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

Moussa, C.

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

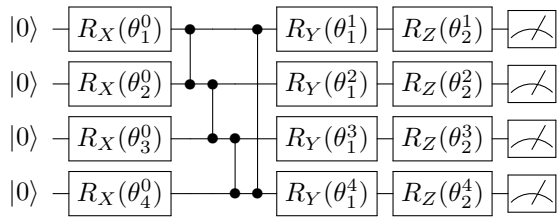

Figure 2.1: An example of quantum circuit with 4 qubits. Rotations are applied with arbitrary angles denoted by θ symbols. A layer of R_X rotations is first applied, followed by controlled-Z gates and R_Y, R_Z rotations. Finally, measurements are carried out to get a bitstring as an outcome.

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.

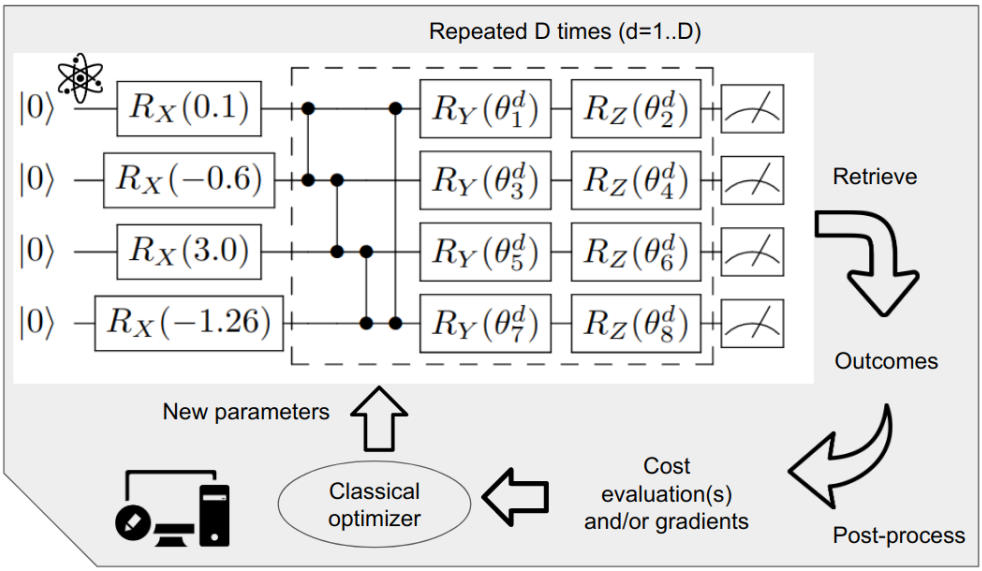

Figure 2.2: Diagram of a variational quantum algorithm. The user submits the algorithm where a quantum computer and a classical one work together. The user defines a quantum circuit with a set of parameters θ_f^d, $f = 1 \cdots 8$, $d = 1 \cdots D$. The dashed part corresponds to repeating the quantum 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.

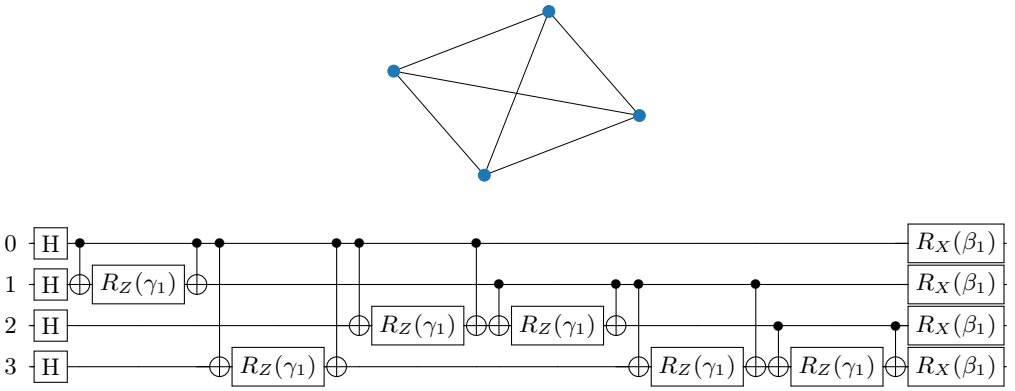

Figure 2.3: An example of QAOA circuit for MaxCut on a graph with 4 nodes. Here $p = 1$, corresponding to 2 parameters γ_1, β_1. The register of qubits is initialized as the $|0\rangle^{\otimes N}$ state. The first operation, the Hadamard transform, will prepare the state $|+\rangle^{\otimes N}$. Then 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.

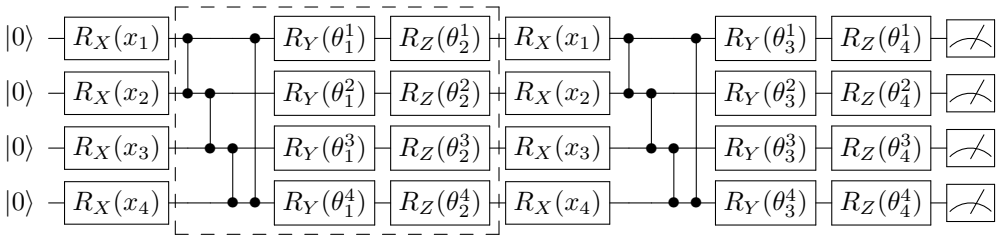

Figure 2.4: Parameterized quantum circuit architecture example with 4 qubits and ring connectivity (qubit 1 is connected to 2, 2 to 3, 3 to 4, and 4 to 1 makes a ring). The first layer of R_X is the encoding layer U_{enc}, taking a data instance $x \in \mathbb{R}^4$ as input. The entangling part with 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.