Molecular Dynamics Workflow Decomposition for Hybrid Classic/Quantum Systems

Sandeep Suresh Cranganore*
Vincenzo De Maio*, Ivona Brandic*
*Institute of Information Systems Engineering
Vienna University of Technology
Vienna, Austria
{sandeep.cranganore,vincenzo.maio,ivona.brandic}@tuwien.ac.at

Tu Mai Anh Do‡
Ewa Deelman‡
‡Information Sciences Institute
University of Southern California
Los Angeles, USA
{tudo,deelman}@isi.edu

***Abstract*—Since we are entering the Post-Moore Law era and consequently the limit of Von Neumann's architecture, the scientific community is looking for alternatives to satisfy the growing computing power demands of scientific applications.**

Quantum computing promises to achieve a computational advantage over the classic Von Neumann architecture. However, the limited capabilities of current noisy intermediate-scale quantum (NISQ) devices require quantum computers to interoperate with classic systems, forming the so-called hybrid quantum systems. Research on hybrid quantum systems led to the design of Variational Quantum Algorithms, currently the most promising way to move towards quantum advantage.

However, execution time and accuracy of variational quantum algorithms are affected by different hyperparameters, including selected cost functions and parametrized quantum circuits. Consequently, providing developers with methods to select the right set of parameters is of paramount importance.

In this work, we provide a formal method for the selection of hyperparameters in variational quantum algorithms, which will support quantum algorithms developers in the design of quantum applications, and evaluate it on a real-world scientific application, showing a reduction of error up to 31%.

I. Introduction

Research on scientific applications and workflows gained significant momentum due to the recent COVID-19 Pandemic. Because of the complexity of scientific applications, they often rely on HPC clusters for their execution. However, since we are entering the Post-Moore law era [1], there is a strong possibility that classical architectures will not be able to meet such high computational demands in the future, calling for research efforts on alternative forms of computing.

Quantum computing has the potential to offer a significant computational advantage over Von-Neumann's architectures, which allows for solving different intractable problems in different application domains (i.e., from finance, molecular dynamics, computational chemistry [2]) and its native modeling of many scientific phenomena [3]. Due to the limited number of resources available and the high noise in their results, quantum processing units (QPUs) are combined with classic architectures, defining so-called Hybrid Quantum Systems [4].

Variational Quantum Algorithms (VQAs) are the most promising way to exploit hybrid quantum systems and achieve so-called quantum advantage [5]. VQAs rely on a parametrized quantum circuit (PQC), i.e., a circuit whose expected value depends on a set of free parameters Θ encoding a solution to a specific problem, and a cost function C on Θ. VQAs aim to iteratively tune parameters in Θ by optimizing $C(\Theta)$ on a classical computer and then computing quantum circuit expectation using Θ. The output of this process is a (close-to) optimal solution to the target problem.

However, VQAs execution is strongly affected by hyperparameters, such as (*i*) PQC selection, (*ii*) cost function and (*iii*) classic optimizer. Considering the growing interest of the scientific community in quantum computing, finding methods to improve the execution of quantum computations is of paramount importance for the development of quantum computing.

In this work, we focus on workflow decomposition for the execution on hybrid quantum systems. We identify workflow tasks that are suitable for execution on quantum machines and design a quantum implementation for selected tasks. Afterward, we collect data about quantum tasks' execution considering different quantum hyperparameters, identifying the setting providing the best accuracy. Finally, we evaluate our method on a molecular dynamics (MD) workflow.

We focus on the computation of inter-atomic distance, for which we design a quantum circuit, and computation of the largest eigenvalue based on Variational Quantum Eigensolver, for which we identify the set of hyperparameters allowing to reduce error in eigenvalues' calculations.

Our results represent a first step in the employment of quantum machines as accelerators for scientific computations. Data collected for VQA execution show that hyperparameter selection can affect accuracy up to 31%, providing insights on the setup of VQAs and showing the necessity for methods supporting quantum developers in selecting hyperparameters.

The paper is organized as follows: first, we introduce notions about quantum computing and VQAs in Section II. Afterward, we describe the target MD use case and hybrid quantum decomposition in Section III, while our method is described in Section IV. Afterwards, we describe experimental setup in Section V, commenting our results in Section VI. Related works are discussed in Section VII, while conclusions and future work are presented in Section VIII.

II. Background

A. Quantum Computing

The basic unit of quantum computation is the *qubit*. In contrast to classic bits, which can be either 0 or 1, a qubit can be in a *superposition* of both. A set of n qubits taken together forms a *quantum register*. Quantum computation is performed by manipulating qubits within a quantum register. The state of a n-qubits register $|\psi\rangle$ is represented as a linear combination (superposition) of n orthonormal basis, denoted as n column vectors, i.e.,

$$\begin{aligned} |0\rangle &\mapsto [1, 0, \ldots, 0]^T \\ |1\rangle &\mapsto [0, 1, \ldots, 0]^T \\ &\vdots \\ |n-1\rangle &\mapsto [0, 0, \ldots, 1]^T. \end{aligned}$$

$|\psi\rangle = \sum_{i=0}^{n-1} c_i |i\rangle$ with suitable *complex* coefficients (weights) $c_0, c_1, \ldots, c_{n-1} \in \mathbb{C}^n$ known as the *complex amplitudes*. The normalization condition for the quantum state $|\psi\rangle$ implies that $\sum_{i=0}^{n-1} |c_i|^2 = 1$. The simplest intuitive visualization of a quantum bit (qubit) is a unit vector living on a so-called *Bloch-sphere*. A qubit $|\psi\rangle$ is a two-level quantum system, described by a two-dimensional complex Hilbert space. It is spanned by two orthonormal states $\{|0\rangle, |1\rangle\}$ and can be written in a consise form in the Bloch-sphere representation [6],

$$|\Psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{1}$$

with $\theta \in [0, \pi] \wedge \phi \in [0, 2\pi]$. Thus, in contrast to a classical bit which can be either 0 or 1, a qubit is an arbitrary point on this sphere taking uncountably infinite values on this sphere.

In constrast to classic registers, a quantum register is in a *superposition* of each state, i.e., can be in each one of $|0\rangle, \ldots |n-1\rangle$ at the same time. At the moment of observation, the probability that we will find it in a particular basis state $|i\rangle$ is $P(|i\rangle) = |\langle i|\psi\rangle|^2 = |c_i|^2$.

Given the probabilistic nature of quantum computation, the quantum computation must be repeated for a given number of times, s, and the result of quantum computation will be the most frequent result over s executions. As a consequence, the goal of quantum computation is to manipulate registers to obtain the solution to the target problem with high probability. Quantum registers are manipulated through *quantum circuits*. Quantum circuit model a quantum computation as a sequence of (1) initialization of qubits, (2) application of specific unitary matrices called quantum *gates*, which can be compared to classical logical gates for quantum computing, (3) measurement of the circuit and (4) resets.

B. Hybrid Quantum Systems

Hybrid Quantum Systems define a class of systems that combine classic and quantum computers to solve a problem. The main advantage of using this approach is that allows the exploitation of the strengths of classic computers for specific tasks (e.g., error correction) and additional capabilities

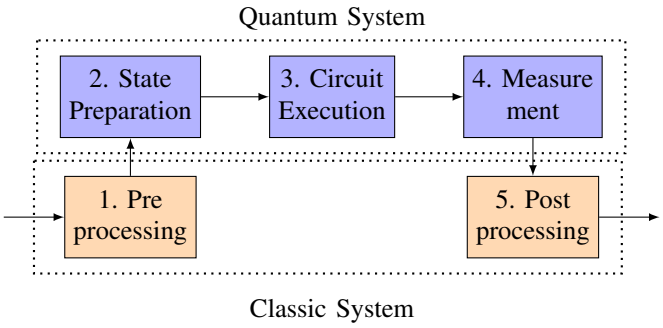


Fig. 1: Hybrid Quantum Systems.

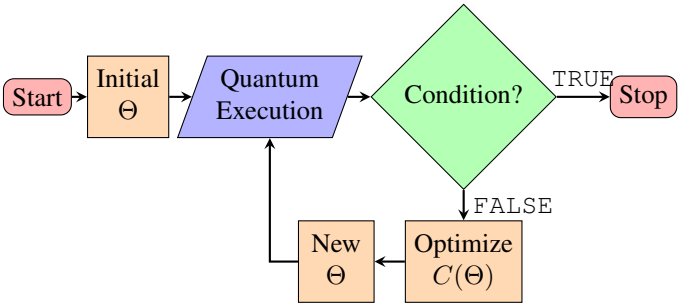


Fig. 2: Variational Quantum Algorithms.

provided by quantum machines (e.g., quantum parallelism) [4]. Hybrid quantum systems are depicted in Figure 1: in step 1, data are pre-processed on the classic system for execution on the quantum system; in step 2, the quantum state is prepared based on preprocessed input, and manipulated in step 3 using quantum circuit modeling required computation; in step 4, the quantum state is measured and post-processed in step 5.

C. Variational Quantum Eigensolver (VQE)

Variational Quantum Algorithms (VQA) are known as the most promising way to achieve quantum advantage [5], as they allow to reduce the number of qubits required by other quantum algorithms and to fully exploit the capabilities of hybrid quantum systems. The main idea of VQAs is to minimize a cost function C representing a specific property of a physical system (e.g., the ground state of a Hamiltonian, H_G). The state of the physical system is modeled by a Parametrized Quantum Circuit (PQC), which is a quantum circuit whose state is determined by a set of input parameters Θ. The main idea is that by minimizing $C(\Theta)$ we will obtain the specific property we are interested in, similarly to Simulated Annealing metaheuristic [7]. Therefore, VQAs are to find the set of parameters Θ^* which minimizes C.

Typical VQA execution is summarized in Figure 2. First, an initial set of parameters Θ is provided in input to a parametrized quantum circuit, which is then executed on the PQC deployed on the quantum machine (step 2). After execution, measurements of quantum execution are performed to verify the termination condition. If the termination condition is `FALSE`, then a new set Θ is identified through the optimization

of C employing a classic optimizer. The process is repeated until the termination condition is `TRUE`.

The most known VQAs are Quantum Approximate Optimization Algorithm (QAOA), which is often used for combinatorial optimization problems, and Variational Quantum Eigensolver (VQE), which is used to compute matrix eigenvalues to find the ground state energy E_G of a Hamiltonian [5]. In this work, we focus on VQE. As in Figure 2, VQAs are defined by (1) the initial Θ state preparation, (2) selected PQC, (3) optimizer, and (4) cost function C. VQE cost function is defined as

$$\Theta^* = \arg\min_{\Theta} C(\Theta) = \langle \psi(\Theta) | H | \psi(\Theta) \rangle , \tag{2}$$

meaning that for each state $|\psi(\Theta)\rangle$ one tries to minimize expectation of H, with $|\psi(\Theta)\rangle = PQC(\Theta)|\psi_i\rangle$. It follows from the variational principle that, $\arg\min_\Theta C(\Theta) \geq E_G$, therefore minimizing $C(\Theta)$ enables us to approximate the ground state of H, E_G, which corresponds to the minimum eigenvalues of H.

III. MD Workflow Decomposition

Here we describe the target MD application used for our method evaluation.

A. Molecular Dynamics

Experiments to study the dynamics of complex structures are usually sophisticated and time-consuming to conduct in the lab. Computer simulations provide a method to validate theoretical models that are not experimentally attainable in reality. Computer simulations help to bridge the gap between theory and experiment and accommodate a better understanding of real-life systems. Molecular dynamics (MD) simulation is a popular model computing the atomic states of a molecular system evolving by observing microscopic interactions between atoms. MD simulation serves as a productive method to observe important processes at atomic resolution for a better understanding of the behaviors of the system. Rather than dealing with the complexity of setting an experiment environment, MD simulation offers a powerful approach to control the configurations of the molecular systems, such as temperature and pressure. Due to these discussed advantages, MD simulations have been extensively applied to various scientific domains of chemistry, material sciences, molecular biology, and drug design.

The biological insight into a molecular system is obtained by analyzing trajectories, which are a time series of coordinate snapshots or frames periodically generated in a specific interval by the simulation during simulation time. The frames comprise of atomic coordinates, which obtained by solving Newton's motion equations for atomic-level interations of the molecular structure. Trajectory analysis allows the user to trace dynamics of the considered molecular system, which serves as a key point in detecting biologically relevant processes, such as protein folding and conformational changes [8].

B. Target Application

Since MD trajectories are high-dimensional objects, trajectories are not suitable for applying direct interpretation. Many atomic motions in the frames are not equally important, such as high-frequency thermal fluctuations are usually of no interest. Therefore, methods of reducing the dimensionality of trajectories or describing frames using a smaller number of variables are commonly employed to transform trajectories into a data format that is easier to analyze. Metadynamics [9] is one example, in which scientists use well-chosen collective variables (CVs) to capture important molecular motions in the region of interest. Technically, a CV is defined as a function of the atomic coordinates in one frame that helps to reconstruct the free-energy surface for enhanced sampling. Since trajectories are reduced to time series of a small number of such CVs, simulated molecular processes are much more amenable to interpretation and further analysis. A CV can be as simple as the distance between two atoms or can involve complex mathematical operations on a large number of atoms. The CV that we will use in this work is the Largest Eigenvalue of the Bipartite Matrix (LEBM).

Specifically, to capture the structural changes between two amino acid segments, instead of considering all atoms composing of the amino acids, we extract only the positions of α-Carbon (Cα) backbone atoms. Those backbone atoms are then used to form a bipartite matrix whose the maximum eigenvalue is a proxy for discovering structural changes in the molecular system. Formally, given two amino acid segments I and J, if d_{ij} is the Euclidean distance between C_α atoms i and j, then the symmetric bipartite matrix $B_{IJ} = [b_{ij}]$ is defined as:

$$b_{ij} = \begin{cases} d_{ij}, & \text{if } i \in I \text{ and } j \in J \\ d_{ij}, & \text{if } i \in J \text{ and } j \in I \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

We note that B_{IJ} is symmetric and has zeroes in its diagonal. Johnston *et al.* [10] showed that the largest eigenvalue of B_{IJ} is an efficient measurement to monitor changes in the conformation of I relative to J.

C. Hybrid Quantum Decomposition

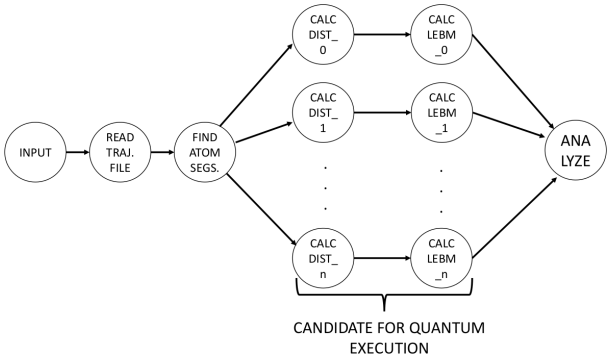


Fig. 3: Target MD Simulation.

Figure 3 visualizes target MD application. After reading input from the user, the application reads a trajectory file, which provides information about the molecule structure, and identifies the atom segments that have to be considered in our calculations. For each pair of atom segments, the application performs parallel computations to calculate B_{IJ} matrices which are used as an input for the LEBM calculation. At the end of this process, results for different B_{IJ} are collected and analyzed. Following from the previous section, we identify that two parts are most suitable for quantum execution: the B_{IJ} and LEBM calculation.

Generating the matrix B_{IJ} involves the computation of Euclidean distances between different atom segments. Exploiting the capability of quantum machines in manipulating large numbers of higher-dimensional vectors and tensors, makes them highly suited for performing vector-based machine learning tasks and operations such as vector dot products, overlaps, norms, etc., in N-dimensional vector spaces.

It was shown in [11] and by S. Aaranson [12] that sampling post-processed vectors and estimating distances or inner products between these post-processed vectors is exponentially hard on a classical computer and has $O(N)$ complexity on a classical system.

Instead, replacing the traditional classical computing codes with quantum circuits would significantly reduce the runtime. This is because, classical data (typically vectors or tensors), expressed in terms of N- dimensional complex valued vectors can be encoded into $log_2 N$ qubits. These mapped vectors can then be stored in a quantum random access memory (qRAM). This mapping can be achieved in $O(log_2 N)$ steps [11]. This stored quantum information can later be post-processed by various quantum algorithms like *Quantum Fourier Transforms* **(QFT)** (cf. [6]) or *matrix inversion techinques*, which take time $O(poly(logN))$. While, estimating distances and inner products between these post-processed vectors can be completed in time $O(logN)$ on a quantum computer [13].

Furthermore, computing the LEBM of the target matrix B_{IJ} can be achieved efficiently using Variational Quantum Eigensolver (VQE) [5]. The VQE circuit then performs the optimization task through a hybrid classical-quantum scheme. Therefore we apply VQE machinery to identify LEBM.

D. Assumptions and Limitations

1) Input size: While on classic architectures the amount of atoms and segments we can process is dependent by the amount of RAM available in the system, in the quantum machines we are limited by the amount of qubits of the machine. In general, to embed a $k \times k$ matrix, $\lceil \log_2 k^2 \rceil$ qubits are required. For this reason, we limit our input, i.e. the length of amino acid segments used to form the bipartite matrix, according to the limitations of target machines. Let n denote the chosen segment length, then size of the bipartite matrix, which is generated as in Equation 3, is $2n \times 2n$.

To compute Euclidean distance, the quantum circuit used for our study comprises of a single execution for each atom pair. Hence, a total number of n repeated circuit executions are required for calculating the distances between n atom pairs using the quantum architechture. Ideally, developing a quantum circuit yielding the Euclidean distances between all the atom pairs on one single execution should also be possible. One such method involves the introduction of an *oracle* (black-box) function that can encode multiple classical vector data [14]. Alternatively, multiple fidelities/innerproducts between the input qubits can be computed using the SWAP-test technology as developed for a quantum neuron model [15].

With respect to 5 available qubits, we consider a bipartite matrix of size 32×32. We evaluate hyperparameters selection on 32×32 off-diagonal block matrices.

The atom coordinates are vectors in $\mathbb{R}^3$. Since each qubit has two possible states, the number of coordinates of the classical vector must necessarily be 2^n [16]. For this reason, the three dimensional vector is padded 0 as the fourth coordinate. This leads to a vector in form $(x, y, z, 0)$ with $2^2 = 4$ coordinates, which can be encoded into a 2-qubit quantum register.

2) Matrix shape: The symmetric bipartite matrix B_{IJ} can be partioned into a block matrix form.

$$B_{IJ} = \begin{pmatrix} 0^{n\times n} & E_{IJ} \\ E_{IJ}^{\intercal} & 0^{n\times n} \end{pmatrix} \tag{4}$$

The diagonal entries of the bipartite block matrix contains the zero matrix. The off-diagonal entries E_{IJ} ($n \times n$ matrix) and its transpose $E_{IJ}^{\intercal}$ contain as entries the Euclidean distances d_{ij} as defined in Eq.(3). The Euclidean distance (metric) is a function defined on vector space $\mathbb{V}$,

$$d : \mathbb{V} \times \mathbb{V} \mapsto \mathbb{R}.$$

Therefore, the block matrix E_{IJ} takes values only over the field $\mathbb{R}$. The matrix representation is given by,

$$E_{IJ} = \begin{pmatrix} d_{ij}^{11} & \cdots & d_{ij}^{1n} \\ \vdots & \ddots & \vdots \\ d_{ij}^{n1} & \cdots & d_{ij}^{nn} \end{pmatrix}. \tag{5}$$

For given two segments I and J, the Euclidean metric between C_α carbon atoms i and j reads,

$$d_{ij} = d(i,j) = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2 + (i_z - j_z)^2}. \tag{6}$$

Lemma 1. *Input matrix B_{IJ} is a **Hermitian** matrix.*

Proof. Let $B_{IJ} \in \mathbb{R}^{2n} \times \mathbb{R}^{2n}$, with $n \in \mathbb{N}$. Note that from Equation 4,

$$B_{IJ}^{\intercal} = \begin{pmatrix} 0^{n\times n} & E_{IJ} \\ E_{IJ}^{\intercal} & 0^{n\times n} \end{pmatrix}^{\intercal} = B_{IJ}$$

Hence, B_{IJ} is a **real symmetric matrix** Since, every real symmetric matrix is a Hermitian matrix, B_{IJ} is Hermitian. ■

Since the bipartite matrix is Hermitian, its eigenspectrum is in $\mathbb{R}$. Therefore, we can use B_{IJ} as the input to the VQE protocol without further transformations, since states of quantum systems (physical observables) described by operators that are necessarily linear and Hermitian matrices.

IV. Proposed Method

A. Generation of Distance Matrix

1) Quantum circuit: The generation of B_{IJ} in our case is achieved using a quantum algorithm called the *SWAP* test. The SWAP-test algorithm is a quantum subroutine that was first introduced in the context of quantum fingerprinting [17] and computes the *fidelity* or overlap between two quantum states in terms of the measurement probability of the control qubit in the state $|0\rangle$. The Fidelity F between two normalized quantum states $|\phi\rangle, |\psi\rangle$ is mathematically expressed as, $F(\phi, \psi) = |\langle\phi|\psi\rangle|^2$ [18].

Hence, higher the fidelity, closer are the quantum states to each other. $F(\phi, \psi) = 0$ means that the quantum states are orthogonal to each other. The SWAP-test algorithm is one such method that can be used as a fidelity estimator of two pure states. The quantum circuit with the amplitude encoded vectors for four atomic-pairs is presented in Figure 4.

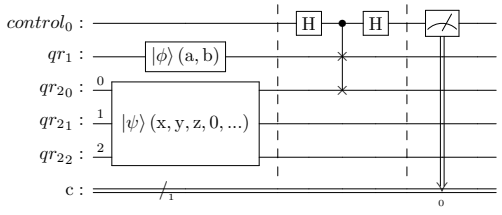


Fig. 4: Visualization of SWAP-test circuit.

This quantum circuit has been used extensively in accelerating the k-nearest neighbor (kNN) classification algorithm in supervised machine learning [19]. We shall implement this quantum circuit to compute the euclidean distances between the C_α atoms and generates the distance matrix B_{IJ}.

Thus quantum algorithms offer a major advantage in working with large complex chains of molecules consisting of large number of atoms, due to the aforementioned speedup

2) Two state SWAP-test circuit description: To apply our circuit, first we initialize an ancillary qubit $|0\rangle$, two quantum registers $|\psi\rangle$ and $|\phi\rangle$. The combined initial state is

$$|\Psi_I\rangle = |0\rangle \otimes |\phi\rangle \otimes |\psi\rangle, \tag{7}$$

where the symbol $\otimes$ denotes the tensor-product (Kronecker product) of the quantum states. The states $|\phi\rangle$ and $|\psi\rangle$ contain the coordinates of the atoms, encoded by means of a method called *Amplitude encoding*.

Definition: For $\mathbf{x} \in \mathbb{R}^n$, amplitude encoding maps $\mathbf{x} \mapsto E(\mathbf{x})$, and reads [20],

$$|x\rangle = \sum_i^n f_i(x)\,|i\rangle,$$

where, $|f_i(x)|^2 = 1$.

Calculation of amplitude corresponds to the *state preparation* phase as described in Figure 1. The qubits are initialized with the state amplitudes (coefficients), in the following manner:

$$|\phi\rangle = \frac{1}{\sqrt{W}}(||\vec{u}||\,|0\rangle - ||\vec{v}||\,|1\rangle), \tag{8a}$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|u, 0\rangle + |v, 1\rangle), \tag{8b}$$

with $||\vec{u}||$, $||\vec{v}||$ being the Euclidean norm of the coordinates of two different atoms and $W = ||\vec{u}||^2 + ||\vec{v}||^2$. Where, the compact notation $|x, y, .., z\rangle$ denotes $|x\rangle \otimes |y\rangle \otimes .... \otimes |z\rangle$.

The amplitude encoded vectors are given by,

$$|u\rangle = \sum_{i=0}^{N-1} \frac{u_i}{||\vec{u}||}\,|i\rangle, \tag{9a}$$

$$|v\rangle = \sum_{i=0}^{N-1} \frac{v_i}{||\vec{v}||}\,|i\rangle \tag{9b}$$

In Figure 4, the 3-qubit quantum register $|\psi\rangle = |qr\_2_0 = 0\rangle \otimes |qr\_2_1 = 0\rangle \otimes |qr\_2_2 = 0\rangle$ is then initialized with the concatenated atom-pair coordinate values $(x1, y1, z1, x2, y2, z2, 0, 0) \in \mathbb{R}^8$.

Afterwards, we apply Hadamard gate H on the ancillary qubit. This is followed by a controlled swap operation, which is performed using the three-qubit Fredkin gate on the other two registers. The ancillary qubit works like control bit. The total state of the system after these two gate operations is,

$$|\Psi_{II}\rangle = \frac{1}{\sqrt{2}}\big(\,|0, \psi, \phi\rangle + |1, \phi, \psi\rangle\,\big). \tag{10}$$

Application of another Hadamard gate on the ancillary qubit $|0\rangle$ yields

$$\frac{1}{2}\,|0\rangle\,(|\phi, \psi\rangle + |\psi, \phi\rangle) + \frac{1}{2}\,|1\rangle\,(|\phi, \psi\rangle + |\psi, \phi\rangle),$$

following from [18]. After application of Hadamard gate, probability of measuring state $'0'$, i.e., of control qubit yields,

$$Pr(0) = \frac{1}{2} + \frac{1}{2}|\,\langle\phi|\psi\rangle\,|^2.$$

Euclidean distances between atoms can be obtained using Equations 8, 9.

$$d(\vec{u}, \vec{v})^2 = 2W|\,\langle\phi|\psi\rangle\,|^2 = 4W(Pr(0) - 0.5). \tag{11}$$

Euclidean distances computed using a quantum machine constitute the matrix elements of the matrix B_{IJ} described in Equation 4.

B. Calculation of LEBM

To calculate LEBM, as defined in Section III-B, we employ VQE as described in Section II-C.

Our goal is to identify LEBM as close as possible to the real value, i.e., the value calculated on a classic machine. However, since VQE performs iterative optimization of a cost function C (Equation 2), there is no guarantee it will reach the optimal value [5]. To this end, we define the error function that we use to evaluate VQE results.

From Section II-C, we notice that VQE execution is defined by different hyperparameters, which affect results of its execution. Therefore, first, we identify the hyperparameters which affect execution of VQE, then we define our data-driven method to identify the most suitable hyperparameters' setting.

1) Error: For a distance matrix B_{IJ}, we define its classic LEBM $\Lambda_c(B_{IJ})$ and its LEBM calculated with VQE using a specific hyperparameter setting Π as $\Lambda_{vqe}(B_{IJ}, \Pi)$. In our case, we compare VQE results with results on classic architectures, using Mean Square Error (MSE). For a set of matrices, $\hat{B} = \{B^0_{IJ}, B^1_{IJ}, \ldots, B^n_{IJ}\}$, we define

$$Err(\hat{B}, \Pi) = \sum_{i \in [0, |\hat{B}|]} \frac{(\Lambda_c(B^i_{IJ}) - \Lambda_{vqe}(B^i_{IJ}, \Pi))^2}{|\hat{B}|} \quad (12)$$

as the MSE between the classic and quantum eigenvalues, calculated using hyperparameters Π.

2) Data-Driven Hyperparameters Selection: Π can be composed of different hyperparameters, such as (1) the PQC $k \in K$, where K is the set of available PQCs; (2) the optimizer $o \in O$, where O is the set of all optimizers; (3) a classic hardware node $m^i_c \in M_c$, where optimization of cost function is performed; (4) $m^i_q \in M_q$; (5) $s \in \mathbb{N}$ which defines how many times VQE is executed on the system to ensure a statistically significant execution.

Our goal is to find a set Π^*_{err} such that minimizes an error function $Err()$, namely

$$\Pi^*_{err} : Err(\hat{B}, \Pi^*) \leq Err(\hat{B}, \Pi) \ \forall \ \Pi. \quad (13)$$

Concerning Π, the machines m^i_c, m^i_q on which we execute our application are decided depending on availability in the system; Also s is usually decided in advance for statistical significance. For this reason, we focus our analysis on k and o, since according to our analysis they are the parameters which mostly influence our problem.

Determining Π^* before execution requires complex error prediction models, which are not available at the time we write; Also, development of error model for quantum machine is a non-trivial issue, due to the fact that noise of quantum gates is context-dependent [21]. For this reason, we decide to employ a data-driven approach. Our approach is summarized in Algorithm 1.

First, in line 2 we generate the input for our target problem, in this case, $N \times N$ matrices, for which we calculate LEBM in line 4. Then, in lines 6-18 we calculate Π^*_{err} by trying different configurations Π on selected machines. In line 7, we initialize PQC considering its physical setup, composed of quantum gates, their connectivity topology and entanglement map, and the number of qubit needed by PQC k, which is equal to $\log_2 N$. In line 11, we compute VQE for matrix B^i_{IJ} using current Π setting. Once all $\Lambda_{vqe}(B^i_{IJ}, \Pi)$ values are calculated, we compute the error using input $Err()$ function, find $Err(\hat{B}, \Pi)$ and compare it with the minimum error err. In case $Err(\hat{B}, \Pi)$ is lower than err, Π becomes the new Π^*_{err}. At the end of the algorithm execution, Π^*_{err} is found.

Algorithm 1 Data-Driven Hyperparameter Selection

Input: PQCs configurations K, optimizer configurations O, classic machine m_c, quantum machine m_q, N matrix size, $Err()$ function.
Output: Π^*

```
1:  err ← ∞
2:  B̂ ← generateRandomMatrices(N)
3:  for i ∈ |B̂| do
4:      Λ_c(B^i_IJ) ← calculateLEBM(B^i_IJ)
5:  end for
6:  for k ∈ K do
7:      k ← initPQC(k, log2 N, m_q)
8:      for o ∈ O do
9:          Π ← {k, o}
10:         for i ∈ |B̂| do
11:             Λ_vqe(B^i_IJ) ← VQE(B^i_IJ, m_c, m_q, Π)
12:         end for
13:         if Err(B̂, Π) < err then
14:             Π*_err ← Π
15:             err ← Err(B̂, Π)
16:         end if
17:     end for
18: end for
19: return Π*_err
```

V. Experimental Setup

A. Configuration Selection

For an accurate training of Hyperparameter Optimization, we have to collect data about error Err and convergence time T using different parameters. We collect data from different quantum machines, variating hyperparameters and evaluating results in comparison with classic execution. We focus on two main hyperparameters: PQC and optimizer. Once data have been collected, we validate our method on the MD application described in Section III.

1) PQC: We describe here the PQCs selected for our evaluation. PQCs are selected among Qiskit circuit library [1]. Each PQC is defined by three parameters: *width*, i.e. the number of qubits needed by the PQC, which is dependent from the size of input matrix (i.e., for a $n \times n$ matrix, $log_2 n$ qubits are needed); the *repetitions*, which define how much times the circuit is repeated in serie, and the *entanglement*, which define how the qubits are entangled at the end of each circuit repetition. There are four types of entanglement: *full*, where each qubit is entangled with all the others; *linear*, where each qubit i is entangled to $i + 1$ qubit for all $i \in [0, \ldots, n - 2]$, where n is the number of qubits; *circular* is linear entanglement, but with an additional entanglement between n and 0 qubits; and *shifted-circular-alternating* (SCA), proposed in [22], where entanglement between 0 and n is shifted by one for each repetition and role of control and target qubits are swapped at each layer.

[1] https://qiskit.org/documentation/apidoc/circuit_library.html

PQC	Width	Repetitions	Entanglement
`SU2`	$[1,5]$	$[1,5]$	{full, linear, SCA, circular}
`RealAmplitudes`	$[1,5]$	$[1,5]$	{full, linear, SCA}
`PauliTwo`	$[1,5]$	$[1,5]$	{linear}
`ExcitationPreserving`	$[1,5]$	$[1,5]$	{full, linear, SCA}

TABLE I: Target PQCs

Node id	Version	Processor	Qubits
ibmq_manila	1.0.29	`Falcon r5.11L`	5
ibmq_santiago	1.4.1	`Falcon r4L`	5
ibm_lagos	1.0.27	`Falcon r5.11H`	7
ibm_jakarta	1.0.33	`Falcon r5.11H`	7

TABLE II: Selected Quantum Machines.

a) SU2: Circuit SU2 is summarized in Figure 5. For each qubit, two rotations are performed, applying gate $\mathrm{R_Y}$ and $\mathrm{R_Z}$ in serie, depending on input Θ. Results are then entangled according to the chosen entanglement protocol for the number of block repetition performed.

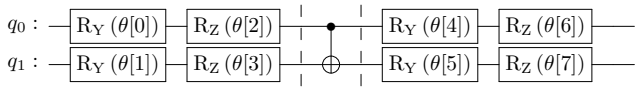


Fig. 5: SU2, (width=2, repetitions=1)

b) RealAmplitudes (RA): RA is visualized in Figure 6 and it is often used as Ansatz in chemistry applications. It is composed of layers of rotations on the Y axis, entangled according to the chosen entanglement protocol.

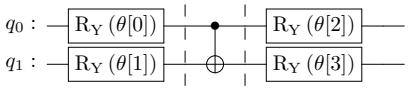


Fig. 6: RA, (width=2, repetitions=1).

c) Pauli Two-Design (P2): P2 has been proposed in [23] and it is often used in quantum machine learning literature [24]. As in Figure 7, it is composed of layers of rotations and entanglements, where first rotation before performed on the Y axis with $\frac{\pi}{4}$ as parameter, while the subsequent rotation on Z is based on input θ.

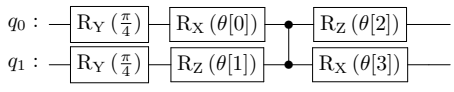


Fig. 7: P2, (width=2, repetitions=1).

d) ExcitationPreserving (ExP): ExP PQC, visualized in Figure 8, consists of layers of Z rotations and $2-$qubits entanglement. Entanglement is defined by selected entanglement protocol.

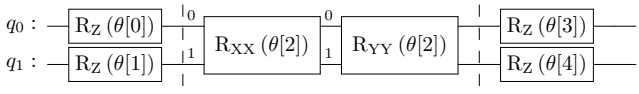


Fig. 8: ExP, (width=2, repetitions=1).

2) Optimizer: Similarly to [25], we select three off-the-shelf optimizers for our evaluation:

a) Constrained Optimization BY Linear Approximations (COBYLA): COBYLA performs linear approximations of both target and constraints function, aiming at optimizing a simplex within a trust region of the parameter space.

b) SPSA: SPSA is a stochastic optimization methods focusing on measurement of objective function. Gradient approximation is performed by taking two measurements of the objective function per iteration, regardless of the problem size.

c) Gradient Descent: Minimizes target objective function f by iteratively moving in the direction of steepest descent, defined by the negative of the gradient. Gradient is updated according to the learning factor, which we set to 0.1 in our experiments.

B. Target machines

Our goal is to select the hyperparameters that allow to reduce error of VQE on different machines. To this end, we collect execution data of VQE on different machines for different hyperparameters configuration. Details of machines selected for our experiments are summarized in Table II. Machines are provided by the IBM Quantum Researchers program (http://quantum-computing.ibm.com) and feature different number of qubits and noise models. We simulate different machines using Qiskit `aer_simulator` backend, to which we provide the noise model of target machine as input.

C. Software APIs

We implement our benchmarks using Python 3.9 and IBM Qiskit 0.36. The choice of Qiskit for our evaluation, as it allows to exploit state-of-the-art IBM machines and IBM Quantum Runtime API. Also, it provides implementation of VQE and allows to set up all parameters needed by our benchmarks, including state preparation, PQC and optimizer. Analysis of data is performed using scikit-learn 1.02 functions for MSE. Experiments' source code is available at the repository https://github.com/vindem/quantumMD.

VI. Results

We evaluate our approach in different ways: first, we focus on generation of distance matrix, which is performed by means of C-SWAP test described in Section IV-A. Afterwards, we evaluate calculation of LEBM described in Section IV-B. Finally, we integrate both approaches in target application described in Section III-B.

Node id	Average MSE	Variance
ibmq_manila	0.2317	0.000199
ibmq_santiago	0.2832	0.000264
ibm_lagos	0.2249	0.000190
ibm_jakarta	0.2037	0.000149

TABLE III: CSWAP MSE on Different Machines.

A. Distance Matrix Generation

To evaluate generation of distance matrix, we calculate the MSE between d_{ij}^Q, i.e., d_{ij} calculated with CSWAP test and d_{ij}. The input for CSWAP test are segments of different sizes (number of atoms), over which we calculate Euclidean distance as specified by Equation 6. Once we calculate Euclidean distance both on classic and quantum machines, we generate two matrices: B_{IJ}^Q, which is the matrix B_{IJ}, as defined in Equation 3, but with Euclidean distances calculated on the quantum machine, and the classic counterpart, defined as B_{IJ}. Finally, we calculate the MSE between correspondings b_{ij} in each matrix. MSE is calcualted between 100 pairs of randomly generated matrices, variating segment size in the set $\{1, 2, 4, 8, 16\}$. Table III shows result of evaluation for generation of distance matrix on quantum machines described in Table II. We show the average MSE, calculated between MSE with different segment sizes. Higher MSE for ibmq_santiago is probably dependent by the older quantum processor in comparison with other machines. The variance of MSE also shows that MSE is not affected by different segment sizes.

Results show that CSWAP results are very close to the classic values, with average MSE between $0.2 - 0.3$ in all the examined cases. We also observe that MSE does not change, regardless the increasing number of qubit required by the quantum circuit to perform calculation on segments of increasing sizes.

B. Hyperparameters Selection

Benchmarking results for VQE are visualized in Figures 9-11. For brevity, we focus only on the most important results, while complete experiment data are available at Zenodo repository[2]. We show the value of MSE between the classic and the quantum value, with respect to the number of qubits, since higher number of qubits introduce a higher noise in the system. In the process of our experiments, we noticed that variating optimizer has minimum effect on VQE results, which is mostly affected by the selected PQC. Regarding PQC, as shown by Figure 10, the lower MSE is achieved by using linear entanglement, therefore we show only results for linear entanglement and `COBYLA` optimizer for brevity. Also, there is no correlation between number of circuit repetitions ($reps$) and reduction of MSE, since, as shown by Figure 11, for each PQC there is a value of $reps$, which we call $reps^*$, which guarantees the minimum value for MSE, and increasing this value does not provide any improvement in MSE. Actually, while some PQCs seem to achieve lower MSE with higher number of repetitions (i.e., P2, RA in Figures 11b, 11c), others achieve lower MSE at higher $reps$, others (S2, Figure 11a) have lower MSE for lower $reps$, while ExP MSE seems to not be affected at all by $reps$ parameter (Figure 11d). As a consequence, in Figure 9, we set the value of $reps$ to $reps^*$ for each PQC. From our benchmarks we can see that the PQC is the parameter which mostly impact the error, with different values of $reps^*$ depending on the amount of qubits of VQE input, which depends on the size of the input matrix.

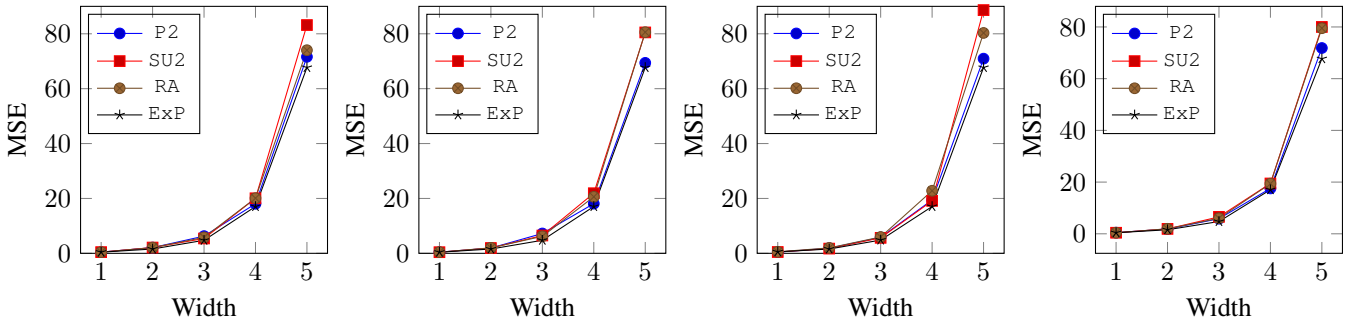
Based on these results, we plug our calculation of VQE inside target MD application. To validate our approach, we collect execution data for selected PQCs on `ibmq_qasm_simulator` backend to identify Π_{mse}^* and examine its convergence to the classic value with increasing number of iterations of different optimizers. We also evaluate MSE for different Π configuration. Results are shown in Figure 12. In Figure 12a, we observe convergency of LEBM calculation with increasing number of iterations for selected optimizers. While all algorithms are converging, we observe a slightly faster convergence for COBYLA in this scenario, therefore we employ this optimizer for evaluation of hyperparameter selection. Results for hyperparameter selection in target use case are visualized in Figure 12b, where we compare the results of hyperparameter selections Π_{mse}^* with other configurations. We can notice that results of Π_{mse}^* affect value of MSE up to 33%, showing the necessity of accurate selection of hyperparameters to improve VQE convergence.

VII. Related Work

Quantum computing and its applications to scientific computing have attracted lot of interest from the scientific community. [26] is one of the first works considering the use of quantum computers for MD simulations. More precisely, in [2] possible advantages of application of quantum computing for modelling of scientific systems, including the native 3D modelling offered by quantum computing. Also, in [27] it is discussed how many problems inherent to MD simulations can be more efficiently solved by applying quantum computing methods. More focused on MD simulation, [28] provides a quantum computing perspective and discusses advantages of VQAs for MD simulations. In [29], ACSE, a quantum computing method for calculation of eigenvalues in MD simulations is proposed, without considering VQE. Also, in addition to eigenvalue calculation, we consider also calculation of inter-atomic distances applying CSWAP-test, which has been succesfully used in Quantum-ML applications such as k-Neareast Neighbor [19]. Similar to our work, [30] provides an implementation of MD simulation on IBM quantum machines, using VQE to compute ground state of the molecular system, without examining VQE hyperparameters. In this work, we focus on decomposition of a scientific workflow on a hybrid system, identifying tasks that could benefit from execution on quantum hardware and design methods to perform execution of these tasks on quantum machines.

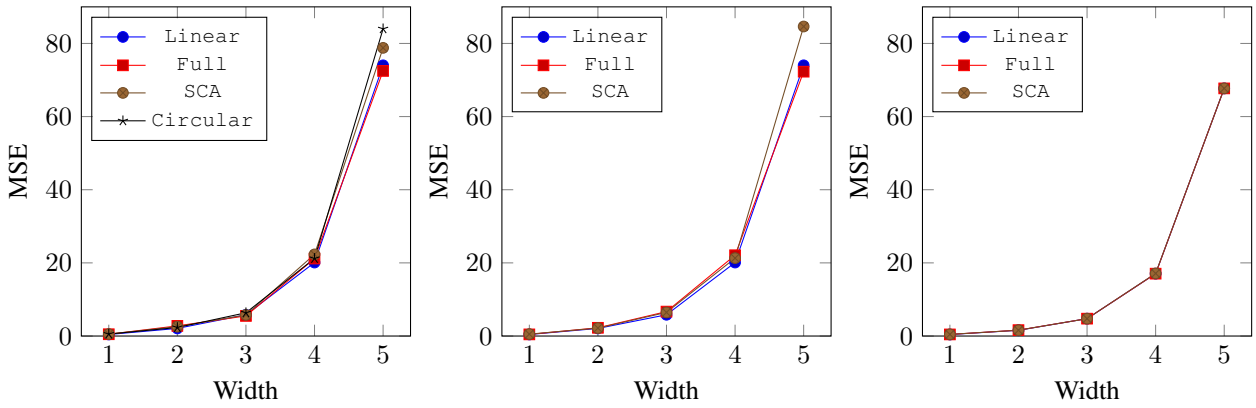
A thorough discussion about VQAs and their applications is given in [5]. However, this paper does not provide insights

[2]https://zenodo.org/record/6477732#.YpdEhzlByWg



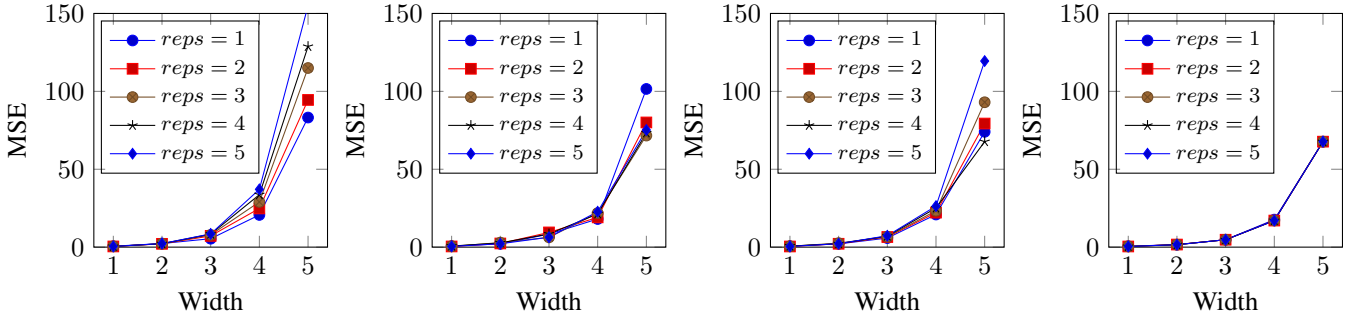
(a) ibm_jakarta (b) ibm_lagos (c) ibmq_manila (d) ibmq_santiago

Fig. 9: MSE w.r.t. Qubits, $m_q \in$ Table II. $K = \{\texttt{P2}, \texttt{SU2}, \texttt{RA}, \texttt{ExP}\}$, $o = \texttt{COBYLA}$, $reps = reps^*$, linear entanglement.



(a) SU2 (b) RA (c) ExP

Fig. 10: MSE w.r.t. Qubits, m_q=ibm_jakarta, o = COBYLA, $reps$=1, ent={Linear, Full, SCA, Circular}



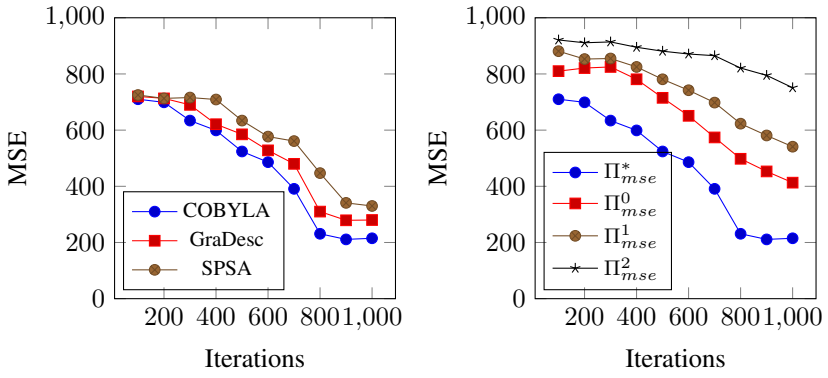
(a) SU2 (b) P2 (c) RA (d) ExP

Fig. 11: MSE w.r.t. Qubits, $m_q \in \{ibm\_jakarta\}$. $K = \{\texttt{P2}, \texttt{SU2}, \texttt{RA}, \texttt{ExP}\}$, $o = \texttt{COBYLA}$, $reps = [1, 5]$, linear entanglement.

on hyperparameters' selection and methods to improve VQA execution. Concerning benchmarking of VQAs, [31] Quantum Approximate Optimization Algorithm algorithm is analyzed in detail on different applications, focusing however more on transpilation-related parameters (i.e., number of CNOT gates of transpiled circuit) than on higher level metrics, i.e., runtime or accuracy. Very similarly to this work, in [25] a study about optimizers' hyperparameters for different VQAs is performed, which we further extend by considering different optimizers and also quantum-hardware related parameters, such as PQCs. A thorough analysis of VQE is provided in [32]. In [33], a method to improve VQE using adiabatic quantum computing is discussed. Measurement-based methods to improve VQE are instead discussed in [34]. Our work provides a model of execution for VQE and identifies the main hyperparameters affecting its execution.

VIII. Conclusion and Future Work

In this paper, we describe the first steps towards execution of scientific applications in hybrid quantum systems. First,



(a) MSE w.r.t. Optimizers' Iterations. (b) MSE w.r.t. Optimizers' Iterations, Π.

Fig. 12: Evaluation on Target Use Case.

we identify tasks that can be executed on quantum machines and guarantee a future quantum advantage. Based on that, we design methods to execute selected tasks on quantum machines and perform extensive evaluation of proposed methods. Results of our evaluation and data of execution on quantum machines are available online and could provide a valuable support to computer scientists which are interested in working with quantum machines. Finally, we plug identified methods into a real MD application, showing the feasibility of our approach.

In future, we plan to include a wider range of hyperparameters, which might require to employ more sophisticated methods for hyperparameter selections, i.e., Bayesian Optimization or Grid Search. Also, we will investigate AI-assisted compilation of quantum circuits to improve mapping on target quantum hardware. Finally, we plan to design methods for software-assisted design of VQAs and PQCs, to support future quantum developers in the use of hybrid quantum systems.

Acknowledgment

We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team. This work has been partially funded through the Rucon project (Runtime Control in Multi Clouds), Austrian Science Fund (FWF): Y904-N31 START-Programm 2015 and by the CHIST-ERA grant CHIST-ERA-19-CES-005, by the Austrian Science Fund (FWF): I 5201-N.

References

[1] G. M. Ștefan, "Let's consider moore's law in its entirety," in *2021 International Semiconductor Conference (CAS)*, 2021, pp. 3–10.

[2] S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, and R. Buyya, "Quantum computing: A taxonomy, systematic review and future directions," *Software: Practice and Experience*, vol. 52, no. 1, pp. 66–114, 2022.

[3] W.-L. Chang, J.-C. Chen, W.-Y. Chung, C.-Y. Hsiao, R. Wong, and A. V. Vasilakos, "Quantum speedup and mathematical solutions of implementing bio-molecular solutions for the independent set problem on ibm quantum computers," *IEEE Transactions on NanoBioscience*, vol. 20, no. 3, pp. 354–376, 2021.

[4] S. A. Stein, R. L'Abbate, W. Mu, Y. Liu, B. Baheri, Y. Mao, G. Qiang, A. Li, and B. Fang, "A hybrid system for learning classical data in quantum states," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2021, pp. 1–7.

[5] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[7] C. Venkateswaran, M. Ramachandran, K. Ramu, V. Prasanth, and G. Mathivanan, "Application of simulated annealing in various field," *Materials and its Characterization*, vol. 1, no. 1, pp. 01–08, 2022.

[8] A. Hospital, J. R. Goñi, M. Orozco, and J. L. Gelpí, "Molecular dynamics simulations: advances and applications," *Advances and applications in bioinformatics and chemistry : AABC*, vol. 8, pp. 37–47, 11 2015. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/26604800

[9] A. Barducci, M. Bonomi, and M. Parrinello, "Metadynamics," *WIREs Computational Molecular Science*, vol. 1, no. 5, pp. 826–843, 2011. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.31

[10] T. Johnston, B. Zhang, A. Liwo, S. Crivelli, and M. Taufer, "In situ data analytics and indexing of protein trajectories," *Journal of Computational Chemistry*, vol. 38, no. 16, pp. 1419–1430, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.24729

[11] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Phys. Rev. Lett.*, vol. 100, p. 160501, Apr 2008. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.100.160501

[12] S. Aaronson, "Bqp and the polynomial hierarchy," 2009. [Online]. Available: https://arxiv.org/abs/0910.4698

[13] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," 2013. [Online]. Available: https://arxiv.org/abs/1307.0411

[14] A. Basheer, A. Afham, and S. K. Goyal, "Quantum k-nearest neighbors algorithm," 2020. [Online]. Available: https://arxiv.org/abs/2003.09187

[15] P. Li and B. Wang, "Quantum neural networks model based on swap test and phase estimation," *Neural Networks*, vol. 130, pp. 152–164, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608020302446

[16] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, 1st ed. Springer Publishing Company, Incorporated, 2018.

[17] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, "Quantum fingerprinting," *Phys. Rev. Lett.*, vol. 87, p. 167902, Sep 2001. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.87.167902

[18] D. Kopczyk, "Quantum machine learning for data scientists," 2018. [Online]. Available: https://arxiv.org/abs/1804.10068

[19] N. Wiebe, A. Kapoor, and K. M. Svore, "Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning," *Quantum Info. Comput.*, vol. 15, no. 3–4, p. 316–356, mar 2015.

[20] R. LaRose and B. Coyle, "Robust data encodings for quantum

classifiers," *Phys. Rev. A*, vol. 102, p. 032420, Sep 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.102.032420

[21] S. Resch and U. R. Karpuzcu, "Benchmarking quantum computers and the impact of quantum noise," 2019.

[22] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, oct 2019.

[23] Y. Nakata, C. Hirche, C. Morgan, and A. Winter, "Unitary 2-designs from random x-and z-diagonal unitaries," *Journal of Mathematical Physics*, vol. 58, no. 5, p. 052203, 2017.

[24] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, pp. 1–6, 2018.

[25] X. Bonet-Monroig, H. Wang, D. Vermetten, B. Senjean, C. Moussa, T. Bäck, V. Dunjko, and T. E. O'Brien, "Performance comparison of optimization methods on variational quantum algorithms," 2021.

[26] B. M. Boghosian and W. Taylor, "Simulating quantum mechanics on a quantum computer," *Physica D: Nonlinear Phenomena*, vol. 120, pp. 30–42, 1998.

[27] H. Liu, G. H. Low, D. S. Steiger, T. Häner, M. Reiher, and M. Troyer, "Prospects of quantum computing for molecular sciences," *Materials Theory*, vol. 6, p. 11.

[28] P. J. Ollitrault, A. Miessen, and I. Tavernelli, "Molecular quantum dynamics: A quantum computing perspective," *Accounts of Chemical Research*, vol. 54, no. 23, pp. 4229–4238, 2021.

[29] S. E. Smart and D. A. Mazziotti, "Quantum solver of contracted eigenvalue equations for scalable molecular simulations on quantum computing devices," *Phys. Rev. Lett.*, vol. 126, p. 070504, Feb 2021.

[30] D. A. Fedorov, M. J. Otten, S. K. Gray, and Y. Alexeev, "iab initio/i molecular dynamics on quantum computers," *The Journal of Chemical Physics*, vol. 154, no. 16, p. 164103, apr 2021.

[31] M. Fellner, K. Ender, R. ter Hoeven, and W. Lechner, "Parity quantum optimization: Benchmarks," 2021. [Online]. Available: https://arxiv.org/abs/2105.06240

[32] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, and J. Tennyson, "The variational quantum eigensolver: a review of methods and best practices," 2021. [Online]. Available: https://arxiv.org/abs/2111.05176

[33] S. M. Harwood, D. Trenev, S. T. Stober, P. Barkoutsos, T. P. Gujarati, S. Mostame, and D. Greenberg, "Improving the variational quantum eigensolver using variational adiabatic quantum computing," *ACM Transactions on Quantum Computing*, vol. 3, no. 1, jan 2022.

[34] R. R. Ferguson, L. Dellantonio, A. A. Balushi, K. Jansen, W. Dür, and C. A. Muschik, "Measurement-based variational quantum eigensolver," *Phys. Rev. Lett.*, vol. 126, p. 220501, Jun 2021.