by «pen». A higher curve corresponds to better solutions reached. At iteration 20, the basic TS value would be 0.38, while the lowest QAOA curve value is 0.967. The $m = 1000$ runs and BF are over 0.99 starting at the 8th iteration. At iteration 5, the penalized $p = 2, m = 10$ version is slightly better than the others, even BF (0.9049 vs 0.9007). At the 6th, the $m = 10$ versions are above BF (respectively 0.9645 and 0.9578 for unpenalized and penalized $p = 2$, 0.9555 and 0.9470 for $p = 1$, and 0.9409 for BF).

once per iteration with BIPOP-CMAES [145, 75] optimizing from one set of angles [2]. In the following, we give a few examples to illustrate the exploration possibilities.

We take instances where BF simulations required few iterations and where the basic TS was beaten in target. Namely, instance 1d for $k = 15$ and $TT = 5$ and 1e for $k = 15$ and $TT = 10$. We run Alg. 2 for 10 times, limiting them to 20 TS iterations. Different numbers of samples are used for generating a new candidate per iteration: just once, 10 times, and 1000 times. The latter could be considered a quasi-brute-force approach in these runs. We denote as m the number of samples used in the following.

[2]When using BIPOP-CMAES, we run circuits with 1000 measurements to estimate expectation values. The optimizer stops when it has reached 2000 evaluations. We obtained great performances in terms of averaged ratios (as the evaluations divided by the optimum of the subproblem), superior to 0.97 at the considered depths.

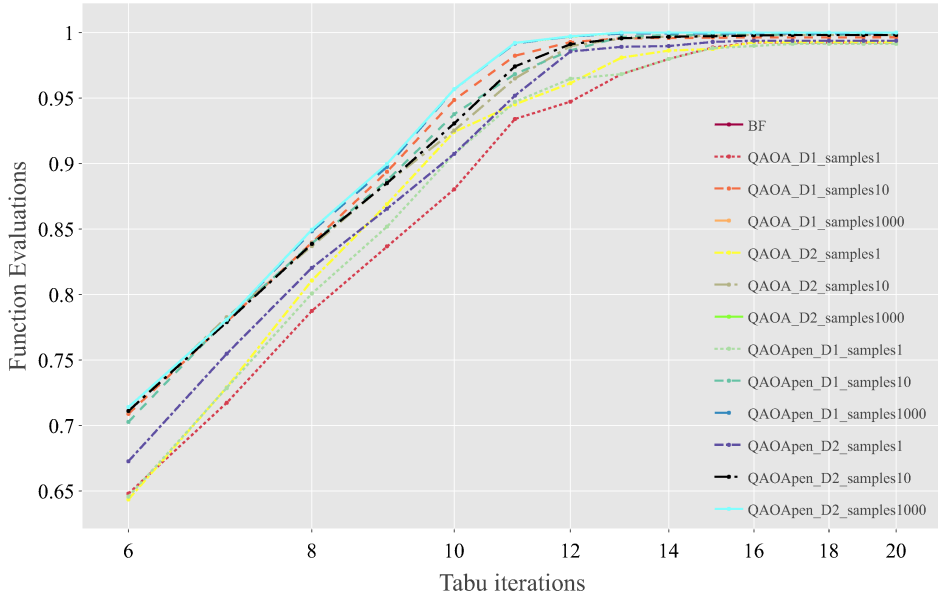


Figure 5.6: Median of best normalized evaluations achieved over TS iteration for instance 1e. At iteration 20, the lowest QAOA curve value is 0.991 (while 0.19 for the basic TS). The $m = 1000$ runs and BF are over 0.99 starting at the 11th iteration. At iteration 7, the original $p = 2$ QAOA using 10 samples point is higher than the others, even BF (0.7827 vs 0.7808). This also happens at iteration 12, with the penalized $p = 2, m = 1000$ (0.997085 vs 0.996902).

Fig. 5.6 and Fig. 5.5 show the median of the best normalized evaluation obtained per run by iteration. We observe from iteration 7 for 1d and 10 for 1e that $m = 1000$ is equivalent (in median) to the BF generation. In general, the more samples used, the better the solution found. However, we had at a few iterations median runs that achieved higher values than BF. For instance, this happened for the penalized $p = 2$ QAOA at the 5th iteration with $m = 10$ for 1d and $m = 1000$ at the 12th for 1e. We consider also the frequency of runs for which the basic TS was beaten, and the optimum was found. Table 5.2 summarizes our results. Increasing m improves the frequency of successful runs. We observe that new paths were found, mainly with an extra iteration or two but exceptionally one run or two over 10 could save one iteration. This was the case on instance 1e with 12 iterations instead of 13, exclusively with the penalized version. These examples are numerically in favor of a greedy (or quasi) approach to solving subproblems. However, QAOA allows, through a trade-off

between exploration and exploitation, to discover new paths toward optimality that are still interesting in terms of number of iterations.

Table 5.2: Frequency of successful runs in beating the basic TS and finding the optima for instance 1d for $k = 15$ and $TT = 5$, and instance 1e for $k = 15$ and $TT = 10$ (separated by /), in terms of QAOA settings (with the number of measurements noted m). We report also the number of iterations that led to the optimum.

QAOA	m	Frequency beating TS (/10)	Frequency optimum (/10)	Iterations to optimum
D1	1	0 / 0	0 / 0	
D1pen		0 / 0	0 / 0	
D2		0 / 1	0 / 1	/ 14
D2pen		0 / 0	0 / 0	
D1	10	7 / 2	1 / 2	18 / 15,16
D1pen		4 / 2	0 / 2	/ 15,15
D2		5 / 1	2 / 1	11 / 14
D2pen		4 / 0	1 / 0	10 /
D1	1000	8 / 7	8 / 7	9,10,12 / 13
D1pen		8 / 5	8 / 5	10 / 12,13
D2		9 / 9	9 / 9	10,11 / 13,14
D2pen		10/ 7	10 / 7	9,10 / 12,13

5.5 Conclusion and outlook

In this chapter, we studied sampling aspects when quantum approaches, specifically the QAOA algorithm, are considered in combinatorial optimization. We considered a practically relevant setting where a gate-based quantum algorithm, limited in the number of qubits, is utilized in a hybrid quantum-classical framework to solve large optimization instances faster. Our framework constitutes a powerful yet simple heuristic, Tabu Search, in tandem with QAOA as a local neighborhood sampler.

As a starting point, numerical experiments over open-source QUBO problems up to 500 variables validate using QAOA as a proxy to explore larger neighborhoods, under the assumption that subproblems are solved optimally. Continuing, we investigated the exploration possibilities given by QAOA output at small depth. User-defined parameters such as depth and number of measurements used to generate a candidate can be increased to favor exploitation. In our examples, solving subproblems emphasizing more on the latter gave better general performances. Yet, we found that exploration

can be beneficial. Iterations can be saved with our QAOA procedure, illustrating that missing to generate the solution of a subproblem in previous iterations could yield faster paths towards better solutions. Hence, the QAOA-based algorithm we introduce in this work becomes a very flexible tool in such hybrid quantum-classical settings.

We see numerous possibilities for future work. First, our model allows for many hyperparameters whose function needs to be explored, and, as is usually done in many local search methods, the exploration/exploitation trade-offs can be made online-adaptive. Second, the effect of real-world limitations, most importantly noise, and hardware connectivity, calls for further investigation. Although QAOA can be run on real hardware [79, 201], its output quality will improve as the quantum devices decrease in error. Angle-setting strategies such as the ones studied in chapter 4 could be incorporated to reduce the runtime. Finally, it would be interesting to propose different frameworks (e.g. [147, 148, 67, 120]) with special emphasis on the exploration possibilities given by small-depth quantum algorithms, and cross-compare with standard techniques in future works. We believe our approach combined with these types of analyses will provide new promising ways to maximize the use of limited near-term quantum computing architectures for real-world and industrial optimization problems.

Chapter 6

Performance comparison of optimization methods on variational quantum algorithms

While for QAOA one can leverage concentration properties to avoid relying heavily on a classical optimizer as discussed in Chapter 4, this is not the case for other VQAs. In the case of VQE for quantum chemistry applications presented in chapter 2.3, fast and reliable classical optimization algorithms are required. Hence, understanding and optimizing how off-the-shelf optimization methods perform is an important research area. In this chapter[1], we study the performance of two commonly used gradient-free optimization methods: CMA-ES, and SPSA. We do so on the task of finding ground-state energies of a range of small chemistry and material science problems. SPSA was used frequently as an optimizer for VQE while CMA-ES is a state-of-art optimizer for difficult optimization problems in continuous search spaces. Hence, the underlying motivation was to benchmark both optimizers. We find that, with proper hyperparameter tuning, CMA-ES is competitive with and sometimes outperforms SPSA.

[1]Contents of this chapter are based on [24];Xavier Bonet-Monroig, Hao Wang, Diederick Vermetten, Bruno Senjean, Charles Moussa, Thomas Bäck, Vedran Dunjko, and Thomas E. O'Brien. Performance comparison of optimization methods on variational quantum algorithms. *Phys. Rev. A*, 107:032407, Mar 2023.

6.1 Introduction

The performance of VQAs is dependent on the ability of classical optimization algorithms to solve such tasks. Finding their limitations for different VQA tasks is very important in research and industry applications. To this end, optimization algorithms can be benchmarked on a wide variety of systems. However, when we worked on the topic, no extensive performance comparison of the most common optimization methods existed for chemistry and material science problems yet. The Simultaneous perturbation stochastic approximation algorithm (SPSA) [182, 183] seemed most often used though as it was designed for optimization on noisy functions. As stated previously, our goal was to benchmark CMA-ES, a state-of-art optimizer for difficult optimization problems in continuous search spaces, against the more common SPSA for VQA on several chemistry and material science problems.

We first carried out hyperparameter tuning for CMA-ES and SPSA from which we conclude that the two methods are comparable in performance across many problems. One can outperform the other depending on the task. Additionally, the accuracy of the optimized parameters is investigated. For this purpose, we define a 'sampling noise floor': a limit on the accuracy that an optimizer can achieve when the optimal parameters correspond to the best-ever function evaluation. We demonstrated numerically that CMA-ES can outperform this 'sampling noise floor'.

The structure of the chapter is as follows. Section 6.2 presents the settings of running VQE. We present the optimizers and the systems considered in Section 6.3. Section 6.4 presents the results of hyperparameter tuning while Section 6.5 concerns the accuracy of the optimized parameters. Finally, we conclude this chapter in Section 6.6.

6.2 VQE methods

We have seen in Chapter 2 the typical VQA workflow and how they are applied for chemistry under the naming VQE in Chapter 2.3. We remind here how VQE works. Given a PQC preparing the state $|\Psi(\boldsymbol{\theta})\rangle$ and an observable O specifying the problem, the cost function to optimize is:

$$\mathcal{C}(\boldsymbol{\theta}) = \langle O \rangle = \langle \Psi(\boldsymbol{\theta}) | O | \Psi(\boldsymbol{\theta}) \rangle.$$

In order to measure the expectation value of O on a quantum computer, it is

typical to write O as a linear combination of easy-to-measure operators, i.e., the Pauli operators $\hat{P}_i \in \{\mathbb{I}, X, Y, Z\}^{\otimes N}$. By linearity of the expectation operator, we get:

$$O = \sum_i c_i \hat{P}_i \rightarrow \mathcal{C}(\boldsymbol{\theta}) = \langle O \rangle = \sum_i c_i \langle \hat{P}_i \rangle. \tag{6.1}$$

Such a cost function is estimated by measuring many times many circuits for each Pauli-based operator above-mentioned. Such estimation is then passed to a classical optimization algorithm to find the set of parameters minimizing $\mathcal{C}(\boldsymbol{\theta})$.

The total number of samples (also called shots) to be measured on quantum devices is usually of the order of $\sim 10^9$. Such limit calls for a balance between exploration and estimation of the cost function such that the optimal parameters are reliably obtained. In naive settings, the total number of shots per Pauli operator is fixed. We refer to this approach as one-stage optimization. Cade et al. [36] split the total shot budget between three stages, showing improved performance with VQAs.

Our numerical experiments are performed with different shot budgets of 10^7, 10^8, and 10^9. On the one-stage method, the total number of function evaluations is fixed to 10^4 and 10^3, 10^4, and 10^5 shots per Pauli operator per function call used respectively. Within the three-stage procedure, for a fair comparison, the function calls are fixed at $7150 - 2145 - 715$, and the shots per Pauli operator at every stage are $10^2 - 10^3 - 10^4$, $10^3 - 10^4 - 10^5$, and $10^4 - 10^5 - 10^6$, respectively.

6.3 Optimizers and systems considered

In this work, two gradient-free optimization algorithms are compared across multiple problems of different sizes:

1. Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [78] is a state-of-art population-based optimization algorithm using self-adaptation of its internal variables to the energy landscape.

2. Simultaneous perturbation stochastic approximation algorithm (SPSA) [182, 183] employs a stochastic perturbation vector to compute simultaneously an approximate gradient of the objective and performs well on noisy functions.

As for their implementation, we use PyCMA [76] for CMA-ES and a modified version of SPSA based on the code in [124].

Our numerical results use different chemistry and material science problems. For VQE, we require their hamiltonians (generated via the open-source electronic structure

H_2O	Equilibrium	Stretched
O	(0.0, 0.0, 0.1173)	(0.0, 0.0, 0.0)
H	(0.0, 0.7572, -0.4692)	(0.0, 1.8186, 1.4081)
H	(0.0, -0.7572, -0.4692)	(0.0, -1.8186, 1.4081)

Table 6.1: Table describing the configurations of the atoms for the two water molecule problems used in this work.

System	# Parameters
H_4 chain	14
H_4 square	10
H_2O eq.	26
H_2O stret.	26
Hub. 1x6	15
Hub. 2x2	6
Hub. 2x3	16

Table 6.2: Number of parameters of the ansatze for each target problem.

package OpenFermion [126]) and the ansatz. The systems are presented in Table 6.2 with the respective number of parameters of the ansatz used.

The first kind of hamiltonians are Fermi-Hubbard Hamiltonians which describe the behavior of fermions (fundamental particles such as electrons) on a lattice of $n_x \times n_y$ sites. The variational circuit for this task is from Cade et al. [36].

The second kinds are molecular systems in different configurations (specified by $3D$ coordinates): H_4 and H_2O. For the H_4 in the chain configuration, the first hydrogen atom is located at 0.0 in all coordinates, then every atom is separated in the x-direction by 1.5Å. In the square configuration, we fix the hydrogen atoms in 2-dimensions. The positions of the atoms are parametrized by their polar coordinates with $R = 1.5$Å and $\theta = \frac{\pi}{4}$, and we locate them at $(x, y, 0), (x, -y, 0), (-x, y, 0), (-x, -y, 0)$ with $x = R\cos(\theta)$ and $y = R\sin(\theta)$. For the water molecule problems, the (x, y, z)-coordinates of the atoms are given in Table 6.1. Concerning the ansatz, we use the Unitary Coupled-Cluster ansatz [152, 56, 160], state-of-art for chemistry applications.

6.4 Hyperparameter tuning results

After presenting in the previous sections how a VQE is run and the problems considered, we start benchmarking under optimal hyperparameters found. In this work, we

use the iterated racing for automatic algorithm configuration or shortly IRACE [121], to tune the settings of SPSA and CMA-ES for the molecular systems. Additionally, we perform hyperparameter tuning of CMA-ES for the Hubbard model on three different configurations; 1×6, 2×2, and 2×3. For SPSA, however, we take the results of [36] where its hyperparameters were optimized.

For comparison, we use the relative energy error,

$$\Delta_r E = \left| \frac{\mathcal{C}(\boldsymbol{\theta}_{\text{opt}}) - E_0}{E_0 - c_0} \right|, \tag{6.2}$$

where $\mathcal{C}(\boldsymbol{\theta}_{\text{opt}})$ is the noiseless cost function evaluated at the optimized parameters $\boldsymbol{\theta}_{\text{opt}}$ obtained from a noisy optimization. E_0 is the lowest eigenvalue of the problem, computed exactly. c_0 is the coefficient of the identity operator, which is the largest term of the Hamiltonian and can be measured with exact precision. The results of these numerical simulations are shown in figure 6.1.

SPSA performs better in the weakly-correlated problems H_4 chain, H_2O equilibrium, and 2×2 Hubbard model. For the more interesting strongly-correlated systems (categorized as more challenging in chemistry), CMA-ES slightly outperforms with the mean values following mostly within error bars (orange ticks in fig. 6.1). Finally, we observe that CMA-ES starts to outperform SPSA when the system size and number of parameters increase. Such scenarios are more interesting for quantum simulations. We leave the study of larger systems for future work.

6.5 The sampling noise floor

In VQA, given the real objective $\mathcal{C}(\boldsymbol{\theta})$, one obtains optimal parameters returned dealing with a sampled version $\bar{\mathcal{C}}$ with variance $\text{Var}[\bar{\mathcal{C}}]$. By construction, it is possible that the evaluation of the outputted parameters is lower than its corresponding noiseless evaluation due to statistical fluctuations. Thus, by considering the parameters achieving the best-ever function evaluation, we can obtain worse results than the global minimum (assuming its existence).

Let us assume $\mathcal{C}(\boldsymbol{\theta})$ has a global minimum $\boldsymbol{\theta}_g$ with noiseless value $\mathcal{C}_g$. Then, under sampling noise, by evaluating the cost over multiple realizations of $\boldsymbol{\theta}$, including one at $\boldsymbol{\theta}_g$, we end up with a confidence interval for the evaluation at $\boldsymbol{\theta}_g$ with probability $1 - p$:

$$\Delta_p = [\mathcal{C}_g + m(p)\sqrt{\text{Var}[\bar{\mathcal{C}}]}, -\inf), \tag{6.3}$$

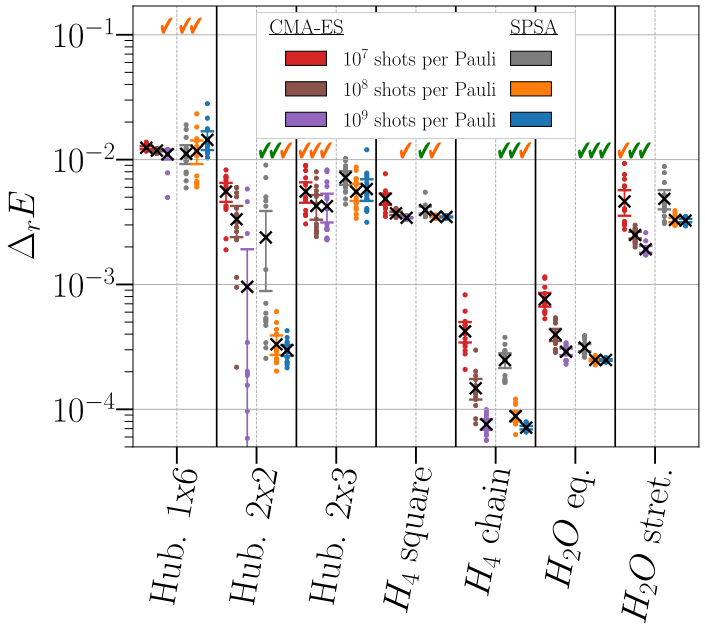


Figure 6.1: Comparison of optimized hyper-parameters of CMA-ES (red, brown, and purple dots) and SPSA (grey, orange, and blue dots). The black cross depicts the mean value, with the error bar showing the 95% confidence interval of 15 independent runs. The green ticks indicate that the optimization wins overall with a better mean and without overlapping in standard error. The orange ticks indicate the optimization wins in mean value, but its standard error overlaps with one or more optimization methods.

where $m(p) \sim \log(p)$ defines the size of the interval for the distribution of $\bar{\mathcal{C}}$. Assuming this distribution is symmetric, for $\boldsymbol{\omega} \neq \boldsymbol{\theta}_g$ satisfying $\mathcal{C}(\boldsymbol{\omega}) - m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]} \notin \Delta_p$, $\mathcal{C}(\boldsymbol{\omega})$ will lie outside Δ_p with probability strictly greater than $1-p$. Thus, with probability strictly greater than $1-p^2$, $\boldsymbol{\theta}_g$ can be correctly identified as optimal parameters. However, when the conditions are not met, the true minimum cannot be determined and alternative candidates can be drawn with probability p from the region:

$$\Omega(p) = \{\boldsymbol{\omega} \colon \mathcal{C}(\boldsymbol{\omega}) < \mathcal{C}(\boldsymbol{\theta}_g) + 2m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}\}, \tag{6.4}$$

True cost values an optimizer returning the best-measured candidate in this region lie in:

$$\mathbb{C}_p = \left[\mathcal{C}(\boldsymbol{\theta}_g), \mathcal{C}(\boldsymbol{\theta}_g) + 2m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}\right], \tag{6.5}$$

The quantity $2m(p)\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}$ is defined as the *sampling floor*. To be completely defined, the value of p should be set but it is not accessible as it depends upon the optimizer's convergence. Yet, one can demonstrate numerically the effect of the *sampling floor* on the candidates returned by optimizers.

CMA-ES returns two different candidates; the best-ever measured and a so-called favourite which uses all accumulated prior information during optimization. Such information contains many more shots than a single function call. In principle, the sampling noise can be averaged out and beat the sampling noise floor. We investigate such a phenomenon in this section. In figure 6.2, we present the results of the sampling noise floor on the optimization performance. For every system, we obtained the evaluations: $\bar{\mathcal{C}}(\boldsymbol{\theta}_{\mathrm{best}}), \mathcal{C}(\boldsymbol{\theta}_{\mathrm{best}})$ and $\mathcal{C}(\boldsymbol{\theta}_{\mathrm{fav}})$. The energy error is then computed as:

$$\Delta E = \frac{\mathcal{C}(\boldsymbol{\theta}_{\mathrm{opt}}) - E_0}{|E_0 - c_0|}. \tag{6.6}$$

Firstly, we observe that the best function evaluation (orange points) is often below the true energy due to sampling noise. The comparison should be done between $\mathcal{C}(\boldsymbol{\theta}_{\mathrm{best}})$ (red points) and $\mathcal{C}(\boldsymbol{\theta}_{\mathrm{fav}})$ (purple points). The average value of $\mathcal{C}(\boldsymbol{\theta}_{\mathrm{best}})$ estimates the sampling floor. In all cases, we observe the average evaluation of the favourite candidate is below the best candidate's. By using the favourite, CMA-ES overcomes the sampling floor. For the Hubbard model and the H_4 systems, this is not significant (up to a 95% confidence interval), but for the two geometries of the water molecule, the difference is much larger (up to a 3-fold reduction of error). Such a difference between problems may come from the different optimization landscapes

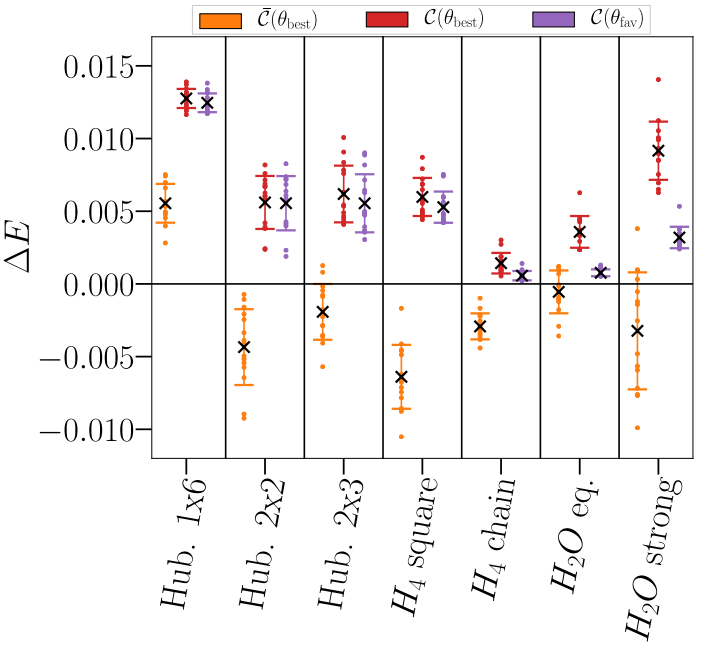


Figure 6.2: Comparison of the cost function evaluated at the best-ever measured and favourite candidate given by CMA-ES optimization under noisy optimization for the problems considered in this work. Each optimization uses 10^7 shots per Pauli over the course of the entire experiment: individual estimations of $\bar{\mathcal{C}}(\boldsymbol{\theta})$ are made using only 10^4 shots per Pauli. On the y-axis, we depict relative energy error where the column shows the mean and the overlaying points are the values of these energies of 15 independent runs. For each problem, from left to right we plot: (orange) the best-ever measured function evaluation during the CMA-ES optimization, (red) the best-ever candidate evaluated without noise, and (purple) the favourite candidate guessed by CMA-ES, evaluated without noise.

of the different problems. We leave this study for future work.

6.6 Conclusion

VQA algorithms require careful investigation of the performances of the underlying optimizers they depend on. Such optimizers may have been benchmarked using other types of problems. However quantum applications can be inherently different and optimizers will require to be adapted or discarded for tackling such settings. In this chapter, we benchmarked two state-of-art gradient-free optimizers on several chemistry and material science problems: SPSA and CMA-ES.

The optimizers were compared under a one-stage and three-stage sampling method from Ref. [36]. Hyperparameter optimization is performed showing comparable performances between SPSA and CMA-ES. The latter obtains better results on the more challenging systems. Additionally, we study the effect of sampling noise on the optimization performance using CMA-ES. Indeed, as the evaluation is not exact, using the best-ever evaluation out of the optimizer can be misleading. To overcome such a problem that we named *sampling noise floor*, using the favourite candidate yielded by CMA-ES was a better strategy. We leave as future work more benchmarking using larger systems of interest and against other VQA-tailored optimizers.

Chapter 7

Hyperparameter Importance of Quantum Neural Networks Across Small Datasets

Machine learning has not been spared in the quest for meaningful quantum computing applications. One of the more investigated approaches is the use of a special type of quantum circuit – a so-called quantum neural network – to serve as a basis for a machine learning model. We introduced the latter workflow in Chapter 2.5. Roughly speaking, as the name suggests, a quantum neural network can play a similar role to a neural network. However, specifically for applications in machine learning contexts, very little is known about suitable circuit architectures, or model hyperparameters one should use to achieve good learning performance. In this chapter[1], we apply the functional ANOVA framework to quantum neural networks to analyze which of the hyperparameters were most influential for their predictive performance. We analyze one of the most typically used quantum neural network architectures.

[1]Contents of this chapter are based on [137]; Charles Moussa, Jan N. van Rijn, Thomas Bäck, and Vedran Dunjko. Hyperparameter importance of quantum neural networks across small datasets. In Poncelet Pascal and Dino Ienco, editors, *Discovery Science*, pages 32–46, Cham, 2022. Springer Nature Switzerland.

7.1 Introduction

Quantum models can exhibit clear potential in special datasets where we have theoretically provable separations with classical models [94, 117, 163, 188]. More theoretical works also study these models from a generalization perspective [38]. Quantum circuits with adjustable parameters, also called quantum neural networks, have been used to tackle regression [130], classification [83], generative adversarial learning [208], and reinforcement learning tasks [94, 180].

However, the value of quantum machine learning on real-world datasets is still to be investigated in any larger-scale systematic fashion [82, 154]. Currently, common practices from machine learning, such as large-scale benchmarking, hyperparameter importance, and analysis have been challenging tools to use in the quantum community [171]. Given that there exist many ways to design quantum circuits for machine learning tasks, this gives rise to a hyperparameter optimization problem. However, there is currently limited intuition as to which hyperparameters are important to optimize and which are not. Such insights can lead to much more efficient hyperparameter optimization [31, 60, 133].

In order to fill this gap, we employ functional ANOVA [92, 181], a tool for assessing hyperparameter importance. This follows the methodology of [192, 175], who employed this across datasets, allowing for more general results. For this, we selected a subset of several low-dimensional datasets from the OpenML-CC18 benchmark [22], that are matching the current scale of simulations of quantum hardware. We defined a configuration space consisting of ten hyperparameters from an aggregation of quantum computing literature and software. We extend this methodology by an important additional verification step, where we verify the performance of the internal surrogate models. Finally, we perform an extensive experiment to verify whether our conclusions hold in practice. While our main findings are in line with previous intuition on a few hyperparameters and the verification experiments, we also discovered new insights. For instance, setting well the learning rate is deemed the most critical hyperparameter in terms of marginal contribution on all datasets, whereas the particular choice of entangling gates used is considered the least important except on one dataset.

7.2 Background

In this section, we introduce the necessary background on functional ANOVA, quantum computing, and quantum circuits with adjustable parameters for supervised learn-

ing.

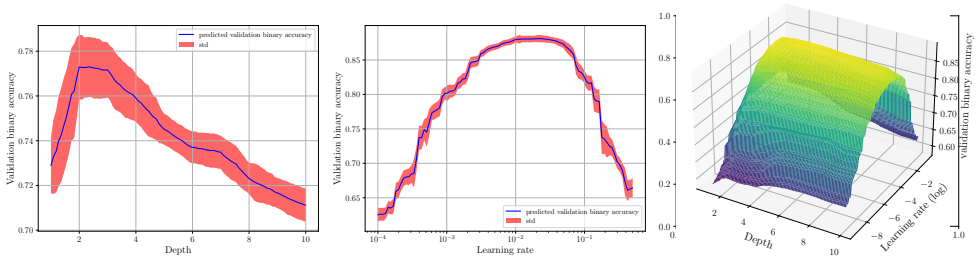
7.2.1 Functional ANOVA

When applying a new machine learning algorithm, it is unknown which hyperparameters to modify in order to get high performances on a task. Several techniques exist that assess hyperparameter importance, such as functional ANOVA [164]. The latter framework can detect the importance of both individual hyperparameters and interaction effects between different subsets of hyperparameters. We first introduce the relevant notation, based on the work by Hutter *et al.* [92].

Let A be a machine learning algorithm that has n hyperparameters with domains $\Theta_1, \ldots, \Theta_n$ and *configuration space* $\boldsymbol{\Theta} = \Theta_1 \times \ldots \times \Theta_n$. An instantiation of A is a vector $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_n\}$ with $\theta_i \in \Theta_i$ (this is also called a *configuration* of A). A partial instantiation of A is a vector $\boldsymbol{\theta}_U = \{\theta_{i_1}, \ldots, \theta_{i_k}\}$ with a subset $U = \{i_1, \ldots, i_k\} \subseteq N = [n] = \{1, \ldots, n\}$ of the hyperparameters fixed, and the values for other hyperparameters unspecified. Note that $\boldsymbol{\theta}_N = \boldsymbol{\theta}$.

Functional ANOVA is based on the concept of a marginal of a hyperparameter, i.e., how a given value for a hyperparameter performs, averaged over all possible combinations of the other hyperparameters' values. The *marginal performance* $\hat{a}_U(\boldsymbol{\theta}_U)$ is described as the average performance of all complete instantiations $\boldsymbol{\theta}$ that have the same values for hyperparameters that are in $\boldsymbol{\theta}_U$. As an illustration, Fig. 7.1 shows marginals for two hyperparameters of a quantum neural network and their union. As the number of terms to consider for the marginal can be very large, the authors of [92] used tree-based surrogate regression models to calculate efficiently the average performance. Such a model yields predictions $\hat{y}$ for the performance p of arbitrary hyperparameter settings.

Functional ANOVA determines how much each hyperparameter (and each combination of hyperparameters) contributes to the variance of $\hat{y}$ across the algorithm's hyperparameter space $\boldsymbol{\Theta}$, denoted $\mathbb{V}$. Intuitively, if the marginal has a high variance, the hyperparameter is highly important to the performance measure. Such framework has been used for studying the importance of hyperparameters of common machine learning models such as support vector machines, random forests, Adaboost, and residual neural networks [192, 175]. We refer to [92] for a complete description and introduce the quantum supervised models considered in this study along with the basics of quantum computing.



(a) **(b)** **(c)**

Figure 7.1: Examples of marginals for a quantum neural network with validation accuracy as performance on the banknote-authentication dataset. The hyperparameters correspond to the learning rate used during training (a), the number of layers, also known as depth (b), and their combination (c). The hyperparameter values for the learning rate are on a log scale. When considered individually, we see for instance that depth and learning rate should not be set too high for better performances. However, when grouped together, the learning rate seems most influential.

7.2.2 Supervised learning with Parameterized Quantum Circuits

A parameterized quantum circuit (also called *ansatz*) can be represented by a quantum circuit with adjustable real-valued parameters $\boldsymbol{\theta}$. The latter is then defined by a unitary $U(\boldsymbol{\theta})$ that acts on a fixed n-qubit state (e.g., $|0^{\otimes n}\rangle$). The ansatz may be constructed problem-independent generic construction. The latter are often designated as *hardware-efficient*.

For a machine learning task, this unitary encodes an input data instance $x \in \mathbb{R}^d$ and is parameterized by a trainable vector $\boldsymbol{\theta}$. Many designs exist but hardware-efficient parameterized quantum circuits [97] with an alternating-layered architecture are often considered in quantum machine learning when no information on the structure of the data is provided. This architecture is depicted in an example presented in Fig. 7.2 and essentially consists of an alternation of encoding unitaries U_{enc} and variational unitaries U_{var}. In the example, U_{enc} is composed of single-qubit rotations R_X, and U_{var} of single-qubit rotations R_z, R_y and entangling Ctrl-Z gates, represented as ┇ in Fig. 7.2, forming the entangling part of the circuit. Such entangling part denoted U_{ent}, can be defined by connectivity between qubits.

These parameterized quantum circuits are similar to neural networks where the circuit architecture is fixed and the gate parameters are adjusted by a classical optimizer such as gradient descent. They have also been named quantum neural networks.

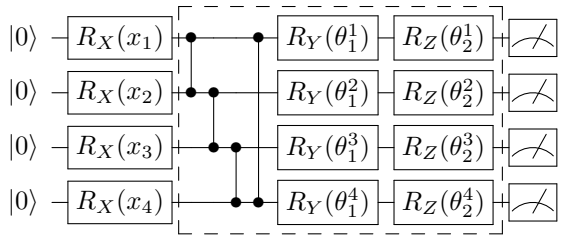


Figure 7.2: Parameterized quantum circuit architecture example with 4 qubits and ring connectivity (qubit 1 is connected to 2, 2 to 3, 3 to 4, and 4 to 1 makes a ring). The first layer of R_X is the encoding layer U_{enc}, taking a data instance $x \in \mathbb{R}^4$ as input. It is followed by the entangling part with Ctrl-Z gates. Finally, a variational layer U_{var} is applied. Eventually, we do measurements to be converted into predictions for a supervised task. The dashed part can be repeated many times to increase the expressive power of the model.

The parameterized layer can be repeated multiple times, which increases its *expressive power* like neural networks [179]. The data encoding strategy (such as reusing the encoding layer multiple times in the circuit - a strategy called *data reuploading*) also influences the latter [151, 174].

Finally, the user can define the observable(s) and the post-processing method to convert the circuit outputs into a prediction in the case of supervised learning. Commonly, observables based on the single-qubit Z operator are used. When applied on $m \leq n$ qubits, the observable is represented by a $2^m - 1$ square diagonal matrix with $\{-1, 1\}$ values and is denoted $\mathcal{O} = Z \otimes Z \otimes \cdots \otimes Z$.

Having introduced parameterized quantum circuits, we present the hyperparameters of the models, the configuration space, and the experimental setup for our functional ANOVA-based hyperparameter importance study.

7.3 Methods

In this section, we describe the network type and its hyperparameters and define the methodology that we follow.

7.3.1 Hyperparameters and configuration space

Many designs have been proposed for parameterized quantum circuits depending on the problem at hand or motivated research questions and contributions.

Such propositions can be aggregated and translated into a set of hyperparameters and configuration space for the importance study. As such, we first did an extensive literature review on parameterized quantum circuits for machine learning [18, 83, 84, 93, 94, 116, 123, 129, 130, 154, 168, 180, 194, 195, 202, 208] as well as quantum machine learning software [6, 19, 34]. This resulted in a list of 10 hyperparameters, presented in Table 7.1. We choose them so we balance between having well-known hyperparameters that are expected to be important, and less considered ones in the literature. For instance, many works use Adam [103] as the underlying optimizer, and the learning rate should generally be well chosen. On the contrary, the entangling gate used in the parameterized quantum circuit is generally a fixed choice.

From the literature, we expect data encoding strategy/circuit to be important. We choose two main forms for U_{enc}. The first one is the hardware-efficient $\bigotimes_{i=1}^{n} R_X(x_i)$. The second takes the following form from [19, 93, 83]:

$$U_{\text{enc}}(\boldsymbol{x}) = U_z(\boldsymbol{x}) H^{\otimes n} \tag{7.1}$$

$$U_z(\boldsymbol{x}) = \exp\left(-\mathrm{i}\pi \left[\sum_{i=1}^{n} x_i Z_i + \sum_{\substack{j=1, \\ j>i}}^{n} x_i x_j Z_i Z_j \right] \right). \tag{7.2}$$

Using data-reuploading [151] results in a more expressive model [174], and this was also demonstrated numerically [94, 151, 180]. Finally, pre-processing of the input is also sometimes used in encoding strategies that directly feed input features into Pauli rotations. It also influences the expressive power of the model [174]. In this work, we choose a usual activation function *tanh* commonly used in neural networks. We do so as its range is $[-1, 1]$, which is the same as the data features during training after the normalization step.

The list of hyperparameters we take into account is non-exhaustive. It can be extended at will, at the cost of more software engineering and budget for running experiments.

7.3.2 Assessing Hyperparameter Importance

Once the list of hyperparameters and configuration space are decided, we perform the hyperparameter importance analysis with the functional ANOVA framework. Assessing the importance of the hyperparameters boils down to four steps. Firstly, the models are applied to various datasets by sampling various configurations in a hyperparameter

optimization process. The performances or metrics of the models are recorded along. The sampled configurations and performances serve as data for functional ANOVA. As functional ANOVA uses internally tree-based surrogate models, namely random forests [32], we decided to add an extra step with reference to [192]. In the second step, we verify the performance of the internal surrogate models. We cross-evaluate them using regression metrics commonly used in surrogate benchmarks [53]. Surrogates performing badly at this step are then discarded from the importance analysis, as they can deteriorate the quality of the study. Thirdly, the marginal contribution of each hyperparameter over all datasets can be then obtained and used to infer a ranking of their importance. Finally, a verification step similar to [192] is carried out to confirm the inferred ranking previously obtained. We explain such a procedure in the following section.

7.3.3 Verifying Hyperparameter Importance

When applying the functional ANOVA framework, an extra verification step is added to confirm the output from a more intuitive notion of hyperparameter importance [192]. It is based on the assumption that hyperparameters that perform badly when fixed to a certain value (while other hyperparameters are optimized), will be important to optimize. The authors of [192] proposed to carry out a costly random search procedure fixing one hyperparameter at a time. In order to avoid a bias to the chosen value to which this hyperparameter is fixed, several values are chosen, and the optimization procedure is carried out multiple times. Formally, for each hyperparameter θ_j we measure $y^*_{j,f}$ as the result of a random search for maximizing the metric, fixing θ_j to a given value $f \in F_j, F_j \subseteq \Theta_j$. For categorical θ_j with domain Θ_j, $F_j = \Theta_j$ is used. For numeric θ_j, the authors of [192] use a set of 10 values spread uniformly over

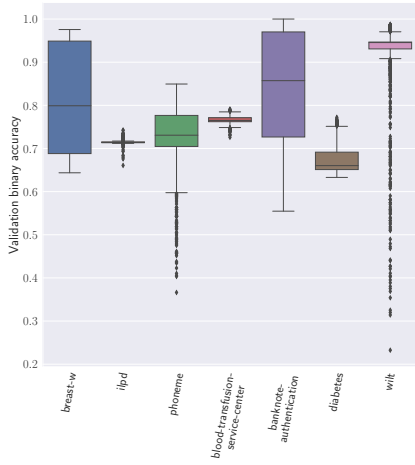


Figure 7.3: Performances of 1 000 quantum machine learning models defined by different configurations of hyperparameters over each dataset. The metric of interest in the study is the 10-fold cross-validation accuracy. We take the best-achieved metric per model trained over 100 epochs.

θ_j's range. We then compute $y_j^* = \frac{1}{|F_j|}\sum_{f\in F_j} y_{j,f}^*$, representing the score when not optimizing hyperparameter θ_j, averaged over fixing θ_j to various values it can take. Hyperparameters with lower values for y_j^* are assumed to be more important since the performance should deteriorate more when set sub-optimally.

In our study, we extend this framework to be used on the scale of quantum machine learning models. As quantum simulations can be very expensive, we carry out the verification experiment by using the predictions of the surrogate instead of fitting new quantum models during the verification experiment. The surrogates yield predictions $\hat{y}$ for the performance of arbitrary hyperparameter settings sampled during a random search. Hence, they serve to compute $y_{j,f}^*$. This is also why we assessed the quality of the built-in surrogates as the second step. Poorly-performing surrogates can deteriorate the quality of the constructed marginals, and therefore lead to poorly-supported conclusions.

7.4 Dataset and inclusion criteria

To apply our quantum models and study the importance of the previously introduced hyperparameters, we consider classical datasets. Similarly to [192], we use datasets from the OpenML-CC18 benchmark suite [22]. In our study, we consider only the case where the number of qubits available is equal to the number of features, a common setting in the quantum community. As simulating quantum circuits is a costly task, we limit this study to the case where the number of features is less than 20 after preprocessing.[2] Our first step was to identify which datasets fit this criterion. We include all datasets from the OpenML-CC18 that have 20 or fewer features after categorical hyperparameters have been one-hot-encoded, and constant features are removed. Afterwards, the input variables are also scaled to unit variance as a normalization step. The scaling constants are calculated on the training data and applied to the test data.

The final list of datasets is given in Table 7.2. In total, 7 datasets fitted the criterion considered in this study. For all of them, we picked the OpenML Task ID giving the 10-fold cross-validation task. A quantum model is then applied using the latter procedure, with the aforementioned preprocessing steps.

[2] A 10-fold cross-validation run in our experiment takes on average 262 minutes for 100 epochs with Tensorflow Quantum [34].

7.5 Results

In this section, we present the results obtained using the hyperparameters and the methodology defined in Section 7.3 with the datasets described in Section 7.4. First, we show the distribution of performances obtained during a random search where configurations are independently sampled for each dataset. Then we carry out the surrogate verification. Finally, we present the functional ANOVA results in terms of hyperparameter importance with marginal contributions and the random search verification per hyperparameter.

7.5.1 Performance distributions per dataset

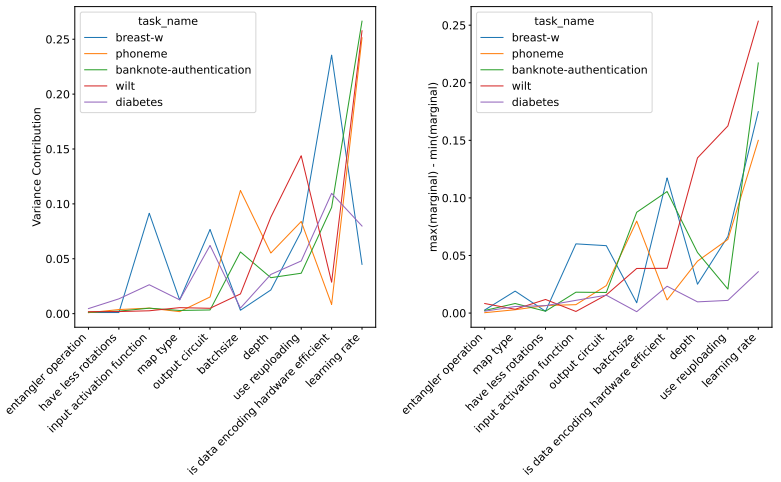
For each dataset, we sampled independently 1 000 hyperparameter configurations and run the quantum models for 100 epochs as budget. As a performance measure, we recorded the best validation accuracy obtained over 100 epochs. Fig. 7.3 shows the distribution of the 10-fold cross-validation accuracy obtained per dataset. We observe the impact of hyperparameter optimization by the difference between the least performing and the best model configuration. For instance, on the wilt dataset, the best model gets an accuracy close to 1, and the least below 0.25. We can also see that some datasets present a smaller spread of performances. ilpd and blood-transfusion-service-center are in this case. It seems that hyperparameter optimization does not have a real effect, because most hyperparameter configurations give the same result. As such, the surrogates could not differentiate between various configurations. In general, hyperparameter optimization is important for getting high performances per dataset and detecting datasets where the importance study can be applied.

7.5.2 Surrogate verification

Functional ANOVA relies on an internal surrogate model to determine the marginal contribution per hyperparameter. If this surrogate model is not accurate, this can have a severe limitation on the conclusions drawn from functional ANOVA. In this experiment, we verify whether the hyperparameters can explain the performances of the models. Table 7.3 shows the performance of the internal surrogate models. We notice low regression scores for the two datasets (less than 0.75 R2 scores). Hence we remove them from the analysis.

7.5.3 Marginal contributions

For functional ANOVA, we used 128 trees for the surrogate model. Fig. 7.4(a,b) shows the marginal contribution of each hyperparameter over the remaining 5 datasets. We distinguish 3 main levels of importance. According to these results, the learning rate, depth, and the data encoding circuit and reuploading strategy are critical. These results are in line with our expectations. The entangler gate, connectivity, and whether we use R_X gates in the variational layer are the least important according to functional ANOVA. Hence, our results reveal new insights into these hyperparameters that are not considered in general.



(a) **(b)**

Figure 7.4: The marginal contributions per dataset are presented as a) the variance contribution and b) the difference between the minimal and maximal value of the marginal of each hyperparameter. The hyperparameters are sorted from the least to most important using the median. We distinguish from the plot 3 main levels of importance.

7.5.4 Random search verification

In line with the work of [192], we perform an additional verification experiment that verifies whether the outcomes of functional ANOVA are in line with our expectations. However, the verification procedure involves an expensive, post-hoc analysis: a random search procedure fixing one hyperparameter at a time. As our quantum simulations are costly, we used the surrogate models fitted on the current dataset considered over the

1 000 configurations obtained initially to predict the performances one would obtain when presented with a new configuration.

Fig. 7.5 shows the average rank of each run of random search, labeled with the hyperparameter whose value was fixed to a default value. A high rank implies poor performance compared to the other configurations, meaning that tuning this hyperparameter would have been important. We witness again the 3 levels of importance, with almost the same order obtained. However, the input_activation_function is deemed more important while the batch size is less.

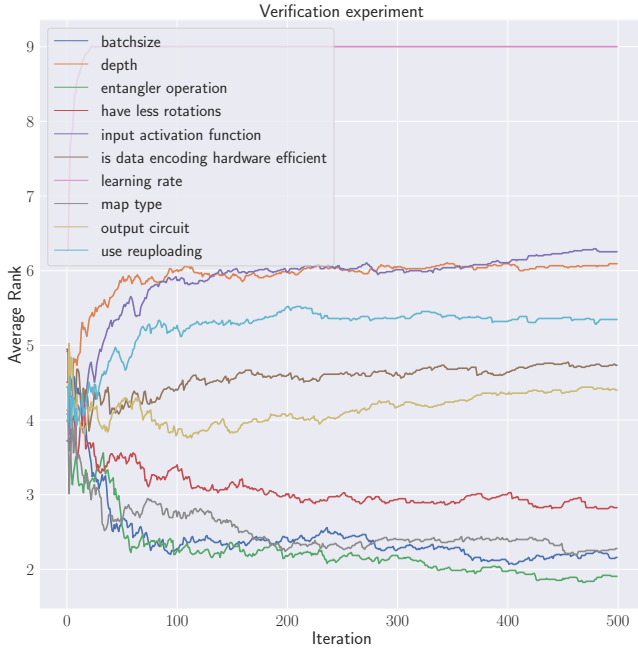


Figure 7.5: Verification experiment of the importance of the hyperparameters. A random search procedure up to 500 iterations excluding one parameter at a time is used. A lower curve means the hyperparameter is deemed less important.

More simulations with more datasets may be required to validate the importance. However, we retrieve empirically the importance of well-known hyperparameters while considering less important ones. Hence functional ANOVA becomes an interesting tool for quantum machine learning in practice.

7.6 Conclusion

In this chapter, we study the importance of hyperparameters related to quantum neural networks for classification using the functional ANOVA framework. Our experiments are carried out over OpenML datasets that match the current scale of quantum hardware simulations (i.e., datasets that have at most 20 features after pre-processing operators have been applied, hence using 20 qubits). We selected and presented the hyperparameters from an aggregation of quantum computing literature and software. Firstly, hyperparameter optimization highlighted datasets where we observed high differences between configurations. This underlines the importance of hyperparameter optimization for these datasets. There were also datasets that showed little difference. These led us to extend the methodology by adding an additional verification step of the internal surrogate performances. From our results, we distinguished 3 main levels of importance. On the one hand, Adam's learning rate, depth, and data encoding strategy are deemed very important, as we expected. On the other hand, the less considered hyperparameters such as the particular choice of the entangling gate and using 3 rotation types in the variational layer are in the least important group. Hence, our experiment both confirmed expected patterns and revealed new insights for quantum model selection.

For future work, further methods from the field of automated machine learning can be applied to quantum neural networks [31, 60, 133]. Indeed, our experiments have shown the importance of hyperparameter optimization, and this should become part of the protocols applied within the community. We further envision functional ANOVA to be employed in future works related to quantum machine learning and understanding how to apply quantum models in practice. For instance, it would be interesting to consider quantum data, for which quantum machine learning models may have an advantage. Plus, extending hyperparameter importance to techniques for scaling to a large number of features with the number of qubits, such as dimensionality reduction or divide-and-conquer techniques, can be left for future work. Finally, this type of study can also be extended to different noisy hardware and to algorithm/model selection and design. If we have access to a cluster of different quantum computers, then choosing which hardware works best for machine learning tasks becomes possible. One could also extend our work with meta-learning [31], where a model configuration is selected based on meta-features created from dataset features. Such types of studies already exist for parameterized quantum circuits applied to combinatorial optimization and we presented them in chapters 3 and 4 [135, 138].

Table 7.1: List of hyperparameters considered for hyperparameter importance for quantum neural network, as we named them in our Tensorflow-Quantum code.

Hyperparameter	Values	Description
Adam learning rate	$[10^{-4}, 0.5]$ (log)	The learning rate with which the quantum neural network starts training. The range was taken from the automated machine learning library Auto-sklearn [60]. We uniformly sample taking the logarithmic scale.
batch size	16, 32, 64	Number of samples in one batch of Adam used during training
depth	$\{1, 2, \cdots, 10\}$	Number of variational layers defining the circuit
is data_encoding hardware efficient	True, False	Whether we use the hardware-efficient circuit $\bigotimes_{i=1}^{n} R_X(x_i)$ or an IQP circuit defined in Eq.7.1 to encode the input data.
use reuploading	True, False	Whether the data encoding layer is used before each variational layer or not.
have less rotations	True, False	If True, only use layers of R_Y, R_Z gates as the variational layer. If False, add a layer of R_X gates.
entangler operation	cz, sqiswap	Which entangling gate to use in U_{ent}
map type	ring, full, pairs	The connectivity used for U_{ent}. The ring connectivity use an entangling gate between consecutive indices $(i, i+1), i \in \{1, \ldots, n\}$ of qubits. The full one uses a gate between each pair of indices $(i, j), i < j$. Pairs connect even consecutive indices first, then odd consecutive ones.
input activation function	linear, tanh	Whether to input $tanh(x_i)$ as rotations or just x_i.
output circuit	2Z, mZ	The observable(s) used as output(s) of the circuit. If 2Z, we use all possible pairs of qubit indices defining $Z \otimes Z$. If mZ, the tensor product acts on all qubits. Note we do not use single-qubit Z observables although they are quite often used in the literature. Indeed, they are provably not using the entire circuit when it is shallow. Hence we decided to use $Z \otimes Z$ instead. Also, a single neuron layer with a sigmoid activation function is used as a final decision layer similar to [168].

Table 7.2: List of datasets used in this study. The number of features is obtained after a usual preprocessing used in machine learning methods, such as one-hot encoding.

Dataset	OpenML Task ID	Number of features	Number of instances
breast-w	15	9	699
diabetes	37	8	768
phoneme	9952	5	5 404
ilpd	9971	11	583
banknote-authentication	10093	4	1 372
blood-transfusion-service-center	10 101	4	748
wilt	146820	5	4 839

Table 7.3: Performances of the surrogate models built within functional ANOVA over a 10-fold cross-validation procedure. We present the average coefficient of determination (R2), root mean squared error (RMSE), and Spearman's rank correlation coefficient (CC). These are common regression metrics for benchmarking surrogate models on hyperparameters [53]. The surrogates over ilpd and blood-transfusion-service-center obtain low scores (less than .75 R2), hence we remove them from the study.

Dataset	R2 score	RMSE	CC
breast-w	0.8663	0.0436	0.9299
diabetes	0.7839	0.0155	0.8456
phoneme	0.8649	0.0285	0.9282
ilpd	0.1939	0.0040	0.4530
banknote-authentication	0.8579	0.0507	0.9399
blood-transfusion-service-center	0.6104	0.0056	0.8088
wilt	0.7912	0.0515	0.8015

Chapter 8

Resource frugal optimizer for quantum machine learning

Variational quantum machine learning algorithms have the potential to solve practical problems on real hardware, particularly when involving quantum data. In Chapter 7, we assessed the importance of hyperparameters of quantum neural networks on classical data and in idealized simulation settings. However, training these algorithms can be challenging and calls for tailored optimization procedures. Specifically, QML applications can require a large shot-count overhead due to the large datasets involved. In this chapter[1], we advocate for simultaneous random sampling over both the dataset as well as the measurement operators that define the loss function. We consider a highly general loss function that encompasses many QML applications, and we show how to construct an unbiased estimator of its gradient. This allows us to propose a shot-frugal gradient descent optimizer called Refoqus (REsource Frugal Optimizer for QUantum Stochastic gradient descent). Our numerics indicate that Refoqus can save several orders of magnitude in shot cost, even relative to optimizers that sample over measurement operators alone.

[1] Contents of this chapter are based on [136]; Charles Moussa, Max Hunter Gordon, Michal Baczyk, M. Cerezo, Lukasz Cincio, and Patrick J. Coles. Resource frugal optimizer for quantum machine learning. *arXiv:2211.04965*, 2022.

8.1 Introduction

The field of quantum machine learning (QML) revolves mostly around variational methods, which involve classically training a parameterized quantum model. Variational QML, henceforth referred to as QML for simplicity, is indeed a leading candidate for implementing QML in the near term. However, it has faced various sorts of trainability issues.

Exponentially vanishing gradients, known as barren plateaus [125, 42, 87, 88, 176, 122, 190, 9, 153], as well as the prevalence of local minima [23, 7] are two issues that can impact the complexity of the training process. Quantum hardware noise also impacts trainability [197, 184]. All of these issues contribute to increasing the number of shots and iterations required to minimize the QML loss function. Indeed, a detailed shot-cost analysis has painted a concerning picture [200].

It is therefore clear that QML requires careful frugality in terms of the resources expended during the optimization process. Indeed, novel optimizers have been developed in response to these challenges. Quantum-aware optimizers aim to replace off-the-shelf classical optimizers with ones that are specifically tailored to the quantum setting [185, 107, 140]. Shot-frugal optimizers [109, 72, 189] have been proposed in the context of variational quantum eigensolver (VQE), whereby one can sample over terms in the Hamiltonian instead of measuring every term [10]. While significant progress has been made on such optimizers, particularly for VQE, we argue that very little work has been done to specifically tailor optimizers to the QML setting. The cost functions in QML go well beyond those used in VQE and hence QML requires more general tools.

In this work, we generalize previous shot-frugal and iteration-frugal optimizers, such as those in Refs. [109, 72, 10], by extending them to the QML setting. Specifically, we allow for random, weighted sampling over both the input and the output of the loss function estimation circuit. In other words, and as shown in Fig. 8.1, we allow for sampling over the dataset as well as over the measurement operators used to define the loss function. Our sampling approach allows us to unlock the frugality (i.e., to achieve the full potential) of adaptive stochastic gradient descent optimizers, such as iCANS [109] and gCANS [72].

We discuss how our approach applies to various QML applications such as perceptron-based quantum neural networks [17, 176], quantum autoencoders [161], variational quantum principal component analysis (PCA) [110, 41], and classifiers that employ the mean-squared-error loss function [170, 189]. Each of these applica-

tions can be unified under one umbrella by considering a generic loss function with a highly general form. Thus we state our main results for this generic loss function. We establish an unbiased estimator for this loss function and its gradient. In turn, this allows us to provide convergence guarantees for certain optimization routines, like stochastic gradient descent. Furthermore, we show that for this general loss function one can use the form of the estimator to inform a strategy that distributes shots to obtain the best shot frugal estimates.

Finally, we numerically investigate the performance of our new optimization approach, which we call Refoqus (REsource Frugal Optimizer for QUantum Stochastic gradient descent). For a quantum PCA task, Refoqus significantly outperforms state-of-the-art optimizers in terms of the shot resources required. Refoqus even outperforms Rosalin [10] - a shot-frugal optimizer that samples only over measurement operators. Hence, Refoqus will be a crucial tool to minimize the number of shots and iterations required in near-term QML implementations.

8.2 Background

8.2.1 Stochastic Gradient Descent

One of the most popular optimization approaches is gradient descent, which involves the following update rule for the parameter vector:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \boldsymbol{\nabla} \mathcal{L}(\boldsymbol{\theta}^{(t)}) \,. \tag{8.1}$$

Here, $\mathcal{L}$ is the loss function, α is the learning rate, and $\boldsymbol{\theta}^{(t)}$ is the parameter vector at iteration t.

Oftentimes one only has access to noisy estimates of the gradient $\boldsymbol{\nabla}\mathcal{L}$, in which case the optimizer is called stochastic gradient descent. In the quantum setting, this situation arises due to shot noise or noise due to sampling from terms in some expansion of the gradient.

8.2.2 Parameter Shift Rule

Estimating the gradient is clearly an essential step in stochastic gradient descent. For this purpose, one useful tool that is often employed in the quantum case is the so-called parameter shift rule [131, 169]. We emphasize that several assumptions go into this rule. Specifically, suppose we assume that the quantum circuit ansatz $U(\boldsymbol{\theta})$ can

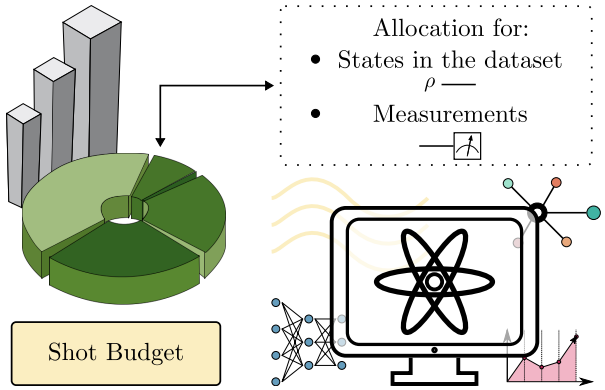


Figure 8.1: **Schematic illustration of Refoqus.** Measurements, or shots, are a precious (and expensive) resource in quantum computing. As such, they should be used sparingly and only when absolutely necessary. This is particularly important in variational QML methods where training a model requires continuously calling a quantum device to estimate the loss function or its gradients. Refoqus provides a shot-frugal optimization paradigm where shots are allocated by sampling over the input data and the measurement operators.

be expressed as $U(\boldsymbol{\theta}) = \prod_x e^{-i\theta_x \sigma_x} W_x$ where W_x are unparametrized unitaries, and where σ_x are Pauli operators. Moreover, we consider the case when the loss function has the simple form $\mathcal{L}(\boldsymbol{\theta}) = \langle 0|U^\dagger(\boldsymbol{\theta}) H U(\boldsymbol{\theta})|0\rangle$ for some Hermitian operator H. (Note that we will consider more complicated loss functions in this work, and hence we are just stating this special case for background information.) In this case, the parameter shift rule gives

$$\partial_x \mathcal{L}(\boldsymbol{\theta}) := \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x} = \frac{\mathcal{L}(\boldsymbol{\theta} + \frac{\pi}{2}\boldsymbol{\delta}_x) - \mathcal{L}(\boldsymbol{\theta} - \frac{\pi}{2}\boldsymbol{\delta}_x)}{2}\,, \tag{8.2}$$

where $\boldsymbol{\delta}_x$ is a unit vector with a one on the x-th component. Equation (8.2) allows one to estimate the gradient by estimating the loss function as specific points on the landscape. Hence it simplifies the procedure to estimate the gradient.

8.2.3 Lipschitz continuity

The loss function $\mathcal{L}$ is called Lipschitz continuous if there is some Lipschitz constant $L \geq 0$ that satisfies the bound

$$\|\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}_a) - \boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}_b)\| \leq L\|\boldsymbol{\theta}_a - \boldsymbol{\theta}_b\|\,, \tag{8.3}$$

for all $\boldsymbol{\theta}_a, \boldsymbol{\theta}_b \in \mathrm{dom}(\mathcal{L})$, where $\|\cdot\|$ is the ℓ_2 norm. This property provides a recipe for choosing an appropriate learning rate, α. Specifically, if (8.3) holds and we have access to the exact gradient, then choosing $\alpha \leq 2/L$ is sufficient to guarantee convergence using the update rule in (8.1).

8.2.4 iCANS

Inspired by an adaptive batch size optimizer used in classical machine learning [13], the iCANS (individual coupled adaptive number of shots) optimizer [109] was introduced as an adaptive method for stochastic gradient in the context of the variational quantum eigensolver. It allows the number of shots per partial derivative (i.e., gradient component) to vary individually, hence the name iCANS.

Consider the gain (i.e., the decrease in the loss function), denoted $\mathcal{G}_x$, associated with updating the x-th parameter θ_x. The goal of iCANS is to maximize the expected gain per shot. That is, for each individual partial derivative, we maximize the shot efficiency:

$$\gamma_x := \frac{\mathbb{E}[\mathcal{G}_x]}{s_x}; \quad x = 1, \ldots, d\,, \tag{8.4}$$

where s_x is the shot allocation for gradient component x and d is the number of gradient components. Solving for the optimal shot allocation gives:

$$s_x = \frac{2L\alpha}{2 - L\alpha} \frac{\sigma_x^2}{g_x^2}. \tag{8.5}$$

Here, g_x is an unbiased estimator for the x-th gradient component, and σ_x is the standard deviation of a random variable X_x whose sample mean is g_x. While iCANS often heuristically outperforms other methods, it can have instabilities.

8.2.5 gCANS

Recently, a potential improvement over iCANS was introduced called gCANS (global coupled adaptive number of shots) [72]. gCANS considers the expected gain $\mathbb{E}[\mathcal{G}]$ over the entire gradient vector. Then the goal is to maximize the shot efficiency

$$\gamma := \frac{\mathbb{E}[\mathcal{G}]}{\sum_{x=1}^{d} s_x}\,, \tag{8.6}$$

where the sum $\sum_{x=1}^{d} s_x$ goes over all components of the gradient. Solving for the optimal shot count then gives:

$$s_x = \frac{2L\alpha}{2 - L\alpha} \frac{\sigma_x \sum_{x'=1}^{d} \sigma_{x'}}{\|\nabla \mathcal{L}(\boldsymbol{\theta})\|^2}. \tag{8.7}$$

(Note that an exponential moving average is used to estimate σ_x and $\|\nabla \mathcal{L}(\boldsymbol{\theta})\|^2$ as their true value is not accessible.) It was proven that gCANS achieves geometric convergence to the optimum, often reducing the number of shots spent for comparable solutions against its predecessor iCANS.

8.2.6 Rosalin

Shot-frugal optimizers like iCANS and gCANS rely on having an unbiased estimator for the gradient or its components. However, this typically places a hard floor on how many shots must be allocated at each iteration, i.e., going below this floor could result in a biased estimator. This is because the measurement operator H is typically composed of multiple non-commuting terms, each of which must be measured individually. Each of these terms must receive some shot allocation to avoid having a biased estimator. However, having this hard floor on the shot requirement is antithetical to the shot-frugal nature of iCANS and gCANS, and ultimately it handicaps these optimizers' ability to achieve shot frugality.

This issue inspired a recent proposal called Rosalin (Random Operator Sampling for Adaptive Learning with Individual Number of shots) [10]. Rosalin employs weighted random sampling of operators in the measurement operator $H = \sum_j c_j H_j$, which allows one to achieve an unbiased estimator without a hard floor on the shot requirement. (Even a single shot, provided that it is randomly allocated according to an appropriate probability distribution, can lead to an unbiased estimator.) When combined with the shot allocation methods from iCANS or gCANS, the operator sampling methods in Rosalin were shown to be extremely powerful in the context of molecular chemistry problems, which often have a large number of terms in H.

We remark that Ref. [10] considered several sampling strategies. Given a budget of s_{tot} shots and N terms, a simple strategy is to distribute shots per term equally ($s_j = s_{\text{tot}}/N$) - referred as uniform deterministic sampling (UDS). Defining $M = \sum_j |c_j|$, one can also use weighted deterministic sampling (WDS) where the shots are proportionally distributed: $s_j = s_{\text{tot}} * \frac{|c_j|}{M}$. One can add randomness by using $p_j = \frac{|c_j|}{M}$ to define a (non-uniform) probability distribution to select which term should

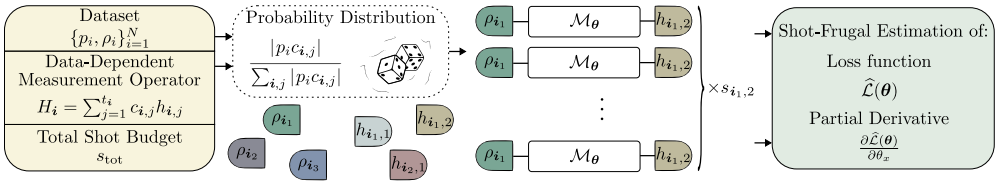


Figure 8.2: **Illustration of a generic variational QML framework.** Given a dataset of quantum states $\mathcal{S}$, the input states to the QML model are tensor product states of the form $\rho_{\boldsymbol{i}} = \rho_{i_1} \otimes ... \otimes \rho_{i_m}$. The number of samples m depends on the QML task at hand. The parameterized quantum model denoted $\mathcal{M}_{\boldsymbol{\theta}}$, acts on m copies of the input Hilbert space. Finally, an operator $H_{\boldsymbol{i}}$ is measured to estimate a quantity to be used when evaluating a loss function $\mathcal{L}(\boldsymbol{\theta})$. The latter evaluation is then inputted to a classical optimizer that proposes new parameters $\boldsymbol{\theta}$ in order to minimize the loss. Hence, one can repeat the quantum-classical loop of evaluations and updates until the desired stopping criteria are satisfied.

be measured. This is referred to as weighted random sampling (WRS). Finally, there exists a hybrid approach where one combines WDS with WRS - referred to as weighted hybrid sampling (WHS). Ref. [10] found that the WRS and WHS strategies performed similarly and they both significantly outperformed the UDS and WDS strategies on molecular ground state problems. Because of these results, we choose to focus on the WRS strategy in our work here.

While Rosalin was designed for chemistry problems, it was not designed for QML, where the number of terms in H is not the only consideration. As discussed below, QML problems involve a (potentially large) dataset of input states. Each input state requires a separate quantum circuit, and hence we are back to the situation of having a hard floor on the shots required due to these multiple input states. This ultimately provides the motivation for our work, which can be viewed as a generalization of Rosalin to the setting of QML.

8.3 Framework

8.3.1 Generic Variational QML Framework

Let us present our general framework for discussing (variational) QML methods; see Fig. 8.2 for an illustration. This framework is meant to unify multiple literature QML algorithms under one umbrella. We discuss how specific literature algorithms are special cases of this framework in Section 8.3.2.

In a generic QML setting, one has a training dataset composed of quantum states:

$$\mathcal{S} = \{\rho_i\}_{i=1}^{N} , \tag{8.8}$$

where each ρ_i is a trace-one positive semi-definite matrix, i.e., a density matrix. Each of these training states may come with an associated probability, with the associated probability distribution denoted as

$$\mathcal{P} = \{p_i\}_{i=1}^{N} . \tag{8.9}$$

In variational QML, one trains a parameterized quantum model, which we write as $\mathcal{M}_{\boldsymbol{\theta}}$ for some set of parameters $\boldsymbol{\theta}$. With a large degree of generality, we can assume that $\mathcal{M}_{\boldsymbol{\theta}}$ is a linear, completely-positive map. In general, $\mathcal{M}_{\boldsymbol{\theta}}$ could act on multiple copies (m copies) of the input Hilbert space. (Multiple copies allow for non-linear operations on the dataset, which can be important in certain classification tasks.) Hence we allow for the output of this action to be of the form:

$$\mathcal{M}_{\boldsymbol{\theta}}(\rho_{\boldsymbol{i}}) = \mathcal{M}_{\boldsymbol{\theta}}(\rho_{i_1} \otimes ... \otimes \rho_{i_m}) = \mathcal{M}_{\boldsymbol{\theta}} \left(\bigotimes_{\alpha=1}^{m} \rho_{i_\alpha} \right) , \tag{8.10}$$

where we employ the notation $\rho_{\boldsymbol{i}} := \rho_{i_1} \otimes ... \otimes \rho_{i_m}$. Given that we are allowing for multiple copies of the input space, one can define an effective dataset $\mathcal{S}_m$ composed of the tensor product of m states in $\mathcal{S}$ and an effective probability distribution $\mathcal{P}_m$.

A QML loss function is then defined in an operational manner so that it could be estimated on a quantum device. This involves considering the aforementioned mathematical objects as well as a measurement operator, or a set of measurement operators. We allow for the measurement operator to be tailored to the input state. Hence we write $H_{\boldsymbol{i}}$, with $\boldsymbol{i} = \{i_1, ..., i_m\}$, as the measurement operator when the input state on m copies of the Hilbert space is $\rho_{\boldsymbol{i}} = \rho_{i_1} \otimes ... \otimes \rho_{i_m}$. Moreover, each measurement operator can be decomposed into a linear combination of Hermitian matrices that can be directly measured:

$$H_{\boldsymbol{i}} = \sum_{j=1}^{t_{\boldsymbol{i}}} c_{\boldsymbol{i},j} h_{\boldsymbol{i},j} . \tag{8.11}$$

Generically, we could write the loss function as an average over training states

chosen from the effective dataset D_m:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \ell(E_{\boldsymbol{i}}(\boldsymbol{\theta}))\,. \tag{8.12}$$

Here, ℓ is an application-dependent function whose input is a measurable expectation value $E_{\boldsymbol{i}}(\boldsymbol{\theta})$. Specifically, this expectation value is associated with the $\rho_{\boldsymbol{i}}$ input state, with the form

$$E_{\boldsymbol{i}}(\boldsymbol{\theta}) = \mathrm{Tr}[\mathcal{M}_{\boldsymbol{\theta}}(\rho_{\boldsymbol{i}}) H_{\boldsymbol{i}}]\,. \tag{8.13}$$

There are many possible forms for the function ℓ. However, there are multiple QML proposals in the literature that involve a simple linear form:

$$\ell(E_{\boldsymbol{i}}(\boldsymbol{\theta})) = E_{\boldsymbol{i}}(\boldsymbol{\theta})\,, \tag{8.14}$$

and in this case, we refer to the overall loss function as a "linear loss function". Alternatively, non-linear functions are also possible, and we also consider polynomial functions of the form:

$$\ell(E_{\boldsymbol{i}}(\boldsymbol{\theta})) = \sum_{z=0}^{D} a_z [E_{\boldsymbol{i}}(\boldsymbol{\theta})]^z\,, \tag{8.15}$$

where D is the degree of the polynomial. In this case, we refer to the loss as a "polynomial loss function".

8.3.2 Examples of QML loss functions

Now let us illustrate how various QML loss functions proposed in the literature fall under the previous framework. Crucially, as shown in Fig. 8.3 these loss functions can be used for a wide range of QML tasks.

Variational quantum error correction

Variational quantum algorithms to learn device-tailored quantum error correction codes were discussed in Refs. [95, 45]. The loss function involves evaluating the input-output fidelity of the code, averaged over a set of input states. The input state is fed into the encoder $\mathcal{E}_{\boldsymbol{\theta}_1}$, then the noise channel $\mathcal{N}$ acts, followed by the decoder $\mathcal{D}_{\boldsymbol{\theta}_2}$. The concatenation of these three channels can be viewed as the overall channel

$$\mathcal{M}_{\boldsymbol{\theta}} = \mathcal{D}_{\boldsymbol{\theta}_2} \circ \mathcal{N} \circ \mathcal{E}_{\boldsymbol{\theta}_1}\,, \tag{8.16}$$

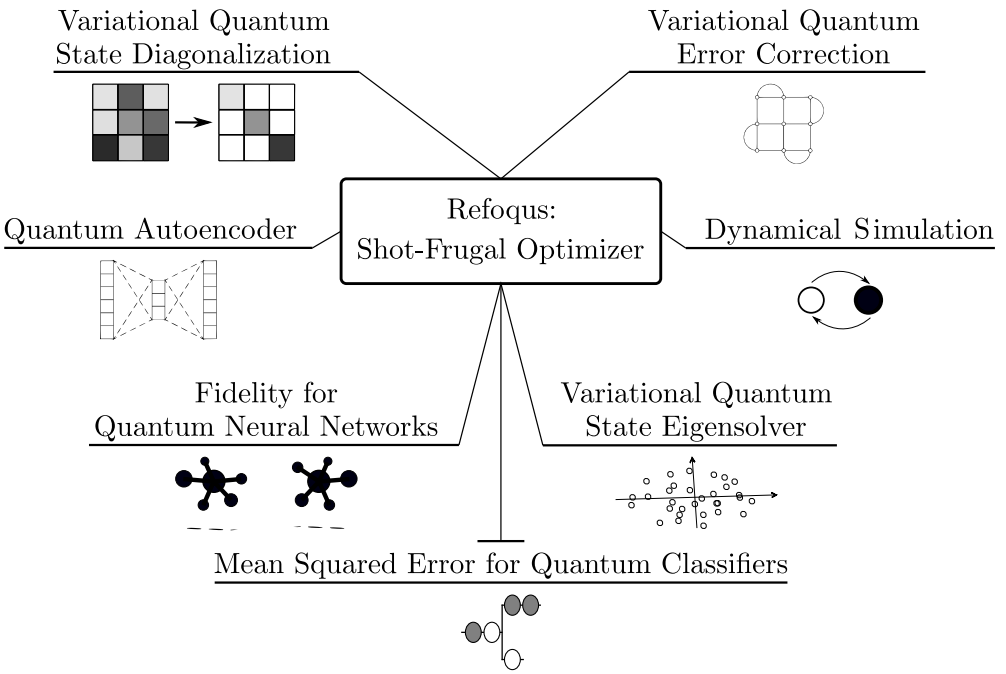


Figure 8.3: **Applications benefiting from Refoqus**. Many variational quantum algorithms employ training data, and many QML models (which naturally analyze datasets) are variational in nature. We give examples of both of these cases in this figure. Our Refoqus optimizer is relevant in all cases.

with parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$. Then the loss function is given by:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{|\psi_i\rangle \in \mathcal{S}} \frac{1}{|\mathcal{S}|} \operatorname{Tr}[|\psi_i\rangle\langle\psi_i| \, \mathcal{M}_{\boldsymbol{\theta}}(|\psi_i\rangle\langle\psi_i|)] \,, \tag{8.17}$$

where $\mathcal{S}$ is some appropriately chosen set of states. It is clear that this loss function is of the form in (8.12) with ℓ having the linear form in (8.14).

Quantum autoencoder

Inspired by the success of classical autoencoders, quantum autoencoders [161] were proposed to compress quantum data by reducing the number of qubits needed to represent the dataset. Consider a bipartite quantum system AB composed of n_{A} and n_{B} qubits, respectively, and let $\{p_i, |\psi_i\rangle\}$ be an ensemble of pure states on AB. The quantum autoencoder trains a gate sequence $U(\boldsymbol{\theta})$ to compress this ensemble into the

A subsystem, such that one can recover each state $|\psi_i\rangle$ with high fidelity from the information in subsystem A. One can think of B as the "trash" since it is discarded after the action of $U(\boldsymbol{\theta})$.

The original proposal [161] employed a loss function that quantified the overlap of the trash state with a fixed pure state:

$$\mathcal{L}_G(\boldsymbol{\theta}) = 1 - \mathrm{Tr}_B[|\mathbf{0}\rangle\langle\mathbf{0}|\,\rho_{\mathrm{B}}^{\mathrm{out}}] \tag{8.18}$$

$$= \mathrm{Tr}_{AB}[H_G U(\boldsymbol{\theta})\rho_{\mathrm{AB}}^{\mathrm{in}} U(\boldsymbol{\theta})^\dagger] \tag{8.19}$$

$$= \sum_i p_i\,\mathrm{Tr}_{AB}[H_G U(\boldsymbol{\theta})\,|\psi_i\rangle\langle\psi_i|\,U(\boldsymbol{\theta})^\dagger]\,. \tag{8.20}$$

Here, $\rho_{\mathrm{AB}}^{\mathrm{in}} = \sum_i p_i\,|\psi_i\rangle\langle\psi_i|$ is the ensemble-average input state, $\rho_{\mathrm{B}}^{\mathrm{out}} = \mathrm{Tr}_A[U(\boldsymbol{\theta})\rho_{\mathrm{AB}}^{\mathrm{in}}U(\boldsymbol{\theta})^\dagger]$ is the ensemble-average trash state, and the measurement operator is $H_G = \mathbb{1}_{AB} - \mathbb{1}_A \otimes |\mathbf{0}\rangle\langle\mathbf{0}|$.

Note that H_G is a global measurement operator, meaning it acts non-trivially on all qubits, which can lead to barren plateaus in the training landscape [42]. To remedy this issue, Ref. [42] proposed a loss function with a local measurement operator acting non-trivially only on a small number of qubits:

$$\mathcal{L}_L(\boldsymbol{\theta}) = 1 - \frac{1}{n_{\mathrm{B}}}\sum_{j=1}^{n_{\mathrm{B}}}\mathrm{Tr}_B\left[\left(|0\rangle\langle 0|_j \otimes \mathbb{1}_{\bar{j}}\right)\rho_{\mathrm{B}}^{\mathrm{out}}\right] \tag{8.21}$$

$$= \sum_i p_i\,\mathrm{Tr}_{AB}[H_L U(\boldsymbol{\theta})\,|\psi_i\rangle\langle\psi_i|\,U(\boldsymbol{\theta})^\dagger]\,, \tag{8.22}$$

where $H_L = \mathbb{1}_{\mathrm{AB}} - \frac{1}{n_{\mathrm{B}}}\sum_{j=1}^{n_{\mathrm{B}}}\mathbb{1}_{\mathrm{A}} \otimes |0\rangle\langle 0|_j \otimes \mathbb{1}_{\bar{j}}$, and $\mathbb{1}_{\bar{j}}$ is the identity on all qubits in B except the j-th qubit.

It is clear from (8.20) and (8.22) that both loss functions fall under our framework. Namely, they have the form in (8.12) with ℓ having the linear form in (8.14).

Dynamical simulation

Recently a QML-based algorithm was proposed for dynamical simulation [63]. Here the idea is to variationally compile the short-time dynamics into a time-dependent quantum neural network [44]. Then one uses the trained model to extrapolate to longer times. The training data for the compiling process can be taken to be product states, due to a generalization result from Ref. [39].

Let $U_{\Delta t}$ be a unitary associated with the short-time dynamics. Let $\{|\Psi_i^P\rangle\}_{i=1}^N$ be a set of product states used for training, where $|\Psi_i^P\rangle = \bigotimes_{j=1}^n |\psi_{i,j}\rangle$, and where n is

the number of qubits. Let $U(\boldsymbol{\theta})$ be the quantum neural network to be trained. Then the loss function is given by:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{N} \operatorname{Tr}\left[H_i V(\boldsymbol{\theta}) \left|\Psi_i^P\right\rangle\!\left\langle\Psi_i^P\right| V(\boldsymbol{\theta})^\dagger\right], \tag{8.23}$$

where we have defined $V(\boldsymbol{\theta}) := U(\boldsymbol{\theta})^\dagger U_{\Delta t}$. Here, the measurement operator is given by $H_i = \mathbb{1} - \frac{1}{n}\sum_{j=1}^{n} |\psi_{i,j}\rangle\langle\psi_{i,j}| \otimes \mathbb{1}_{\overline{j}}$, where $\overline{j}$ is the set of all qubits excluding the j-th qubit.

Once again, this loss function clearly falls under our framework, having the form in (8.12) with ℓ having the linear form in (8.14).

Fidelity for Quantum Neural Networks

Dissipative perceptron-based quantum neural networks (DQNNs) were proposed in Ref. [17] and their trainability was analyzed in Ref. [176]. The loss function is based on the fidelity between the idealized output state and the actual output state of the DQNN. Specifically, we are given access to training data $\{\left|\phi_i^{\text{in}}\right\rangle, |\phi_i^{\text{out}}\rangle\}_{i=1}^N$, and the DQNN is trained to output a state close to $|\phi_i^{\text{out}}\rangle$ when the input is $\left|\phi_i^{\text{in}}\right\rangle$.

For this application, a global loss function was considered with the form

$$\mathcal{L}_G(\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{N} \operatorname{Tr}\left[H_i^G \rho_i^{\text{out}}\right]. \tag{8.24}$$

Here, $\rho_i^{\text{out}} = \mathcal{M}_{\boldsymbol{\theta}}(\left|\phi_i^{\text{in}}\right\rangle\!\left\langle\phi_i^{\text{in}}\right|)$ is the output state of the DQNN, which is denoted by $\mathcal{M}_{\boldsymbol{\theta}}$. The measurement operator is the projector orthogonal to the ideal output state: $H_i^G = \mathbb{1} - |\phi_i^{\text{out}}\rangle\langle\phi_i^{\text{out}}|$.

To avoid the issue of barren plateaus, a local loss function was also considered in Ref. [176]:

$$\mathcal{L}_L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{N} \operatorname{Tr}\left[H_i^L \rho_i^{\text{out}}\right], \tag{8.25}$$

where the measurement operator is

$$H_i^L = \mathbb{1} - \frac{1}{n_{\text{out}}} \sum_{j=1}^{n_{\text{out}}} \left|\psi_{i,j}^{\text{out}}\right\rangle\!\left\langle\psi_{i,j}^{\text{out}}\right| \otimes \mathbb{1}_{\overline{j}}. \tag{8.26}$$

This loss function is relevant whenever the ideal output states have a tensor-product form across the n_{out} output qubits, i.e., of the form $|\phi_i^{\text{out}}\rangle = \left|\psi_{i,1}^{\text{out}}\right\rangle \otimes \cdots \otimes \left|\psi_{i,n_{\text{out}}}^{\text{out}}\right\rangle$.

Clearly, these two loss functions fall under our framework, having the form in (8.12) with ℓ having the linear form in (8.14).

Variational quantum state eigensolver

Near-term methods for quantum principal component analysis have recently been proposed, including the variational quantum state eigensolver (VQSE) [41] and the variational quantum state diagonalization (VQSD) algorithm. Let us first discuss VQSE.

The goals of VQSE are to estimate the m largest eigenvalues of a density matrix ρ and to find quantum circuits that prepare the associated eigenvectors. When combined with a method to prepare the covariance matrix as a density matrix, VQSE can be used for principal component analysis. Note that such a method was proposed in Ref. [70], where it was shown that choosing $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ prepares the covariance matrix for a given quantum dataset $\{p_i, |\psi_i\rangle\}$.

The VQSE loss function can be written as an energy:

$$\mathcal{L}(\boldsymbol{\theta}) = \operatorname{Tr}\left[HU(\boldsymbol{\theta})\rho U(\boldsymbol{\theta})^\dagger\right] \tag{8.27}$$

$$= \sum_i p_i \operatorname{Tr}\left[HU(\boldsymbol{\theta})\,|\psi_i\rangle\langle\psi_i|\,U(\boldsymbol{\theta})^\dagger\right] \tag{8.28}$$

where $U(\boldsymbol{\theta})$ is a parameterized unitary that is trained to approximately diagonalize ρ. Note that we inserted the formula $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ in order to arrive at (8.28).

The measurement operator H is chosen to be non-degenerate over its m-lowest energy levels. For example, one can choose a global version of this operator:

$$H = \mathbb{1} - \sum_{j=1}^{m} r_j |e_j\rangle\langle e_j|\,, \quad r_j > 0 \tag{8.29}$$

or a local version of this operator:

$$H = \mathbb{1} - \sum_{j=1}^{n} r_j Z_j, \quad r_j \in \mathbb{R}\,, \tag{8.30}$$

where Z_j is the Pauli-z operator on the j-th qubit, and with appropriately chosen real coefficients r_j to achieve non-degeneracy over the m-lowest energy levels. Regardless, it is clear that the loss function in (8.28) falls under our framework, having the form in (8.12) with ℓ having the linear form in (8.14).

Variational quantum state diagonalization

The goal of the VQSD algorithm [110] is essentially the same as that of VQSE, i.e., to diagonalize a target quantum state ρ. Let us use:

$$\tilde{\rho} := U(\boldsymbol{\theta})\rho U(\boldsymbol{\theta})^\dagger \tag{8.31}$$

to denote the state after the attempted diagonalization. Here we omit the $\boldsymbol{\theta}$ dependency for simplicity of notation.

In contrast to VQSE, the VQSD loss function depends quadratically on the quantum state. Specifically, global and lost functions have been proposed, respectively given by

$$\mathcal{L}_G(\boldsymbol{\theta}) = \mathrm{Tr}\big[\rho^2\big] - \mathrm{Tr}\big[\mathcal{Z}(\tilde{\rho})^2\big], \tag{8.32}$$

$$\mathcal{L}_L(\boldsymbol{\theta}) = \mathrm{Tr}\big[\rho^2\big] - \frac{1}{n}\sum_{j=1}^{n} \mathrm{Tr}\big[\mathcal{Z}_j(\tilde{\rho})^2\big]\,. \tag{8.33}$$

Here, $\mathcal{Z}$ and $\mathcal{Z}_j$ are quantum channels that dephase (i.e., destroy the off-diagonal elements) in the global standard basis and in the local standard basis on qubit j, respectively.

We can rewrite the terms in the global loss using:

$$\mathrm{Tr}\big[\rho^2\big] = \mathrm{Tr}[(\rho\otimes\rho)\mathrm{SWAP}]\,, \tag{8.34}$$

$$\mathrm{Tr}\big[\mathcal{Z}(\tilde{\rho})^2\big] = \mathrm{Tr}\Big[(\rho\otimes\rho)W_G^\dagger(|\mathbf{0}\rangle\langle\mathbf{0}|\otimes\mathbb{1})W_G\Big]\,, \tag{8.35}$$

where $SWAP$ denotes the swap operator, and where W_G corresponds to the layers of CNOTs used in the so-called diagonalized inner product (DIP) test circuit [110]. Hence, we obtain:

$$\mathcal{L}_G(\boldsymbol{\theta}) = \mathrm{Tr}[(\rho\otimes\rho)H_G] \tag{8.36}$$

$$= \sum_{i,i'} p_i p_{i'}\,\mathrm{Tr}[(|\psi_i\rangle\langle\psi_i|\otimes|\psi_{i'}\rangle\langle\psi_{i'}|)H_G] \tag{8.37}$$

where $H_G = \mathrm{SWAP} - U_G^\dagger(|\mathbf{0}\rangle\langle\mathbf{0}|\otimes\mathbb{1})U_G$. Note that we inserted the relation $\rho = \sum_i p_i\,|\psi_i\rangle\langle\psi_i|$ in order to arrive at (8.37). Also note that one can think of the $q_{\boldsymbol{i}} := p_i p_{i'}$ as defining a probability distribution over the index $\boldsymbol{i} = \{i, i'\}$. Hence it is clear that (8.37) falls under our framework, with $m = 2$ copies of the input Hilbert space, and

with ℓ having the linear form in (8.14).

Similarly, for the local loss, we can write

$$\mathcal{L}_L(\boldsymbol{\theta}) = \mathrm{Tr}[(\rho \otimes \rho) H_L] \tag{8.38}$$

$$= \sum_{i,i'} p_i p_{i'} \, \mathrm{Tr}[(|\psi_i\rangle\langle\psi_i| \otimes |\psi_{i'}\rangle\langle\psi_{i'}|) H_L] \tag{8.39}$$

where $H_L = \mathrm{SWAP} - (1/n) \sum_{j=1}^{n} H_{L,j}$, and $H_{L,j}$ is the Hermitian operator that is measured for the partial diagonalized inner product (PDIP) test [110]. Once again, the local loss falls under our framework, with $m = 2$ copies of the input Hilbert space, and with ℓ having the linear form in (8.14).

Mean squared error for quantum classifiers

The mean squared error (MSE) loss function is widely employed in classical machine learning. Moreover, it has been used in the context of quantum neural networks, e.g., in Ref. [45]. Given a set of labels $y_i \in \mathbb{R}$ for a dataset $\{\rho_i\}_{i=1}^{N}$, and a set of predictions from a machine learning model denoted $\tilde{y}_i(\boldsymbol{\theta})$, the MSE loss is computed as:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \tilde{y}_i(\boldsymbol{\theta}))^2 \, . \tag{8.40}$$

In the case of a quantum model, there is freedom in specifying how to compute the prediction $\tilde{y}_i(\boldsymbol{\theta})$. Typically, this will be estimated via an expectation value, such as $\tilde{y}_i(\boldsymbol{\theta}) = \mathrm{Tr}[\mathcal{M}_{\boldsymbol{\theta}}(\rho_i) H_i] = E_i(\boldsymbol{\theta})$. In this case, the loss function in (8.40) would be a quadratic function of expectation value $E_i(\boldsymbol{\theta})$. Hence, this falls under our framework, with ℓ having the polynomial form in (8.15) with degree $D = 2$.

8.4 Unbiased estimators for gradients of QML losses

Gradient-based optimizers are commonly used when optimizing QML models. In shot frugal versions of these approaches, one of the key aspects is the construction of an unbiased estimator of the gradient [189, 109, 72]. There are two types of loss functions we will consider in this work; those that have a linear dependence on expectation values and those that have a non-linear dependence given by a polynomial function. We will see that for these two types of losses one can construct unbiased estimators of the gradients and that it is also possible to define sampling strategies that massively

reduce the number of shots needed to evaluate such an estimator.

Previous work considered how to construct unbiased estimators of QML gradients. However, the shot frugal resource allocation strategies presented were sub-optimal [189] and leave room for improvement. Furthermore, the more sophisticated shot allocation methods presented in [10] were purpose-built for VQE-type cost functions. Here we unify approaches from these two works and show how one can employ more sophisticated shot allocation strategies in a general QML setting.

First, we consider the simpler case of linear loss functions where we show one can directly employ the parameter shift rule to construct an unbiased estimator for the gradient. Then we turn our attention to polynomial loss functions where we present a general form for an unbiased estimator of the gradient. In both cases, we show that shots can be allocated according to the expansion coefficients in the expressions we derive. These in turn depend on the coefficients in the operators to be measured and the set of quantum states used in the QML data set. Such shot-allocation schemes are an important ingredient in the design of our QML shot-frugal optimizer in the next section.

8.4.1 Loss functions linear in the quantum circuit observables

Using the parameter shift rule

Loss functions that have a linear dependence on expectation values of observables are straightforward to consider. As shown in Ref. [189], the parameter shift rule can be used directly to construct unbiased estimators of the gradient of these loss functions. Previously, we wrote our general linear loss function as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \ell(E_{\boldsymbol{i}}(\boldsymbol{\theta}))\,. \tag{8.41}$$

Let us not consider the case where

$$E_{\boldsymbol{i}}(\boldsymbol{\theta}) = \mathrm{Tr}(\mathcal{M}_{\boldsymbol{\theta}}(\rho_{\boldsymbol{i}})H_{\boldsymbol{i}}) \quad \text{and} \quad \ell(E_{\boldsymbol{i}}(\boldsymbol{\theta})) = E_{\boldsymbol{i}}(\boldsymbol{\theta})\,.$$

By expanding the measurement operator as $H_{\boldsymbol{i}} = \sum_{j=1}^{t_{\boldsymbol{i}}} c_{\boldsymbol{i},j} h_{\boldsymbol{i},j}$, we can then write this loss function as follows

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j} \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle, \tag{8.42}$$

where $q_{\boldsymbol{i},j} = p_{\boldsymbol{i}} c_{\boldsymbol{i},j}$ and $\mathrm{Tr}[\mathcal{M}_{\boldsymbol{\theta}}(\rho_{\boldsymbol{i}}) h_{\boldsymbol{i},j}] = \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$.

Consider the partial derivative of this loss function with respect to the parameter θ_x,

$$\frac{\partial \mathcal{L}}{\partial \theta_x} = \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j} \frac{\partial \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle}{\partial \theta_x} \,. \tag{8.43}$$

Suppose we assume that the quantum channel $\mathcal{M}_{\boldsymbol{\theta}}(\rho_i)$ is a unitary channel:

$$\mathcal{M}_{\boldsymbol{\theta}}(\rho_i) = U(\boldsymbol{\theta}) \rho_i U(\boldsymbol{\theta})^\dagger \,, \tag{8.44}$$

where $U(\boldsymbol{\theta})$ is a trainable quantum circuit whose parametrized gates are generated by Pauli operators. This assumption allows us to directly apply the parameter shift rule (see Sec. 8.2). This leads to

$$\frac{\partial \mathcal{L}}{\partial \theta_x} = \frac{1}{2} \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j} \big(\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta} + \boldsymbol{\delta}_x \frac{\pi}{2})\rangle - \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta} - \boldsymbol{\delta}_x \frac{\pi}{2})\rangle \big) \,. \tag{8.45}$$

Therefore, an unbiased estimator for the gradient can be obtained by combining two unbiased estimators for the loss function. Defining $\widehat{g}_x(\boldsymbol{\theta})$ to be an unbiased estimator of the x-th component of the gradient,

$$\widehat{g}_x(\boldsymbol{\theta}) = \frac{1}{2} [\widehat{\mathcal{L}}(\boldsymbol{\theta} + \boldsymbol{\delta}_x \frac{\pi}{2}) - \widehat{\mathcal{L}}(\boldsymbol{\theta} + \boldsymbol{\delta}_x \frac{\pi}{2})] \,, \tag{8.46}$$

where $\widehat{\mathcal{L}}(\boldsymbol{\theta} \pm \boldsymbol{\delta}_x \frac{\pi}{2})$ are unbiased estimators for the loss function at the different shifted parameter values needed when employing the parameter shift rule. This means that for loss functions that are linear in the expectation values recovered from the quantum circuit, one can then use an unbiased estimator for the cost evaluated at different parameter values to return an unbiased estimator for the gradient.

We can therefore distribute the shots according to the coefficients $q_{\boldsymbol{i},j}$ when evaluating a single or multi-shot estimate of $\widehat{\mathcal{L}}(\boldsymbol{\theta} \pm \boldsymbol{\delta}_x \frac{\pi}{2})$. This can be done by constructing a probability distribution according to the probabilities $\epsilon_{\boldsymbol{i},j} = |q_{\boldsymbol{i},j}| / \sum_{\boldsymbol{i},j} |q_{\boldsymbol{i},j}|$. Note that this distribution strategy relies on the construction of two unbiased estimators of the loss function, which are then combined. In the next section, we show how one can construct such estimators. In practice, we will use the same total number of shots to evaluate both estimators as they have the same expansion weights and are therefore equally important.

Unbiased estimators for Linear loss functions

Let s_{tot} represent the total number of shots. We denote $\widehat{\mathcal{E}}_{\boldsymbol{i},j}$ as the estimator for $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$, and $\widehat{\mathcal{L}}(\boldsymbol{\theta})$ the estimator for the loss. That is,

$$\widehat{\mathcal{L}}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j}\widehat{\mathcal{E}}_{\boldsymbol{i},j}\,, \quad \text{with} \quad \widehat{\mathcal{E}}_{\boldsymbol{i},j} = \frac{1}{\mathbb{E}[s_{\boldsymbol{i},j}]}\sum_{k=1}^{s_{\boldsymbol{i},j}} r_{\boldsymbol{i},j,k}\,. \tag{8.47}$$

Here, $s_{\boldsymbol{i},j}$ is the number of shots allocated to the measurement of $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$. Note that $s_{\boldsymbol{i},j}$ may be a random variable. As we will work in terms of the total shot budget for the estimation, s_{tot}, we impose $\sum_{\boldsymbol{i},j} s_{\boldsymbol{i},j} = s_{\text{tot}}$. Also, each $r_{\boldsymbol{i},j,k}$ is an independent random variable associated with the k-th single-shot measurement of $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$. We will assume that $\mathbb{E}[s_{\boldsymbol{i},j}] > 0$ for all $\boldsymbol{i}, j$. Using these definitions we can show that this defines an unbiased estimator for the loss function in the following proposition.

Proposition 1. *Let $\widehat{\mathcal{L}}$ be the estimator defined in Eq.* (8.47)*. $\widehat{\mathcal{L}}$ is an unbiased estimator for the cost function $\mathcal{L}(\boldsymbol{\theta})$ defined in Eq.* (8.12)*.*

Proof.

$$\begin{aligned}\mathbb{E}[\widehat{\mathcal{L}}] &= \mathbb{E}[\sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j}\frac{1}{\mathbb{E}[s_{\boldsymbol{i},j}]}\sum_{k=1}^{s_{\boldsymbol{i},j}} r_{\boldsymbol{i},j,k}] \\ &= \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j}\frac{1}{\mathbb{E}[s_{\boldsymbol{i},j}]}\mathbb{E}[\sum_{k=1}^{s_{\boldsymbol{i},j}} r_{\boldsymbol{i},j,k}].\end{aligned} \tag{8.48}$$

Here it is useful to recall Wald's equation. Wald's equation states that the expectation value of the sum of N real-valued, identically distributed, random variables, X_i, can be expressed as

$$\mathbb{E}[\sum_{i=1}^{N} X_i] = \mathbb{E}[N]\mathbb{E}[X_1], \tag{8.49}$$

where N is a random variable that does not depend on the terms of the sum. In our case, each shot is indeed independent and sampled from the same distribution. Furthermore, the total number of shots does not depend on the sequence of single-shot measurements. Therefore,

$$\mathbb{E}[\widehat{\mathcal{L}}] = \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j}\frac{\mathbb{E}[s_{\boldsymbol{i},j}]}{\mathbb{E}[s_{\boldsymbol{i},j}]}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle = \mathcal{L}(\boldsymbol{\theta}). \tag{8.50}$$

□

Using the above result, we arrive at the following corollary.

Corollary 1. *Let $\widehat{g}_x(\boldsymbol{\theta})$ be the estimator defined in Eq. (8.46). $\widehat{g}_x(\boldsymbol{\theta})$ is an unbiased estimator for the x-th component of the gradient.*

Proof. The proof follows by taking the expectation values of Eq. (8.46) and employing the result from Prop. 1 and the parameter shift rule. □

Having now constructed an estimator for the loss function as outlined in the previous section combining two single or multi-shot estimates of this function evaluated at the required parameter values will lead to an unbiased estimator of the gradient.

8.4.2 Loss functions with polynomial dependence on the quantum circuit observables

Constructing an unbiased estimator of the gradient

In the case of non-linear dependence on the observables produced by a quantum circuit, estimating the gradient is not as simple as simply applying to parameter shift rule. However, we can still derive estimators for the gradient when the non-linearity is described by a polynomial function. We begin with the general expression for the loss functions we consider in this work

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \ell(E_{\boldsymbol{i}}(\boldsymbol{\theta}))\,. \tag{8.51}$$

Now constructing a polynomial loss function of degree D requires that

$$\ell(E_{\boldsymbol{i}}(\boldsymbol{\theta})) = \sum_{z=0}^{D} a_z [E_{\boldsymbol{i}}(\boldsymbol{\theta})]^z\,, \tag{8.52}$$

which leads to the expression of the loss function,

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i},z} p_{\boldsymbol{i},z} \big[\sum_{j} c_{\boldsymbol{i},j} \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta}) \rangle \big]^z\,, \tag{8.53}$$

where $p_{\boldsymbol{i},z} = p_{\boldsymbol{i}} a_z$. Taking the derivative with respect to θ_x leads to

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x} = \sum_{\boldsymbol{i},z} p_{\boldsymbol{i},z} z \big(\sum_{j} c_{\boldsymbol{i},j} \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta}) \rangle \big)^{z-1} \big(\sum_{j'} c_{\boldsymbol{i},j'} \frac{\partial \langle h_{\boldsymbol{i},j'}(\boldsymbol{\theta}) \rangle}{\partial \theta_x} \big)\,. \tag{8.54}$$

Using the multinomial theorem and given J Hamiltonian terms, we can expand the second sum in the above expression,

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x} = \sum_{\boldsymbol{i},z} p_{\boldsymbol{i},z} z \sum_{b_1+b_2+\cdots+b_J=z-1} \binom{z-1}{b_1, b_2, \cdots b_J} \prod_j (c_{\boldsymbol{i},j} \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle)^{b_j} \Big(\sum_{j'} c_{\boldsymbol{i},j'} \frac{\partial \langle h_{\boldsymbol{i},j'}(\boldsymbol{\theta})\rangle}{\partial \theta_x} \Big), \tag{8.55}$$

where $\binom{z-1}{b_1,b_2,\cdots b_J} = \frac{(z-1)!}{b_1! b_2! \cdots b_J!}$ and b_j are non-negative integers. Therefore, we need to construct unbiased estimators of the terms $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}$ and use the previously established gradient estimators with the parameter shift rule. Then we will need to consider how to distribute shots among this estimator. Rewriting Eq. (8.55) leads to

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x} = \sum_{\boldsymbol{i},z} \sum_{b_1+b_2+\cdots+b_J=z-1} p_{\boldsymbol{i},z} z \frac{(z-1)!}{b_1! b_2! \cdots b_J!} \sum_{j'} \prod_j c_{\boldsymbol{i},j}^{b_j} c_{\boldsymbol{i},j'} \left[\frac{\partial \langle h_{\boldsymbol{i},j'}(\boldsymbol{\theta})\rangle}{\partial \theta_x} \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j} \right]. \tag{8.56}$$

Therefore we can distribute the shots according to the magnitude of the expansion terms $\prod_j c_{\boldsymbol{i},j}^{b_j} c_{\boldsymbol{i},j'} p_{\boldsymbol{i},z} z \frac{(z-1)!}{b_1! b_2! \cdots b_J!}$. Once again we can construct an unbiased estimator with the normalized magnitude of these terms defining the probabilities. We now explore how to construct an unbiased estimator for the term $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}$, which is essential to construct an unbiased estimator for the gradients of these kinds of loss functions.

Constructing an unbiased estimator of polynomial terms

In order to construct an unbiased estimator for the gradient we need an unbiased estimator for terms of the form

$$\frac{\partial \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle}{\partial \theta_x} \prod_j \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}. \tag{8.57}$$

We can use the parameter shift rule for the term on the left-hand side. For the term in the product as each j index corresponds to a different operator in the Hamiltonian, each term will be independent. Therefore, we need to construct estimators for terms of the form $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^z$. This leads us to the following proposition.

Proposition 2. *Let $\widehat{\xi}_{\boldsymbol{i},j}$ be an estimator defined as*

$$\widehat{\xi}_{\boldsymbol{i},j} = \frac{1}{\mathbb{E}[\binom{s_{\boldsymbol{i},j}}{z}]} \sum h^*(r_{\boldsymbol{i},j,k_{\alpha_1}}, ..., r_{\boldsymbol{i},j,k_{\alpha_z}}), \tag{8.58}$$

where the summation is over all subscripts $1 \leqslant \alpha_1 < \alpha_2 < ... < \alpha_z \leqslant s_{\boldsymbol{i},j}$ and $h^*(r_{\boldsymbol{i},j,k_1}, ..., r_{\boldsymbol{i},j,k_z}) = \prod_{\beta=1}^{z} r_{\boldsymbol{i},j,k_{\alpha_\beta}}$. $\widehat{\xi}_{\boldsymbol{i},j}$ is an unbiased estimator for the term $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^z$ estimated with $s_{\boldsymbol{i},j}$ shots where $s_{\boldsymbol{i},j} \geqslant z$.

Eq. (8.58) is inspired by the form of a U-statistic for $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}$. The theory of U-statistics was initially introduced by Hoeffding in the late 1940s [86] and has a wide range of applications. U-statistics presents a methodology on how to use observations of estimable parameters to construct minimum variance unbiased estimates of more complex functions of the estimable parameters. Taking the expectation value and using the U-statistic formalism one can arrive at the desired result. We refer the interested reader to the proof available in Appendix B of [136].

Bringing this all together we can formulate a proposition regarding an unbiased estimator for gradients of loss functions with polynomial dependence.

Proposition 3.

$$\begin{aligned}\hat{g}_x(\boldsymbol{\theta}) = \sum_{\boldsymbol{i},z} \sum_{b_1+b_2+\cdots+b_J=z-1} & \\ p_{\boldsymbol{i},z} z \frac{(z-1)!}{b_1! b_2! \cdots b_J!} & \\ \sum_{j'} \prod_{j=1}^{J} c_{\boldsymbol{i},j}^{b_j} c_{\boldsymbol{i},j'} \frac{1}{\mathbb{E}[s_{\boldsymbol{i},j,j',\boldsymbol{b}}]} & \sum_{k=1}^{s_{\boldsymbol{i},j,j',\boldsymbol{b}}} \left(r^+_{\boldsymbol{i},j,k} - r^-_{\boldsymbol{i},j,k}\right) \\ \frac{1}{\mathbb{E}[\binom{s_{\boldsymbol{i},j,j',\boldsymbol{b}}}{b_j}]} \sum \prod_{\beta=1}^{z} & r_{\boldsymbol{i},j,k_{\alpha_\beta}} ,\end{aligned} \tag{8.59}$$

where

$$\mathbb{E}[r^{\pm}_{\boldsymbol{i},j,1}] = \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta} \pm \boldsymbol{\delta}_x \frac{\pi}{2})\rangle \tag{8.60}$$

and $\boldsymbol{b} = (b_1, b_2, ..., b_J)$. The final sum is over all subscripts $1 \leqslant \alpha_1 < \alpha_2 < ... < \alpha_z \leqslant s_{\boldsymbol{i},j}$ and $h^*(r_{\boldsymbol{i},j,k_1}, ..., r_{\boldsymbol{i},j,k_z}) = \prod_{\beta=1}^{z} r_{\boldsymbol{i},j,k_{\alpha_\beta}}$. $\hat{g}_x(\boldsymbol{\theta})$ is a unbiased estimator for $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x}$

Proof. This can be seen to be true by taking the expectation values and invoking the propositions previously established. □

These same techniques can be used to construct unbiased estimators of the loss

function. In order to provide a concrete example of using the above propositions, we consider the special case of the MSE loss function.

Constructing an estimator for the gradient of the MSE loss function

To clarify the notation used above, let us focus on the special case of the MSE loss function. We consider a slightly more general form than the MSE cost introduced above in Eq. (8.40). Consider a set of labels $y_{\boldsymbol{i}} \in \mathbb{R}$ for a dataset $\{\rho_{\boldsymbol{i}}\}_{\boldsymbol{i}=1}^{N}$, composed of the tensor product of m states. The predictions from the quantum machine learning model are denoted $\tilde{y}_i(\boldsymbol{\theta}) = \sum_j c_{\boldsymbol{i},j}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$. Therefore, we can write the loss as

$$\mathcal{L}_{MSE}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \left[y_{\boldsymbol{i}} - \sum_j c_{\boldsymbol{i},j}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle\right]^2 . \tag{8.61}$$

Expanding the previous equation leads to

$$\begin{aligned}\mathcal{L}_{MSE}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \Big[y_{\boldsymbol{i}}^2 - y_{\boldsymbol{i}} \sum_j c_{\boldsymbol{i},j}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle \\ + \big(\sum_j c_{\boldsymbol{i},j}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle\big)^2\Big].\end{aligned} \tag{8.62}$$

Evaluating the partial derivative with respect to θ_x gives,

$$\begin{aligned}\frac{\partial \mathcal{L}_{MSE}(\boldsymbol{\theta})}{\partial \theta_x} = \sum_{\boldsymbol{i},j,j'} -p_{\boldsymbol{i}} c_{\boldsymbol{i},j} \frac{\partial \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle}{\partial \theta_x} \\ + 2p_{\boldsymbol{i}} c_{\boldsymbol{i},j'} c_{\boldsymbol{i},j} \langle h_{\boldsymbol{i},j'}(\boldsymbol{\theta})\rangle \frac{\partial \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle}{\partial \theta_x} .\end{aligned} \tag{8.63}$$

We can distribute the total number of shots s_{tot} among these terms according to the relative magnitudes of the $p_{\boldsymbol{i}} c_{\boldsymbol{i},j'}$ and $2p_{\boldsymbol{i}} c_{\boldsymbol{i},j'} c_{\boldsymbol{i},j}$ coefficients. This can be achieved by sampling from a multinomial probability distribution where the normalized magnitude of these coefficients defines the probabilities.

It is important to note that some estimators have a different minimum number of shots than others. For example, the estimator for the last term in the above equation, involving a gradient and a direct expectation value, requires a total of 3 for different circuit evaluations. The estimator for this term can be written as

$$\widehat{\mathcal{D}}_{\boldsymbol{i},j,j'} = \frac{1}{2}\widehat{\mathcal{E}}_{\boldsymbol{i},j'}\big(\widehat{\mathcal{E}}^{+}_{\boldsymbol{i},j} - \widehat{\mathcal{E}}^{-}_{\boldsymbol{i},j}\big) . \tag{8.64}$$

Therefore, in this case, the minimum number of shots given to any term can be set to

3 to ensure every estimation of any term in Eq. (8.61) will always be unbiased. We expand on this consideration below.

8.4.3 Distributing the shots among estimator terms

As previously mentioned the shots can be distributed according to a multinomial distribution with probabilities given by the magnitude of the constant factors that appear in the expression for the above estimators. We note that the shots assigned to a given term may be zero. In order to ensure the estimate of the above term using a given number of shots we need to ensure the estimate of each term is also unbiased. One needs at least 2 shots to produce an unbiased estimate of the gradient. For terms of the form $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}$ one needs at least b_j shots. Therefore, care needs to be taken when distributing shots to ensure that each term measured has sufficiently many. One way to ensure this is the case is to distribute shots in multiples of the largest number of shots needed to evaluate any one term. Any leftover shots can then be distributed equally across the terms to be measured.

Each term itself may consist of a product of several expectation values, each with a different required minimum number of shots. The shots distribution with each term can be selected to correspond to this required minimum number of shots per term. To make this concrete consider a term of the form in Eq. (8.57). Estimating the gradient will take at least 2 shots. Estimating the product term will take at least $\sum_{j=1}^{J} b_j$ shots. If we are given $s_{\boldsymbol{i},j}$ shots to use to estimate this term we can assign $\lfloor 2(s_{\boldsymbol{i},j}/(2+\sum_{j=1}^{J} b_j)\rfloor$ to the first term and $\lfloor \sum_{j=1}^{J} b_j(s_{\boldsymbol{i},j}/(2+\sum_{j=1}^{J} b_j)\rfloor$ to the product term, distributing any remaining shots equally among both.

8.5 The Refoqus optimizer

Now that we have defined unbiased estimators of the gradient, we have the tools needed to present our new optimizer. We call our optimizer Refoqus, which stands for REsource Frugal Optimizer for QUantum Stochastic gradient descent. This optimizer is tailored to QML tasks. QML tasks have the potential to incur large shot overheads because of large training datasets as well as measurement operators composed of a large number of terms. With Refoqus, we achieve shot frugality first by leveraging the gCANS [72] rule for allocating shots at each iteration step and second by allocating these shots to individual terms in the loss function via random sampling over the training dataset and over measurement operators. This random sampling allows us

to remove the shot floor imposed by deterministic strategies (while still achieving an unbiased estimator for the gradient), hence unlocking the full shot frugality of our optimizer.

The key insight needed for the construction of the Refoqus protocol is that cost functions that have a linear or polynomial dependence on measurable hermitian matrices, and their gradients can always be written in a form consisting of a summation of different measurable quantities with expansion coefficients. These coefficients can in turn be used to define a multinomial probability distribution to guide the allocation of the shots to each term when evaluating the loss function or its gradient.

We outline the Refoqus optimizer in Algorithm 3. Given a number of shots to distribute among Hamiltonian terms, one evaluates the gradient components and the corresponding variances with the *iEvaluate* subroutine (Line 3). We follow the gCANS procedure to compute the shot budget for each iteration (Line 12). We refer to [72] for more details on gCANS. The iterative process stops until the total shot budget has been used.

The hyperparameters that we employ are similar to those of Rosalin [10], and will come with similar recommendations on how to set them. For instance, the Lipshitz constant L bounds the largest possible value of the derivative. Hence, it depends on the loss expression but can be set as $M = \sum_{\boldsymbol{i},j} |q_{\boldsymbol{i},j}|$ according to [109]. Moreover, a learning rate satisfying $0 < \alpha < 2/L$ can be used.

8.6 Convergence Guarantees

The framework for Refoqus leverages the structure and the update rule of the gCANS optimizer presented in [72]. Therefore, we can apply the same arguments introduced to show geometric convergence. We repeat the arguments and assumptions needed for this convergence result here for convenience.

Proposition 4. *Provided the loss function satisfies the assumptions stated below the stochastic gradient descent routine used in Refoqus achieves geometric convergence to the optimal value of the cost function. That is,*

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta})^{(t)}] - \mathcal{L}^* = \mathcal{O}(\gamma^t) \tag{8.65}$$

where t labels the iteration, $\mathcal{L}^*$ is the optimal value of the loss function, and $0 < \gamma < 1$.

The update rule used in this work and introduced in [72] and the proof presented

Algorithm 3: The optimization loop used in *Refoqus*. The function $iEvaluate(\boldsymbol{\theta}, \boldsymbol{s}, \{f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j})\})$ evaluates the gradient at $\boldsymbol{\theta}$ returning, a vector of the gradient estimates $\boldsymbol{g}$ and their variances $\boldsymbol{S}$. The vectors are calculated using $s_0 s_x$ shots to estimate the x-th component of the gradient and its variance, where s_0 is the minimum number of shots required to obtain an unbiased estimator. Note that shots for each component estimation are distributed in multiples of s_0, according to a multinomial distribution determined by the expansion coefficients that define the gradient estimator. These expansion coefficients are defined by the function $f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j})$, which returns the probability of measuring the term corresponding to the coefficients $p_{\boldsymbol{i}}$, $c_{\boldsymbol{i},j}$. For the case of linear loss functions $f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j}) = \frac{|p_{\boldsymbol{i}} c_{\boldsymbol{i},j}|}{\sum_{i,j} p_{\boldsymbol{i}} c_{\boldsymbol{i},j}}$.

Input: Learning rate α, starting point $\boldsymbol{\theta}_0$, min number of shots per estimation $s_{\min}$, number of shots that can be used in total s_{max}, Lipschitz constant L, running average constant μ, a vector of the least number of shots needed for each gradient estimate s_0, which is loss function dependent and $f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j})$, which is also loss function dependent.

Output: $\boldsymbol{\theta}$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$, $s_{\text{tot}} \leftarrow 0$, $\boldsymbol{g} \leftarrow (0, ..., 0)^T$, $\boldsymbol{S} \leftarrow (0, ..., 0)^T$, $\boldsymbol{s} \leftarrow (s_{\min}, ..., s_{\min})^T$, $\boldsymbol{\chi}' \leftarrow (0, ..., 0)^T$, $\boldsymbol{\chi} \leftarrow (0, ..., 0)^T$, $\boldsymbol{\xi} \leftarrow (0, ..., 0)^T$, $\boldsymbol{\xi}' \leftarrow (0, ..., 0)^T$, $t \leftarrow 0$;

while $s_{tot} < s_{max}$ **do**

  $\boldsymbol{g}, \boldsymbol{S} \leftarrow iEvaluate(\boldsymbol{\theta}, \boldsymbol{s}\{f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j})\})$;

  $s_{\text{tot}} \leftarrow s_{\text{tot}} + \sum_x s_0 s_x$;

  $\boldsymbol{\chi'} \leftarrow \mu\boldsymbol{\chi} + (1-\mu)\boldsymbol{g}$;

  $\boldsymbol{\xi'} \leftarrow \mu\boldsymbol{\xi} + (1-\mu)\boldsymbol{S}$;

  $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi'}/(1-\mu^{t+1})$;

  $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi'}/(1-\mu^{t+1})$;

  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\boldsymbol{g}$;

  for $x \in [1, ..., d]$ **do**

    $s_x \leftarrow \left\lceil \frac{2L\alpha}{2-L\alpha} \frac{\xi_x \sum_x \xi_x}{||\boldsymbol{\chi}||^2} \right\rceil$

  $t \leftarrow t + 1$

here can be directly applied. This update rule guarantees fast convergence in expectation to the optimal value of sufficiently smooth loss functions. The underlying assumption of this result is that the loss function is strongly convex and has Lipschitz-continuous gradients. In most realistic QML scenarios the optimization landscape is not convex, however, if the optimizer were to settle into a convex region then fast convergence is expected.

The exact assumptions needed in order to ensure the geometric convergence of Refoqus to the global minima are as follows:

1. $\mathbb{E}[g_x(\boldsymbol{\theta})] = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x}, \forall x \in [d]$.

2. $\text{Var}[g_x(\boldsymbol{\theta})] = \frac{\text{Var}[X_x]}{s_x}$, where X_x is the sampling-based estimator of $g_x(\boldsymbol{\theta})$.

3. $\mathcal{L}(\boldsymbol{\theta})$ is μ-strongly convex.

4. $\mathcal{L}(\boldsymbol{\theta})$ has L-Lipschitz continuous gradient.

5. α is a constant learning rate satisfying $0 < \alpha < \min\{1/L, 2/\mu\}$.

6. An ideal version of the update rule holds, that is:

$$s_x = \frac{2L\alpha}{2 - L\alpha} \frac{\sigma_x(\sum_{x'=1}^{d} \sigma_{x'})}{||\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta})||^2} \quad \forall x \in [d],$$
$$\text{where} \quad \sigma_x = \sqrt{\text{Var}[X_x]}\,.$$

In the previous sections, we have shown how to construct unbiased estimators for the gradient, satisfying the first assumption. The second assumption is satisfied as the estimate of the gradient is constructed by calculating the mean of several sampling-based estimates. Assumption 5 is satisfied by construction. Assumption 6 is an idealized version of the update rule used in Refoqus. However, the gradient magnitude and estimator variances cannot be exactly known in general, so these quantities are replaced by exponentially moving averages to predict the values of σ_x and $||\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta})||^2$. Assumption 4 is also satisfied for all the loss functions we consider in this work. Finally, as previously noted assumption 5 is not expected to hold in QML landscapes, which are non-convex in general. Nevertheless, the convergence guarantee provides strong analytical motivation for the functionality of the optimizer in an idealized scenario.

8.7 Numerical Results

We benchmark the performance of several optimizers when applied to using VQSE [41] for quantum PCA [119] on molecular ground states. We perform quantum PCA on 3 quantum datasets: ground states of the H_2 molecule in the sto-3g basis (4 qubits), H_2 in the 6-31g basis (8 qubits) and BeH_2 in the sto-3g basis (14 qubits). We use 101 circuits, per dataset. The corresponding covariance matrix [70], which can be expressed as $\frac{1}{101}\sum_{i=0}^{100} |\psi\rangle_i \langle\psi|_i$. We optimize using the local cost introduced in Eq. (8.30) and repeated here for convenience:

$$H = \mathbb{1} - \sum_{j=1}^{n} r_j Z_j, \quad r_j \in \mathbb{R} \tag{8.66}$$

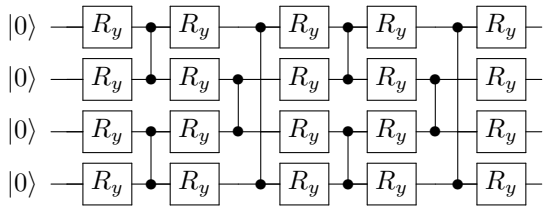


Figure 8.4: **Hardware-efficient ansatz with 2 layers used for VQSE on 4 qubits.** Each R_y rotation is independently parametrized according to $R_y(\theta) = e^{-iY\theta/2}$.

taking coefficients $r_j = 1.0 + 0.2(j-1)$ and $p_i = \frac{1}{101}$ following the presentation in [41]. Hence the latter coefficients form the $q_{i,j} = p_i r_j$ terms used for shot-allocation strategies, as outlined below.

When implementing Rosalin and Refoqus we use the weighted random sampling strategy for allocating shots as it demonstrated superior performance against other strategies in the numerical results of [10]. Hence, the $q_{i,j}$ coefficients that appear in the loss function (and the gradient estimators) are used to construct a multinomial probability distribution defined by the terms $\frac{|q_{i,j}|}{M}$, where $M = \sum_{i,j} |q_{i,j}|$. This distribution is used to probabilistically allocate the number of shots given to each term for each gradient estimation.

We use a circuit consisting of 2 layers of a hardware-efficient ansatz with depth 4, depicted in Fig. 8.4, with 20, 40, and 70 parameters for the 4, 8 and 16 qubit problems respectively. These parameters are optimized in order to minimize the VQSE cost function using the Adam, Rosalin, and Refoqus optimizers[2]. We use the absolute error in estimating the eigenvalues of the system to benchmark the overall performance of the output of the optimized circuit as this is desired output of the algorithm. Given the exact 16 highest eigenvalues λ_i, and their estimation $\tilde{\lambda}_i$ given current parameters θ, the latter is computed as $\epsilon_\lambda = \sum_{i=1}^{16} (\lambda_i - \tilde{\lambda}_i)^2$.

Figure 8.5 shows the results obtained when running each algorithm 20 times with different random initialization of the variational ansatz, up to a total shot budget of 10^8. In general, we see that Refoqus is the best-performing optimizer, achieving a best-case accuracy while using fewer shots for all shot budgets. In summary, for the 4 qubit system, a median eigenvalue error of 6.8×10^{-6}, 5.15×10^{-6} and 6.08×10^{-7}

[2]For Adam, we perform 100 shots per circuit in our simulations. For Rosalin and Refoqus, we use a minimum of 2 shots per circuit (this leads to $s_{\min} = 202$ and $s_{\min} = 2$ for Rosalin and Refoqus respectively) in conjunction with WRS.

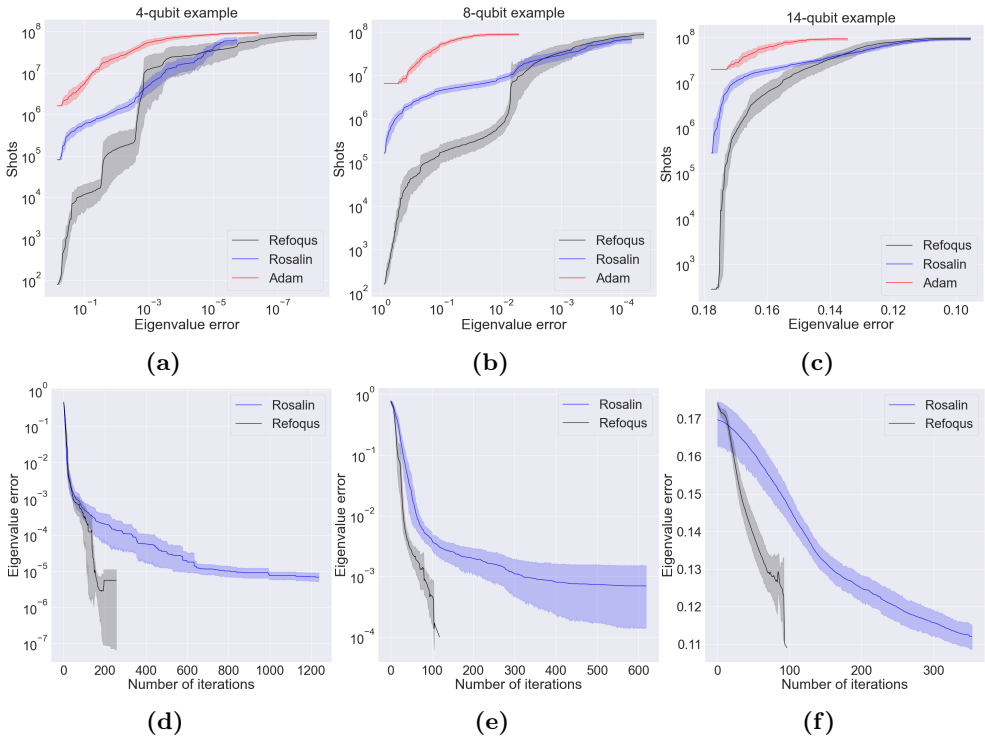


Figure 8.5: **VQSE for quantum PCA of H_2 molecular ground states (a, b, d, and e) and BeH_2 (c and f)**. The upper plots show the budget of shots spent against the best eigenvalue error achieved by the optimizers. The lower plots show the number of iterations used to spend the total shot budget. We display the results obtained from 20 independent optimization runs on a data set of 101 circuits representing molecular ground states calculated using the Adam, Rosalin, and Refoqus optimizers and compare their performance. We show the median (solid lines) and 95% confidence intervals (shaded regions) over the 20 different random initializations.

was obtained using Adam, Rosalin and Refoqus respectively. Although we found better minimal value with Adam compared to Rosalin (3.79×10^{-7} and 1.74×10^{-6} respectively), Rosalin is more advantageous at lower shot budgets. However, Refoqus achieved a minimal error value of 6.13×10^{-9} and demonstrates a clear advantage over both Adam and Rosalin. On both the 8 and 14 qubit systems, we observe a similar trend although eigenvalue errors are worse overall. This is due to the variational ansatz being kept at a fixed depth while increasing the size of the problem, which leads to worse performance. Nonetheless, Refoqus appears to clearly match or outperform both Adam and Rosalin in the number of shots required to reach a given accuracy.

Additionally, Refoqus requires fewer parameter updates (iterations) when compared to Rosalin. Indeed, for the 4, 8 and 14 qubits problems respectively, a median of 117, 89, and 80 iterations were used by Refoqus against 1217, 618, and 353 for Rosalin. We note that this may be interpreted as another desirable feature of the optimizer, as this minimizes the number of iterations needed in the quantum-classical feedback loop to arrive at a solution of the same (or better) quality. Although the number of iterations is not a bottleneck in and of itself, in current hardware the quantum-classical feedback can prove restrictive meaning fewer iterations are favorable for real-device implementation.

Finally, we note that the variance of the Refoqus results appears to be larger, suggesting that sampling over more terms can lead to a larger variety in the quality of the optimization obtained. Nevertheless, the advantage in shot frugality that arises from sampling over more terms is clear and despite this larger variance, Refoqus is clearly the best-performing optimizer.

8.8 Discussion

QML algorithms present a different paradigm for data processing which is particularly well suited to quantum data. VQAs are a key contender in giving a near-term useful quantum advantage. However, both VQAs and QML models require long run times and large resource overheads during training (as many iterations and shots to achieve respectable performances). To address these challenges, we propose Refoqus as a shot-frugal gradient-based optimizer based on state and operator sampling. We outline many cost functions that are of interest to the community and are easily captured within our framework for Refoqus. The new optimizer leverages the loss function of the problem to allocate shots randomly when evaluating gradients. This randomness allows us to make resource-cheap gradient update steps, unlocking shot-frugality. We have shown that Refoqus comes with geometric convergence guarantees under specific assumptions. Additionally, when applying our optimizer to a QML task, namely a quantum PCA task, we obtain significantly better performance in terms of the number of shots and the number of iterations needed to obtain a given accuracy.

A potential future research direction is to extend our analysis to more complicated, non-linear loss functions such as the log-likelihood, exploring in more detail how to introduce shot frugality to gradient-free optimizers. Furthermore, applying Refoqus to a problem of interest on a real device is an interesting next step.

Chapter 9

Conclusions and outlook

In this chapter, we conclude the work done in this thesis, by presenting the research direction, and our results related to this direction on VQAs. We also discuss future research opportunities.

As we have seen throughout the previous chapters in this thesis, VQAs are evolving hybrid quantum-classical algorithms most often used as heuristics and designed to tackle relevant applications for industries in the NISQ era. Such hybrid algorithms can be complex with many components or hyperparameters to work with. They will also have to be compared with classical counterparts. Hence, they will face the problems of *algorithm selection and configuration*, and they will go through empirical studies and domain-specific enhancements. In this thesis, we demonstrated these problems on VQAs for combinatorial optimization, chemistry, and machine learning applications. We demonstrated many benchmarking techniques useful for the design of VQAs and their usage. Next, we will refer back to our research questions and restate the results obtained in this thesis.

1. Our first research question **RQ1** is about understanding when a hybrid quantum-classical algorithm can be used against a classical counterpart. In Chapter 3, we introduce the principle of algorithm selection to quantum optimization algorithms. The algorithm selection problem boils down to the design of a classification algorithm, which can efficiently detect whether a given problem instance satisfies one criterion or many criteria to be used against other algorithms. We demonstrated a case of algorithm selection with QAOA against a classical counterpart and proved the decision to be NP-hard. The algorithm selection methodology can be extended to many other settings where a quantum algorithm will

be considered against a classical counterpart.

2. Our second research question **RQ2** is about identifying the key internal components of hybrid quantum-classical algorithms and how to set them well. In chapters 4, 6 and 8, we addressed the problem of algorithm configuration, specifically to the parameter-setting algorithms (such as classical optimizers), key components of VQAs as they affect greatly their results. To determine the well-performing ones, we use benchmarking methodologies. In each chapter, we tackled this problem in a different context. In Chapter 6, existing optimizers, SPSA and CMA-ES, were studied for VQE settings on several chemistry and material science problems. We obtained comparable performances between them, although CMA-ES obtains better results on the more challenging and interesting systems. On the contrary, a new optimizer was designed in Chapter 8 in the context of QML applications. Its design was guided by using many theoretical concepts to obtain practical improvements over other previous state-of-art QML optimizers. In Chapter 4, we benchmarked unsupervised techniques for setting the parameters of QAOA circuits relying on the concentration property as an argument to save numerous quantum circuit calls that would be used by a classical optimizer. We empirically showed that using instance encodings (by computing features or computing them with a model) for angle-setting strategies yields better results than using angle values only.

3. The principles of algorithm selection and configuration were also demonstrated with a new hybrid algorithm that can help solve problems considering (some of the) limitations of real devices. In Chapter 5, we demonstrated a combination of tabu search with QAOA used as a neighborhood sampler and our results demonstrate potential in solving large problems with limited quantum resources. This also opens up the topics of sampling and multiobjective aspects of quantum algorithms that allow balancing between exploration and exploitation, a very important concept in search algorithms. We believe our approach combined with benchmarking analyses will provide new promising ways to maximize the use of limited near-term quantum computing architectures for real-world and industrial optimization problems.

4. VQAs may come with many possible hyperparameters (such as the number of layers, the circuit architecture, and the used optimizer), adding complexity to their usage in practice. Analyzing which ones matter or not for a given domain can help reduce the usage of (potentially expensive) quantum resources. This

research question, as a part of **RQ2**, was discussed in Chapter 7 through a study done on quantum neural networks and classical datasets. Using functional ANOVA, a hyperparameter importance framework, we distinguished three main levels of importance. On the one hand, Adam's learning rate, depth, and data encoding strategy are deemed very important, as we expected. On the other hand, the less considered hyperparameters such as the particular choice of the entangling gate and using 3 rotation types in the variational layer are in the least important group. We envision such studies with techniques such as functional ANOVA to be employed in future works related to quantum machine learning and understanding how to apply quantum models in practice.

The methodologies used in our studies are agnostic to the considered VQA and the settings upon which a VQA is run. Our goal during research was to show by examples the benefits of such methodologies for algorithm selection and configuration in the context of NISQ algorithms and for designing hybrid quantum-classical algorithms on many domains of applications relevant to industries. The methodologies can be extended to noisy settings, with other noise-mitigation techniques to run VQAs better and faster on real quantum devices. New VQA algorithms that will be designed in the future will also benefit from such benchmarking studies. As quantum hardware keeps improving, with hybrid clusters of many types of hardware, and of course, many possibilities to run VQAs with many targeted applications, the algorithm selection and configuration problems become even more relevant. We hope the work reported in this thesis will help in applying standard and normalized practices for designing better hybrid quantum-classical workflows tailored to many potential applications of quantum computing. In particular, the applications of VQAs related to the energy sector will benefit from such studies as demonstrated for combinatorial optimization, chemistry, and machine learning. Indeed, tailoring and extending these algorithms and studies for other problems and data from the energy sector will be a logical next step for future research to continue exploring the potential value of quantum algorithms in this sector.

Bibliography

[1] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nature Computational Science*, 1(6):403–409, 2021.

[2] Amir Adler, Mauricio Araya-Polo, and Tomaso Poggio. Deep learning for seismic inverse problems: Toward the acceleration of geophysical analysis workflows. *IEEE Signal Processing Magazine*, 38(2):89–119, 2021.

[3] D. Akhiyarov, A. Gherbi, B. Conche, and M. Araya-Polo. Fast machine learning-based oil slick detection on satellite imaging. *82nd EAGE Annual Conference & Exhibition*, 2021(1):1–5, 2021.

[4] V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte. Parameter concentrations in quantum approximate optimization. *Phys. Rev. A*, 104:L010401, Jul 2021.

[5] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. Analysis of Quantum Approximate Optimization Algorithm under Realistic Noise in Superconducting Qubits, 2019. arXiv:1907.09631.

[6] MD SAJID ANIS and et al. Qiskit: An open-source framework for quantum computing, 2021.

[7] Eric R Anschuetz and Bobak T Kiani. Beyond barren plateaus: Quantum variational algorithms are swamped with traps. *arXiv preprint arXiv:2205.05786*, 2022.

[8] Simon Apers and Ronald de Wolf. Quantum Speedup for Graph Sparsification, Cut Approximation and Laplacian Solving, 2019.

[9] Andrew Arrasmith, M. Cerezo, Piotr Czarnik, Lukasz Cincio, and Patrick J Coles. Effect of barren plateaus on gradient-free optimization. *Quantum*, 5:558, 2021.

[10] Andrew Arrasmith, Lukasz Cincio, Rolando D Somma, and Patrick J Coles. Operator sampling for shot-frugal optimization in variational algorithms. *arXiv preprint arXiv:2004.06252*, 2020.

[11] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, and et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, Oct 2019.

[12] Thomas Bäck. *Evolutionary algorithms in theory and practice - evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, 1996.

[13] Lukas Balles, Javier Romero, and Philipp Hennig. Coupling Adaptive Batch Sizes with Learning Rates. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 410–419, 2017.

[14] Panagiotis Kl. Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving Variational Quantum Optimization using CVaR, 2019. arXiv:1907.04769.

[15] J. E. Beasley. Or-library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[16] John Beasley. QUBO instances link - file bqpgka.txt. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/bqpinfo.html.

[17] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J. Osborne, Robert Salzmann, Daniel Scheiermann, and Ramona Wolf. Training deep quantum neural networks. *Nature Communications*, 11(1):808, 2020.

[18] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 2019.

[19] Ville Bergholm, Josh A. Izaac, Maria Schuld, Christian Gogolin, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *CoRR*, abs/1811.04968, 2018.

[20] Hans-Georg Beyer. *The theory of evolution strategies*. Natural computing series. Springer, 2001.

[21] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[22] Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[23] Lennart Bittel and Martin Kliesch. Training variational quantum algorithms is np-hard. *Phys. Rev. Lett.*, 127:120502, Sep 2021.

[24] Xavier Bonet-Monroig, Hao Wang, Diederick Vermetten, Bruno Senjean, Charles Moussa, Thomas Bäck, Vedran Dunjko, and Thomas E. O'Brien. Performance comparison of optimization methods on variational quantum algorithms. *Phys. Rev. A*, 107:032407, Mar 2023.

[25] Michael Booth and Steven P. Reinhardt. Partitioning optimization problems for hybrid classical / quantum execution technical report, 2017.

[26] Fernando G. S. L. Brandão, Michael Broughton, Edward Farhi, Sam Gutmann, and Hartmut Neven. For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances, 2018. arXiv:1812.04170.

[27] Sergey Bravyi, David Gosset, and Robert König. Quantum advantage with shallow circuits. *Science*, 362(6412):308–311, 2018.

[28] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to State Preparation and Variational Optimization from Symmetry Protection, 2019. arXiv:1910.08980.

[29] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to state preparation and variational optimization from symmetry protection, 2019.

[30] Sergey Bravyi, Graeme Smith, and John A. Smolin. Trading classical and quantum computational resources. *Phys. Rev. X*, 6:021043, Jun 2016.

[31] Pavel Brazdil, Jan N. van Rijn, Carlos Soares, and Joaquin Vanschoren. *Metalearning: Applications to Automated Machine Learning and Data Mining*. Springer, 2nd edition, 2022.

[32] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[33] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124, Aug 2010.

[34] Michael Broughton and et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv:2003.02989*, 2020.

[35] C. G. BROYDEN. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

[36] Chris Cade, Lana Mineh, Ashley Montanaro, and Stasja Stanisic. Strategies for solving the fermi-hubbard model on near-term quantum computers. *Phys. Rev. B*, 102:235122, Dec 2020.

[37] Adam Callison and Nicholas Chancellor. Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond. *Phys. Rev. A*, 106:010101, Jul 2022.

[38] Matthias C. Caro, Elies Gil-Fuster, Johannes Jakob Meyer, Jens Eisert, and Ryan Sweke. Encoding-dependent generalization bounds for parametrized quantum circuits. *Quantum*, 5:582, 2021.

[39] Matthias C. Caro, Hsin-Yuan Huang, Nicholas Ezzell, Joe Gibbs, Andrew T. Sornborger, Lukasz Cincio, Patrick J. Coles, and Zoe Holmes. Out-of-distribution generalization for learning quantum dynamics. *arXiv preprint arXiv:2204.10268*, 2022.

[40] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, Sep 2021.

[41] M. Cerezo, Kunal Sharma, Andrew Arrasmith, and Patrick J Coles. Variational quantum state eigensolver. *npj Quantum Information*, 8(1):1–11, 2022.

[42] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1):1–12, 2021.

[43] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *arXiv:1603.02754*, 2016.

[44] Cristina Cirstoiu, Zoe Holmes, Joseph Iosue, Lukasz Cincio, Patrick J. Coles, and Andrew Sornborger. Variational fast forwarding for quantum simulation beyond the coherence time. *npj Quantum Information*, 6(1):1–10, 2020.

[45] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[46] Gavin E Crooks. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv preprint arXiv:1811.08419*, 2018.

[47] Benjamin Doerr and Carola Doerr. Optimal static and self-adjusting parameter choices for the $(1+(\lambda, \lambda))$ genetic algorithm. *Algorithmica*, 80(5):1658–1709, 2018.

[48] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In Peter A. N. Bosman, editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017*, pages 777–784. ACM, 2017.

[49] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. Ioh-profiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv e-prints:1810.05281*, October 2018.

[50] Vedran Dunjko, Yimin Ge, and J. Ignacio Cirac. Computational speedups using small quantum devices. *Phys. Rev. Lett.*, 121:250501, Dec 2018.

[51] Iain Dunning, Swati Gupta, and John Silberholz. What Works Best When? A Systematic Evaluation of Heuristics for Max-cut and QUBO. *INFORMS Journal on Computing*, 30(3):608–624, 2018.

[52] Iain Dunning, Swati Gupta, and John Silberholz. What works best when? a systematic evaluation of heuristics for max-cut and qubo. *INFORMS J. on Computing*, 30(3):608–624, aug 2018.

[53] Katharina Eggensperger, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1114–1120. AAAI Press, 2015.

[54] Gerald Kelechi Ekechukwu, Romain de Loubens, and Mauricio Araya-Polo. Long short-term memory-driven forecast of CO_2 injection in porous media. *Physics of Fluids*, 34(5):056606, May 2022.

[55] Suguru Endo, Zhenyu Cai, Simon C. Benjamin, and Xiao Yuan. Hybrid quantum-classical algorithms and quantum error mitigation. *Journal of the Physical Society of Japan*, 90(3):032001, 2021.

[56] Francesco A Evangelista, Garnet Kin-Lic Chan, and Gustavo E Scuseria. Exact parameterization of fermionic wave functions via unitary coupled cluster theory. *J. Chem. Phys.*, 151(24):244112, 2019.

[57] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014. arXiv:1411.4028.

[58] Edward Farhi and Aram W Harrow. Quantum supremacy through the quantum approximate optimization algorithm, 2016.

[59] Jan Faye. Copenhagen Interpretation of Quantum Mechanics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2019 edition, 2019.

[60] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: Hands-free automl via meta-learning. *arXiv:2007.04074v2 [cs.LG]*, 2021.

[61] Alexey Galda, Xiaoyuan Liu, Danylo Lykov, Yuri Alexeev, and Ilya Safro. Transferability of optimal qaoa parameters between random graphs, 2021.

[62] Yimin Ge and Vedran Dunjko. A hybrid algorithm framework for small quantum computers with application to finding hamiltonian cycles. *Journal of Mathematical Physics*, 61(1):12201, Jan 2020.

[63] Joe Gibbs, Zoe Holmes, Matthias C. Caro, Nicholas Ezzell, Hsin-Yuan Huang, Lukasz Cincio, Andrew T. Sornborger, and Patrick J. Coles. Dynamical simulation via quantum machine learning with provable generalization. *arXiv preprint arXiv:2204.10269*, 2022.

[64] Fred Glover and Jin-Kao Hao. Efficient evaluations for solving large 0-1 unconstrained quadratic optimisation problems. *Int. J. Metaheuristics*, 1, 01 2010.

[65] Fred Glover, Gary Kochenberger, and Bahram Alidaee. Adaptive memory tabu search for binary quadratic programs. *Management Science*, 44:336–345, 03 1998.

[66] Fred W. Glover. *Tabu Search*, pages 1537–1544. Springer US, Boston, MA, 2013.

[67] Fred W. Glover, Zhipeng Lü, and Jin-Kao Hao. Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR*, 8:239–253, 2010.

[68] Michel X. Goemans. Combining approximation algorithms for the prize-collecting tsp. *ArXiv*, abs/0910.0553, 2009.

[69] Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42(6):1115–1145, November 1995.

[70] Max Hunter Gordon, M. Cerezo, Lukasz Cincio, and Patrick J. Coles. Covariance matrix preparation for quantum principal component analysis. *PRX Quantum*, 3:030334, Sep 2022.

[71] Gabriel Greene-Diniz, David Zsolt Manrique, Wassil Sennane, Yann Magnin, Elvira Shishenina, Philippe Cordier, Philip Llewellyn, Michal Krompiec, Marko J. Rančić, and David Muñoz Ramo. Modelling carbon capture on metal-organic frameworks with quantum computing, 2022.

[72] Andi Gu, Angus Lowe, Pavel A Dub, Patrick J. Coles, and Andrew Arrasmith. Adaptive shot allocation for fast convergence in variational quantum algorithms. *arXiv preprint arXiv:2108.10434*, 2021.

[73] G. G. Guerreschi and A. Y. Matsuura. QAOA for Max-cut requires hundreds of qubits for quantum speed-up. *Scientific Reports*, 9(1):6903, 2019.

[74] Gian Giacomo Guerreschi. Solving quadratic unconstrained binary optimization with divide-and-conquer and quantum algorithms, 2021.

[75] Nikolaus Hansen. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In *ACM-GECCO Genetic and Evolutionary Computation Conference*, Montreal, Canada, July 2009.

[76] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.

[77] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Posík. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In Martin Pelikan and Jürgen Branke, editors, *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7-11, 2010, Companion Material*, pages 1689–1696. ACM, 2010.

[78] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001.

[79] Matthew P. Harrigan, Kevin J. Sung, Matthew Neeley, Kevin J. Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C. Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B. Buckley, David A. Buell, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Sean Demura, Andrew Dunsworth, Daniel Eppens, Austin Fowler, Brooks Foxen, Craig Gidney, Marissa Giustina, Rob Graff, Steve Habegger, Alan Ho, Sabrina Hong, Trent Huang, L. B. Ioffe, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Cody Jones, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Seon Kim, Paul V. Klimov, Alexander N. Korotkov, Fedor Kostritsa, David Landhuis, Pavel Laptev, Mike Lindmark, Martin Leib, Orion Martin, John M. Martinis, Jarrod R. McClean, Matt McEwen, Anthony Megrant, Xiao Mi, Masoud Mohseni, Wojciech Mruczkiewicz, Josh Mutus, Ofer Naaman, Charles Neill, Florian Neukart, Murphy Yuezhen Niu, Thomas E. O'Brien, Bryan O'Gorman, Eric Ostby, Andre Petukhov, Harald Putterman, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Andrea Skolik, Vadim Smelyanskiy, Doug Strain, Michael Streif, Marco Szalay, Amit Vainsencher, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Leo Zhou, Hartmut Neven, Dave Bacon, Erik Lucero, Edward Farhi, and Ryan Babbush. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nature Physics*, 17(3):332–336, Mar 2021.

[80] Johan Håstad. Some Optimal Inapproximability Results. *J. ACM*, 48(4):798–859, July 2001.

[81] M. B. Hastings. Classical and Quantum Bounded Depth Approximation Algorithms, 2019. arXiv:1905.07047.

[82] Tobias Haug, Chris N. Self, and M. S. Kim. Large-scale quantum machine learning. *CoRR*, abs/2108.01039, 2021.

[83] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[84] Dirk Heimann, Hans Hohenfeld, Felix Wiebe, and Frank Kirchner. Quantum deep reinforcement learning for robot navigation tasks. *CoRR*, abs/2202.12180, 2022.

[85] Hans Hinterberger. Exploratory data analysis. In *Encyclopedia of Database Systems*, pages 1080–1080. Springer US, Boston, MA, 2009.

[86] Wassily Hoeffding. A class of statistics with asymptotically normal distribution. *The Annals of Mathematical Statistics*, 19(3):293–325, 1948.

[87] Zoë Holmes, Andrew Arrasmith, Bin Yan, Patrick J. Coles, Andreas Albrecht, and Andrew T Sornborger. Barren plateaus preclude learning scramblers. *Physical Review Letters*, 126(19):190501, 2021.

[88] Zoë Holmes, Kunal Sharma, M. Cerezo, and Patrick J Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3:010313, Jan 2022.

[89] Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated Algorithm Selection and Configuration (Dagstuhl Seminar 16412). *Dagstuhl Reports*, 6(10):33–74, 2017.

[90] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R McClean. Power of data in quantum machine learning. *Nature Communications*, 12(1):1–9, 2021.

[91] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057, 2020.

[92] F. Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1130–1144, 2014.

[93] Sofiène Jerbi, Lukas J. Fiderer, Hendrik Poulsen Nautrup, Jonas M. Kübler, Hans J. Briegel, and Vedran Dunjko. Quantum machine learning beyond kernel methods. *CoRR*, abs/2110.13162, 2021.

[94] Sofiène Jerbi, Casper Gyurik, Simon Marshall, Hans J. Briegel, and Vedran Dunjko. Parametrized quantum policies for reinforcement learning. In *Advances in Neural Information Processing Systems 34*, pages 28362–28375, 2021.

[95] Peter D Johnson, Jonathan Romero, Jonathan Olson, Yudong Cao, and Alán Aspuru-Guzik. Qvector: an algorithm for device-tailored quantum error correction. *arXiv preprint arXiv:1711.02249*, 2017.

[96] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry Chow, and Jay Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549:242–246, 09 2017.

[97] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[98] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[99] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[100] Sami Khairy, Ruslan Shaydulin, Lukasz Cincio, Yuri Alexeev, and Prasanna Balaprakash. Learning to optimize variational quantum circuits to solve combinatorial problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2367–2375, April 2020.

[101] Subhash Khot. On the Power of Unique 2-prover 1-round Games. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 767–775, New York, NY, USA, 2002. ACM.

[102] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs? *SIAM J. Comput.*, 37(1):319–357, April 2007.

[103] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[104] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[105] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81, Jul 2014.

[106] Gary A. Kochenberger and Fred Glover. *A Unified Framework for Modeling and Solving Combinatorial Optimization Problems: A Tutorial*, pages 101–124. Springer US, Boston, MA, 2006.

[107] Bálint Koczor and Simon C Benjamin. Quantum natural gradient generalised to non-unitary circuits. *arXiv preprint arXiv:1912.08660*, 2019.

[108] Lars Kotthoff. *Algorithm Selection for Combinatorial Search Problems: A Survey*, pages 149–190. Springer International Publishing, Cham, 2016.

[109] Jonas M Kübler, Andrew Arrasmith, Lukasz Cincio, and Patrick J Coles. An adaptive optimizer for measurement-frugal variational algorithms. *Quantum*, 4:263, 2020.

[110] Ryan LaRose, Arkin Tikku, Étude O'Neel-Judy, Lukasz Cincio, and Patrick J Coles. Variational quantum state diagonalization. *npj Quantum Information*, 5(1):1–10, 2019.

[111] Xinwei Lee, Yoshiyuki Saito, Dongsheng Cai, and Nobuyoshi Asai. Parameters fixing strategy for quantum approximate optimization algorithm. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 10–16, 2021.

[112] Per Kristian Lehre and Xin Yao. Crossover can be constructive when computing unique input–output sequences. *Soft Computing*, 15(9):1675–1687, 2011.

[113] Junde Li, Mahabubul Alam, and Swaroop Ghosh. Large-scale quantum approximate optimization via divide-and-conquer, 2021.

[114] Li Li, Minjie Fan, Marc Coram, Patrick Riley, and Stefan Leichenauer. Quantum Optimization with a Novel Gibbs Objective Function and Ansatz Architecture Search, 2019. arXiv:1909.07621.

[115] Qing-Song Li, Huan-Yu Liu, Qingchun Wang, Yu-Chun Wu, and Guo-Ping Guo. A unified framework of transformations based on the jordan–wigner transformation. *The Journal of Chemical Physics*, 157(13):134104, October 2022.

[116] Jin-Guo Liu and Lei Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, 98:062324, 2018.

[117] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, 2021.

[118] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[119] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, September 2014. Number: 9 Publisher: Nature Publishing Group.

[120] Zhipeng Lü, Fred W. Glover, and Jin-Kao Hao. A hybrid metaheuristic approach to solving the ubqp problem. *Eur. J. Oper. Res.*, 207:1254–1262, 2010.

[121] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[122] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. Entanglement-induced barren plateaus. *PRX Quantum*, 2(4):040316, 2021.

[123] Simon C. Marshall, Casper Gyurik, and Vedran Dunjko. High dimensional quantum machine learning with small quantum computers. *CoRR*, abs/2203.13739, 2022.

[124] Andreas Mayer and Svein Ove Aas. andim/noisyopt, May 2017.

[125] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):1–6, 2018.

[126] Jarrod R McClean, Nicholas C Rubin, Kevin J Sung, Ian D Kivlichan, Xavier Bonet-Monroig, Yudong Cao, Chengyu Dai, E Schuyler Fried, Craig Gidney, Brendan Gimby, Pranav Gokhale, Thomas Häner, Tarini Hardikar, Vojtěch Havlíček, Oscar Higgott, Cupjin Huang, Josh Izaac, Zhang Jiang, Xinle Liu, Sam McArdle, Matthew Neeley, Thomas O'Brien, Bryan O'Gorman, Isil Ozfidan, Maxwell D Radin, Jhonathan Romero, Nicolas P D Sawaya, Bruno Senjean, Kanav Setia, Sukin Sim, Damian S Steiger, Mark Steudtner, Qiming Sun, Wei Sun, Daochen Wang, Fang Zhang, and Ryan Babbush. Openfermion: the electronic structure package for quantum computers. *Quantum Sci. Technol.*, 2020.

[127] Matija Medvidovic and Giuseppe Carleo. Classical variational simulation of the quantum approximate optimization algorithm, 2020.

[128] Alexey A. Melnikov, Leonid Fedichkin, and Alexander Alodjants. Predicting quantum advantage by quantum walk with convolutional neural networks. *New Journal of Physics*, 21, 2019.

[129] Stefano Mensa, Emre Sahin, Francesco Tacchino, Panagiotis Kl. Barkoutsos, and Ivano Tavernelli. Quantum machine learning framework for virtual screening in drug discovery: a prospective quantum advantage. *CoRR*, abs/2204.04017, 2022.

[130] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Physical Review A*, 98:032309, 2018.

[131] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[132] Tom M. Mitchell. *Machine learning, International Edition.* McGraw-Hill Series in Computer Science. McGraw-Hill, 1997.

[133] Felix Mohr and Jan N. van Rijn. Learning curves for decision making in supervised machine learning - A survey. *CoRR*, abs/2201.12150, 2022.

[134] Nikolaj Moll, Panagiotis Barkoutsos, Lev S. Bishop, Jerry M. Chow, Andrew Cross, Daniel J. Egger, Stefan Filipp, Andreas Fuhrer, Jay M. Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Müller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 2017. arXiv:1710.01022.

[135] Charles Moussa, Henri Calandra, and Vedran Dunjko. To quantum or not to quantum: towards algorithm selection in near-term quantum optimization. *Quantum Science and Technology*, 5(4):044009, 2020.

[136] Charles Moussa, Max Hunter Gordon, Michal Baczyk, M. Cerezo, Lukasz Cincio, and Patrick J. Coles. Resource frugal optimizer for quantum machine learning. *arXiv:2211.04965*, 2022.

[137] Charles Moussa, Jan N. van Rijn, Thomas Bäck, and Vedran Dunjko. Hyperparameter importance of quantum neural networks across small datasets. In Poncelet Pascal and Dino Ienco, editors, *Discovery Science*, pages 32–46, Cham, 2022. Springer Nature Switzerland.

[138] Charles Moussa, Hao Wang, Thomas Bäck, and Vedran Dunjko. Unsupervised strategies for identifying optimal parameters in quantum approximate optimization algorithm. *EPJ Quantum Technology*, 9(1), 2022.

[139] Charles Moussa, Hao Wang, Henri Calandra, Thomas Bäck, and Vedran Dunjko. Tabu-driven quantum neighborhood samplers. In Christine Zarges and Sébastien Verel, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 100–119, Cham, 2021. Springer International Publishing.

[140] Ken M Nakanishi, Keisuke Fujii, and Synge Todo. Sequential minimal optimization for quantum-classical hybrid algorithms. *Physical Review Research*, 2(4):043158, 2020.

[141] Quynh T. Nguyen, Louis Schatzki, Paolo Braccia, Michael Ragone, Martin Larocca, Frederic Sauvage, Patrick J. Coles, and M. Cerezo. A theory for equivariant quantum neural networks. *arXiv preprint arXiv:2210.08566*, 2022.

[142] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2000.

[143] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011.

[144] Michael A Nielsen et al. The fermionic canonical commutation relations and the jordan-wigner transform. *School of Physical Sciences The University of Queensland*, 59, 2005.

[145] niko, Youhei Akimoto, yoshihikoueno, Dimo Brockhoff, Matthew Chan, and ARF1. Cma-es/pycma: r3.0.3, April 2020.

[146] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 485–492, New York, NY, USA, 2016. ACM.

[147] Gintaras Palubeckis. Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. *Annals of Operations Research*, 131:259–282, 10 2004.

[148] Gintaras Palubeckis. Iterated tabu search for the unconstrained binary quadratic optimization problem. *Informatica (Vilnius)*, 2, 01 2006.

[149] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425 – 440, 1991.

[150] Tianyi Peng, Aram W. Harrow, Maris Ozols, and Xiaodi Wu. Simulating large quantum circuits on a small quantum computer. *Phys. Rev. Lett.*, 125:150504, Oct 2020.

[151] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.

[152] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):1–7, 2014.

[153] Arthur Pesah, M. Cerezo, Samson Wang, Tyler Volkoff, Andrew T Sornborger, and Patrick J Coles. Absence of barren plateaus in quantum convolutional neural networks. *Physical Review X*, 11(4):041011, 2021.

[154] Evan Peters, João Caldeira, Alan Ho, Stefan Leichenauer, Masoud Mohseni, Hartmut Neven, Panagiotis Spentzouris, Doug Strain, and Gabriel N. Perdue. Machine learning of high dimensional data on a noisy quantum processor. *npj Quantum Information*, 7(1):161, 2021.

[155] Svatopluk Poljak and Zsolt Tuza. Maximum cuts and large bipartite subgraphs. *Combinatorial optimization.(DIMACS series in discrete mathematics and theoretical computer science 20.)*, pages 181–244, 1995.

[156] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[157] Michael Ragone, Quynh T. Nguyen, Louis Schatzki, Paolo Braccia, Martin Larocca, Frederic Sauvage, Patrick J. Coles, and M. Cerezo. Representation theory for geometric quantum machine learning. *arXiv preprint arXiv:2210.07980*, 2022.

[158] H. Razip and M. N. Zakaria. Combining approximation algorithm with genetic algorithm at the initial population for np-complete problem. In *2017 IEEE 15th Student Conference on Research and Development (SCOReD)*, pages 98–103, 2017.

[159] Mathys Rennela, Sebastiaan Brand, Alfons Laarman, and Vedran Dunjko. Hybrid divide-and-conquer approach for tree search algorithms, 2020.

[160] Jonathan Romero, Ryan Babbush, Jarrod R McClean, Cornelius Hempel, Peter J Love, and Alán Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Science and Technology*, 4(1):014008, 2018.

[161] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2(4):045001, 2017.

[162] Gili Rosenberg, Mohammad Vazifeh, Brad Woods, and Eldad Haber. Building an iterative heuristic solver for a quantum annealer. *Computational Optimization and Applications*, 65:845–869, 2016.

[163] Manas Sajjan, Junxu Li, Raja Selvarajan, Shree Hari Sureshbabu, Sumit Suresh Kale, Rishabh Gupta, and Sabre Kais. Quantum computing enhanced machine learning for physico-chemical applications. *CoRR*, arXiv:2111.00851, 2021.

[164] Andrea Saltelli and I.M. Sobol. Sensitivity analysis for nonlinear mathematical models: Numerical experience. *Matematicheskoe Modelirovanie*, 7, 1995.

[165] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[166] Raffaele Santagati, Alan Aspuru-Guzik, Ryan Babbush, Matthias Degroote, Leticia Gonzalez, Elica Kyoseva, Nikolaj Moll, Markus Oppel, Robert M. Parrish, Nicholas C. Rubin, Michael Streif, Christofer S. Tautermann, Horst Weiss, Nathan Wiebe, and Clemens Utschig-Utschig. Drug design on quantum computers, 2023.

[167] Frederic Sauvage, Sukin Sim, Alexander A. Kunitsa, William A. Simon, Marta Mauri, and Alejandro Perdomo-Ortiz. Flip: A flexible initializer for arbitrarily-sized parametrized quantum circuits, 2021.

[168] N. Schetakis, D. Aghamalyan, M. Boguslavsky, and P. Griffin. Binary classifiers for noisy datasets: a comparative study of existing quantum machine learning frameworks and some new approaches. *CoRR*, abs/2111.03372, 2021.

[169] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[170] Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

[171] Maria Schuld and Nathan Killoran. Is quantum advantage the right goal for quantum machine learning? *Corr*, abs/2203.01340, 2022.

[172] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. The quest for a quantum neural network. *Quantum Information Processing*, 13(11):2567–2586, 2014.

[173] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[174] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103:032430, 2021.

[175] Abhinav Sharma, Jan N. van Rijn, Frank Hutter, and Andreas Müller. Hyperparameter importance for image classification by residual neural networks. In *Discovery Science - 22nd International Conference*, volume 11828 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 2019.

[176] Kunal Sharma, M. Cerezo, Lukasz Cincio, and Patrick J Coles. Trainability of dissipative perceptron-based quantum neural networks. *Physical Review Letters*, 128(18):180505, 2022.

[177] Ruslan Shaydulin, Ilya Safro, and Jeffrey Larson. Multistart Methods for Quantum Approximate Optimization, 2019. arXiv:1905.08768.

[178] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Ilya Safro, Susan Mniszewski, and Yuri Alexeev. Quantum Local Search for Graph Community Detection. In *APS March Meeting Abstracts*, volume 2019 of *APS Meeting Abstracts*, page C42.009, January 2019.

[179] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[180] Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko. Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *CoRR*, abs/2103.15084, 2021.

[181] I. M. Sobol. Sensitivity estimates for nonlinear mathematical models. *Mathematical Modelling and Computational Experiments*, 1(4):407–414, 1993.

[182] J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

[183] James C Spall. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins apl technical digest*, 19(4):482–492, 1998.

[184] Daniel Stilck França and Raul Garcia-Patron. Limitations of optimization algorithms on noisy quantum devices. *Nature Physics*, 17(11):1221–1227, 2021.

[185] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum natural gradient. *Quantum*, 4:269, 2020.

[186] Michael Streif and Martin Leib. Comparison of QAOA with Quantum and Simulated Annealing, 2019. arXiv:1901.01903.

[187] Michael Streif and Martin Leib. Training the quantum approximate optimization algorithm without access to a quantum processing unit, 2019.

[188] Ryan Sweke, Jean-Pierre Seifert, Dominik Hangleiter, and Jens Eisert. On the quantum versus classical learnability of discrete distributions. *Quantum*, 5:417, 2021.

[189] Ryan Sweke, Frederik Wilde, Johannes Jakob Meyer, Maria Schuld, Paul K Fährmann, Barthélémy Meynard-Piganeau, and Jens Eisert. Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum*, 4:314, 2020.

[190] AV Uvarov and Jacob D Biamonte. On barren plateaus and cost function locality in variational quantum algorithms. *Journal of Physics A: Mathematical and Theoretical*, 54(24):245301, 2021.

[191] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[192] Jan N. van Rijn and Frank Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, pages 2367–2376. ACM, 2018.

[193] Guillaume Verdon, Michael Broughton, Jarrod R. McClean, Kevin J. Sung, Ryan Babbush, Zhang Jiang, Hartmut Neven, and Masoud Mohseni. Learning to learn with quantum neural networks via classical neural networks, 2019. arXiv: 1907.05415.

[194] Hanrui Wang, Jiaqi Gu, Yongshan Ding, Zirui Li, Frederic T. Chong, David Z. Pan, and Song Han. Quantumnat: Quantum noise-aware training with noise injection, quantization and normalization. *CoRR*, abs/2110.11331, 2021.

[195] Hanrui Wang, Zirui Li, Jiaqi Gu, Yongshan Ding, David Z. Pan, and Song Han. Qoc: Quantum on-chip training with parameter shift and gradient pruning. *CoRR*, abs/2202.13239, 2022.

[196] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

[197] Samson Wang, Enrico Fontana, M. Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications*, 12(1):1–11, 2021.

[198] Yang Wang, Zhipeng Lü, Fred W. Glover, and Jin-Kao Hao. Path relinking for unconstrained binary quadratic programming. *Eur. J. Oper. Res.*, 223:595–604, 2012.

[199] Richard A. Watson and Thomas Jansen. A building-block royal road where crossover is provably essential. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'07)*, pages 1452–1459. ACM, 2007.

[200] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. Progress towards practical quantum variational algorithms. *Physical Review A*, 92:042303, Oct 2015.

[201] Madita Willsch, Dennis Willsch, Fengping Jin, Hans De Raedt, and Kristel Michielsen. Benchmarking the quantum approximate optimization algorithm. *Quantum Information Processing*, 19(7):197, Jun 2020.

[202] Leonard Wossnig. Quantum machine learning for classical data. *CoRR*, abs/2105.03684, 2021.

[203] Cheng Xue, Zhao-Yun Chen, Yu-Chun Wu, and Guo-Ping Guo. Effects of Quantum Noise on Quantum Approximate Optimization Algorithm, 2019. arXiv:1909.02196.

[204] Sheir Yarkoni, Elena Raponi, Thomas Bäck, and Sebastian Schmitt. Quantum annealing for industry applications: introduction and review. *Reports on Progress in Physics*, 85(10):104001, sep 2022.

[205] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications, 2018.

[206] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices, 2018. arXiv:1812.01041.

[207] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, 2018. arXiv:1812.01041.

[208] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):103, 2019.

Samenvatting

Quantumhardware brengt een nieuw paradigma en nieuwe manieren om problemen aan te pakken. Veel moeite moet worden gedaan om te begrijpen hoe deze technologie kan worden gebruikt, en wat deze technologie voor praktische voordelen in sectoren zoals combinatorische optimalisatie of machine learning geeft.

Variationele kwantumalgoritmen (VQA's) zijn geïntroduceerd als een manier om te werken binnen het bereik van de huidige imperfecte en instabiele quantumhardware. Gezien de imperfecte en onstabiele hardware, zijn variationele quantum algoritmen (VQA's) voorgesteld om toepassingen aan te pakken. Een VQA komt neer op een geparametriseerd quantum circuit, dat wil zeggen een quantumcircuit met instelbare reëele parameters. Deze parameters worden meestal aangepast door een klassiek optimalisatiealgoritme om een gewenste grootheid te optimaliseren.

Aangezien VQAs meestal als heuristiek worden gebruikt, is het niet duidelijk of zij werkelijk beter presteren dan de huidige klassieke algoritmen in relevante toepassingsgebieden. Bovendien kan een VQA vele componenten bevatten (ook wel hyperparameters genoemd) waardoor het een (groeiend) complex systeem wordt om de prestaties op vele taken te analyseren. Daarnaast worden we geconfronteerd met de problemen van algoritmeselectie en -configuratie, die we in dit proefschrift aanpakken aan de hand van vele voorbeelden die relevant zijn voor industriële toepassingen. In dit proefschrift, VQA's voor combinatorische optimalisatie, chemie/materiaalwetenschap en machinaal leren worden allen in overweging genomen.

De in onze studies gebruikte methoden zijn agnostisch voor de beschouwde VQA en de instellingen waarop een VQA wordt uitgevoerd. Ons doel tijdens het onderzoek was om aan de hand van voorbeelden de voordelen van dergelijke methodologieën voor algoritmeselectie en -configuratie in de context van NISQ-algoritmen en voor het ontwerpen van hybride quantum-klassieke algoritmen op vele domeinen van toepassingen die relevant zijn voor de industrie.

Samenvatting

Aangezien quantumhardware steeds beter wordt, met hybride clusters van vele typen hardware, en de vele mogelijkheden om VQA's te draaien met verscheidene doelgerichte toepassingen, wordt het succesvol uitvoeren van een VQA nog relevanter en ingewikkelder. Wij hopen dat het in dit proefschrift gerapporteerde werk zal helpen bij de toepassing van standaard- en genormaliseerde praktijken voor het ontwerpen van betere hybride quantum-klassieke workflows op maat gemaakt voor vele potentiële toepassingen van quantum computing.

Summary

Quantum hardware comes with a different computing paradigm and new ways to tackle applications. Much effort has to be put into understanding how to leverage this technology to give real-world advantages in areas of interest for industries such as combinatorial optimization or machine learning.

Variational quantum algorithms (VQAs) have been introduced as a way to work within the scope of reach of the current imperfect and unstable hardware. A VQA boils down to a parameterized quantum circuit, which is a quantum circuit with adjustable real-valued parameters. These parameters are generally tweaked by a classical optimization algorithm to optimize a quantity of interest.

As VQAs are most often used as heuristics, it is not clear whether they genuinely outperform current classical state-of-the-art algorithms in relevant domains of application. Plus, a VQA can come with many components (which can also be called hyperparameters) making it a (growing) complex system to analyze performances on many considered tasks. Consequently, we will face the issues of algorithm selection and configuration, which we tackle through this thesis on many examples relevant to industrial applications. In this thesis, VQAs for combinatorial optimization, chemistry/material science, and machine learning problems were considered.

The methodologies used in our studies are agnostic to the considered VQA and the settings upon which a VQA is run. Our goal during research was to demonstrate by examples the benefits of such methodologies for algorithm selection and configuration in the context of NISQ algorithms and for designing hybrid quantum-classical algorithms on many domains of applications relevant to industries.

As quantum hardware keeps improving, with hybrid clusters of many types of hardware, and of course many possibilities to run VQAs with many targeted applications, how to run successfully a VQA becomes even more relevant and more complicated. We hope the work reported in this thesis will help in applying standard and normalized

practices for designing better hybrid quantum-classical workflows tailored to many potential applications of quantum computing.

Acknowledgements

Firstly, I would like to express my gratitude towards my supervisors, Vedran Dunjko and Thomas Bäck, for their support and guidance all these years. Vedran, I am really grateful for helping me grow, for the patience you showed towards me, and for allowing me such freedom in the Ph.D. time. I would also like to thank, for their support, all of my colleagues and friends at LIACS and Leiden University with whom I shared this journey. I will surely miss our walks, our discussions, Ph.D. activities, and the drinks at the many bars in Leiden.

I would like to give thanks to my colleagues and friends at Leiden University who spent quite some time with me either in discussions, playing beach volleyball, or planning Ph.D. activities: Jan van Rijn, Hao Wang, Furong Ye, Koen van der Blom, Sander van Rijn, Theodoris Georgiou, Yash Patel, Thomas Moerland, Can Wang, Daniella Gawhens, Adrián Pérez Salinas, Lea Trendwalker, Casper Guyrik, David Dechant, Marios Kefalas, Diederick Vermetten, Shuaiqun Pan, Roy de Winter, Jacob de Noel, Lieuwe Vinkhuijzen, Sebastiaan Brand, Alexandra Blank, Marie Anastacio, Andrea Skolik, Katie Saentaweesook, Jackie Ashkin, Marina Gavryushkina, Simon Leyberger, Mario Cangiano, Melissa Thaler, Johannes Schimming, Hugo Proenca, Matthias Köning, Leni Rüland, Kaela Slavik $\cdots$ and all the other people that know me and I could not cite here.

Furthermore, I would like to thank my friends I met before the Ph.D. with whom I shared friendly and working moments during my studies and the start of my quantum journey: Aurelien Citrain, Jeremy Marchand, Vinduja Vasanthan, Henri Calandra, Adrien Suau, Elvira Shishenina, and Gabriel Staffelbach. Additionally, I would like to thank everyone I worked with during my exchanges at Los Alamos and OakRidge National Laboratory, for supporting me when investigating quantum computing.

Finally, I would like to express my gratitude towards my family, my father Issa, my mother May, my 3 brothers Mickael, Simon, and Thomas, as well as too many to cite

other family members such as cousins, aunts, uncles and my passed away grandparents, for their immeasurable support all these years. Without you, I would never have come this far in my life. Last but not least, I would like to express my love toward Solomiia for the tenderness and support you demonstrated to me. I hope everyone will keep safe and find happiness through their journeys.

About the author

Charles Moussa was born in Saint-Claude, Guadeloupe on the 30th of August, 1993, and grew up in the village of Trois-Rivieres. He studied Mathematical Enginnering at the National Institute of Applied Sciences (INSA) of Rouen. Interested in new technologies, he enrolled in an 18-month exchange contract with TotalEnergies and investigated quantum computing and the potential industrial applications at OakRidge National Laboratory, Tennessee. He started his Ph.D. in June 2019 on investigating further quantum algorithms towards industrial applications in the Quantum Computing and Natural Computing groups at LIACS, under the supervision of Dr. Vedran Dunjko and Prof.dr. Thomas Bäck. During his Ph.D., he also got enrolled at the Quantum Computing Summer School organized by the quantum group at Los Alamos National Laboratory, New Mexico, for a period of 10 weeks. His research interests lie in algorithms, machine learning, and quantum computing.